Lab Report – Mininet and OpenFlow

Boubacar DIALLO

October 6, 2025

Contents

1 Prerequisites

- A host machine running **VMware Workstation**.
- The **Mininet 2.3.0 (minimal)** image downloaded from GitHub: `https://github.com/mininet/mininet/releases/tag/2.3.0`
- Internet connection to install additional packages.
- **VMware** for virtualization.
- **Overleaf** for writing and formatting this report.

2 Useful Commands

- `Ctrl + L` : clear the terminal.

3 Part A: Setting up the Virtual Machine

3.1 Importing the VM

- Import the Mininet OVA image into VMware.
- Configure the network mode: either NAT or Bridged depending on your setup.

3.2 Initial Access

- Default credentials: `mininet / mininet`.
- Access is provided through the text-based shell console.

3.3 Adding a Graphical Interface

```
sudo apt-get update && sudo apt-get upgrade -y
sudo apt-get install -y xinit lxde open-vm-tools open-vm-tools-desktop
```

After reboot, the LXDE desktop environment becomes available.

4 Part B: Experimenting with Mininet

4.1 Displaying Help

The following command displays Mininet's startup options (topology, controller, switch type, tests, verbosity, etc.):

```
sudo mn -h
```

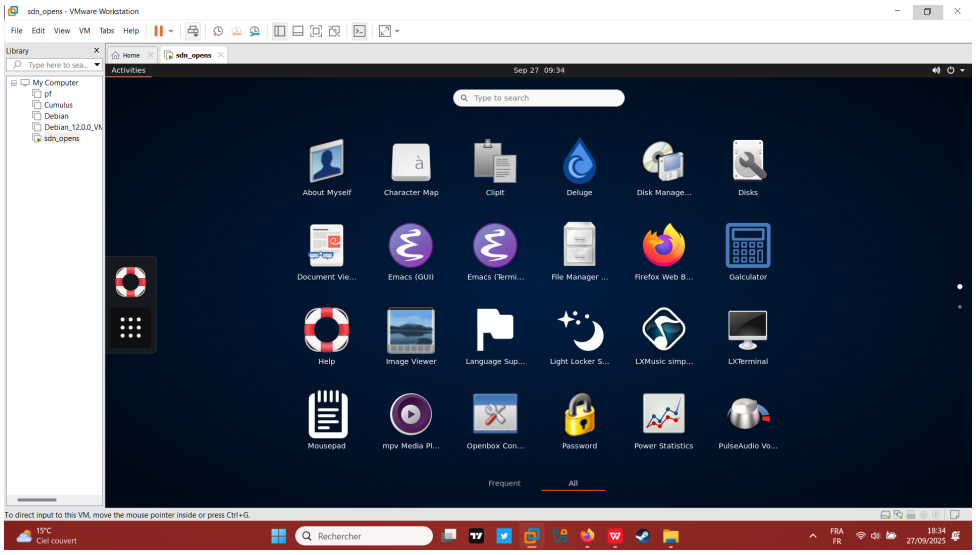


Figure 1: Mininet environment with LXDE desktop on VMware

4.1.1 Initial Traffic Observation

No OpenFlow packets appear before Mininet starts. However, other network protocols are visible:

- **ARP**: Resolves IP to MAC addresses.
- **NTP**: Synchronizes time with a time server.
- **mDNS**: Local service discovery protocol.

These packets come from the host system, not Mininet itself.

5 Part B.1 – Interacting with Hosts and Switches

5.1 Basic Topology

```
sudo mn
```

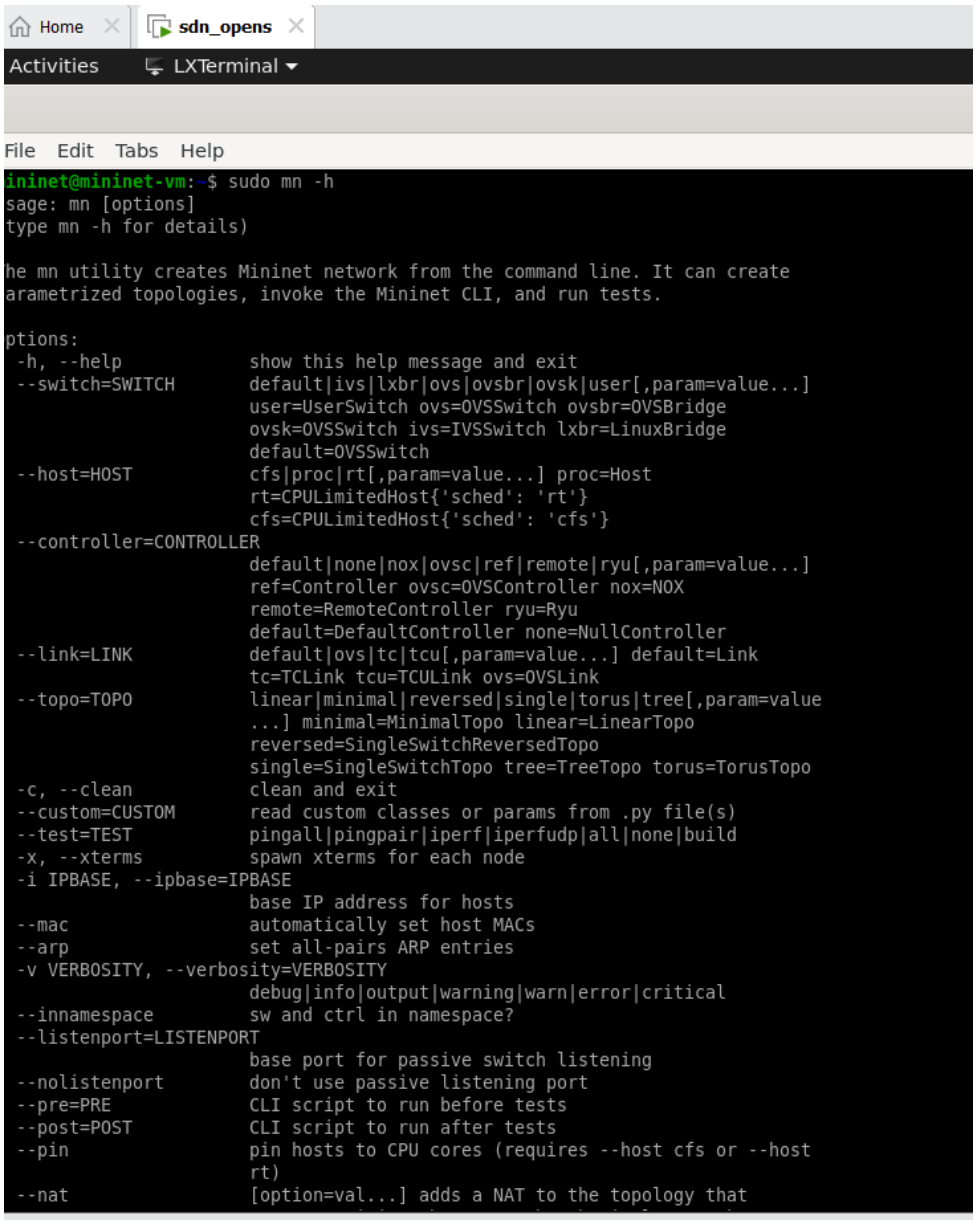
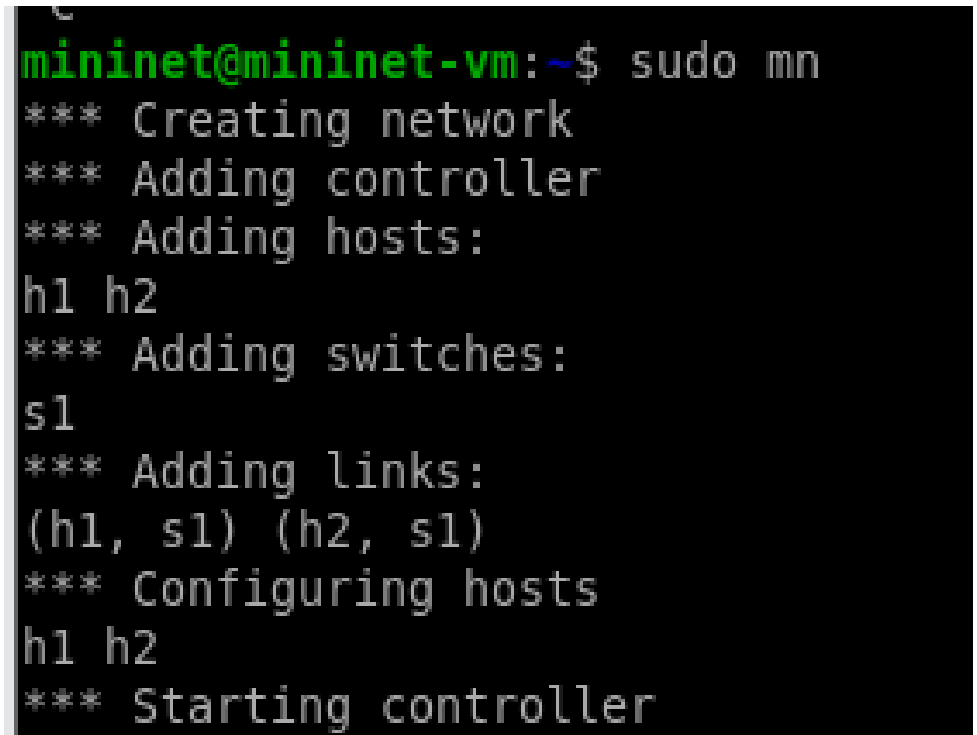


Figure 2: Mininet help options



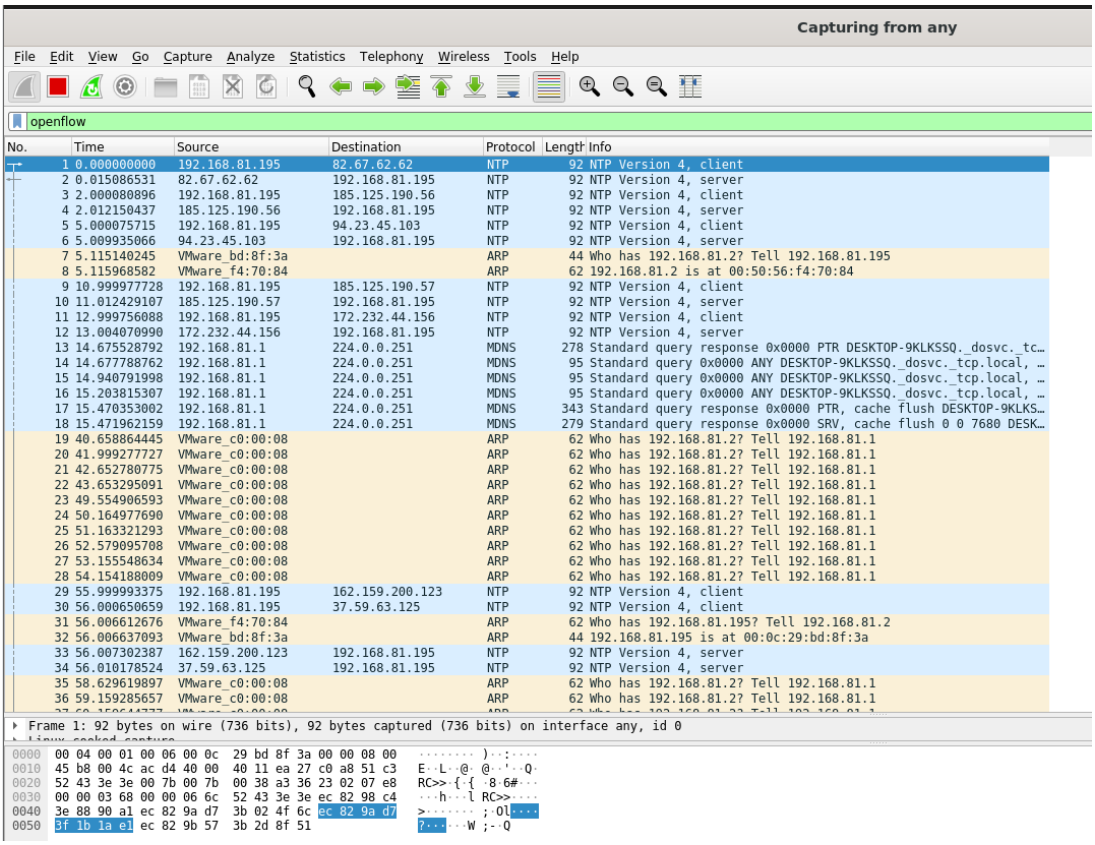


Figure 3: Wireshark capture filtered with `openflow`

5.1.1 Exploring the Topology

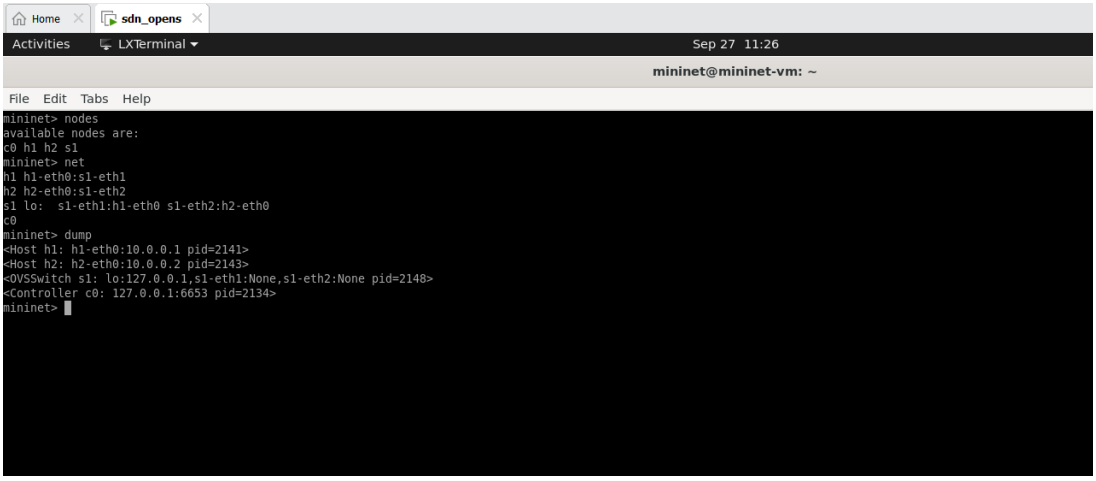


Figure 5: Results of `nodes`, `net`, and `dump` commands in Mininet

`nodes` lists all entities (hosts, switches, controller). `net` displays the links between nodes. `dump` provides details about each node (interfaces, IP, MAC). Each host has its own isolated network namespace.

5.1.2 Visible Processes

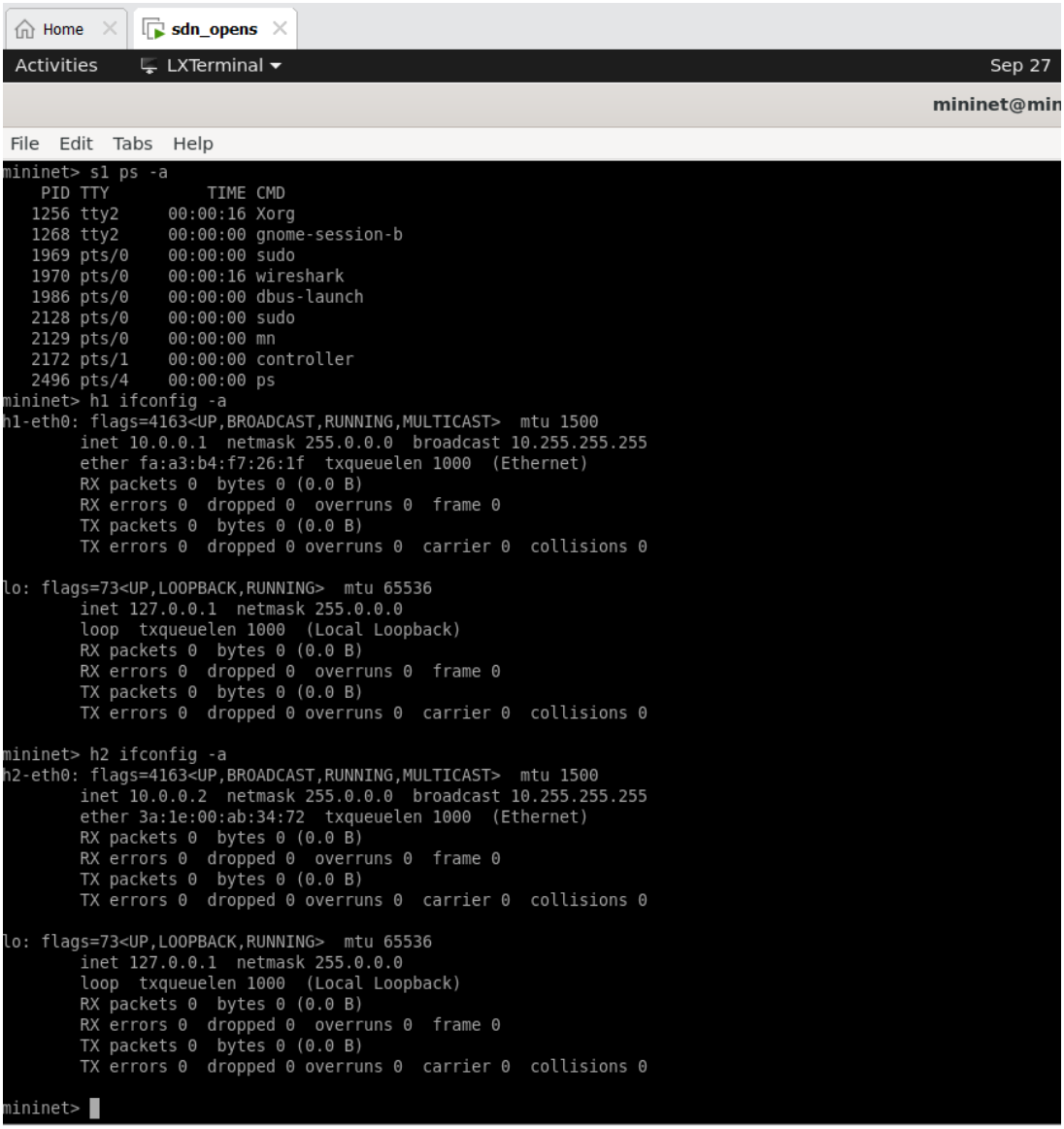


Figure 6: Results of `ps -a` from `h1` and `s1`

The same process list appears in both `h1` and `s1`, meaning that only the network is virtualized; processes and the filesystem are shared among all nodes.

5.1.3 Connectivity Test

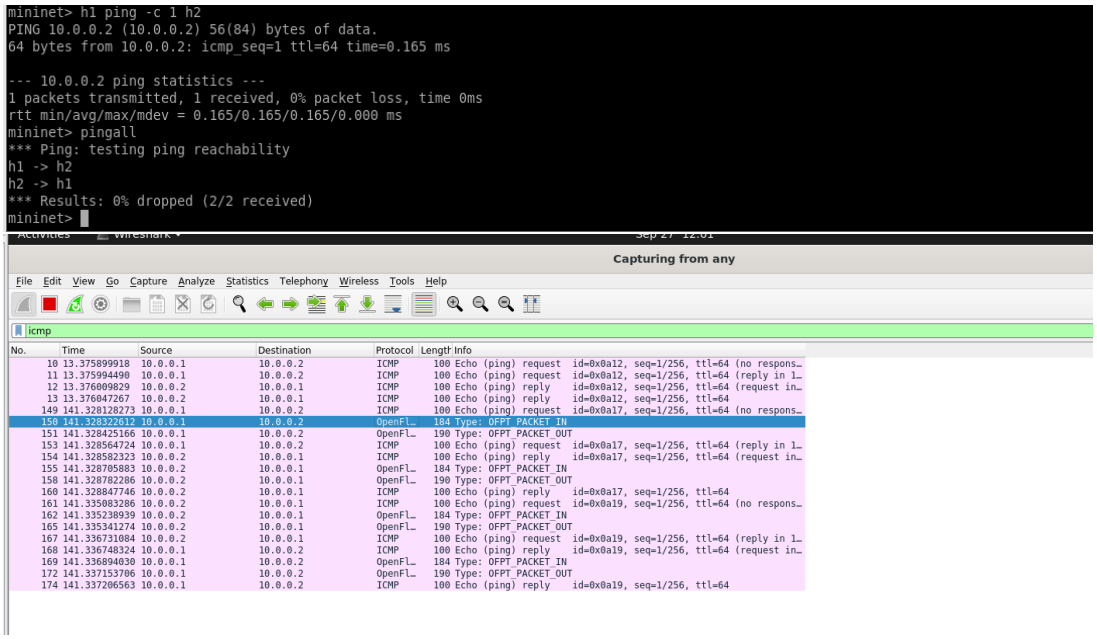


Figure 7: Connectivity test using `ping` and `pingall`

The first `ping` is slower because the controller installs OpenFlow rules (Packet_In / Packet_Out). The second one is faster since flows are already in place. `pingall` confirms full connectivity between hosts.

Figure 8: Application test: HTTP server on h1 and request from h2

This demonstrates that Mininet hosts can run real Linux applications. A simple HTTP server was launched on `h1` (`python3 -m http.server 80`), and `h2` retrieved data with `wget`. The server was stopped using `h1 kill %python`.

5.2 Topologies `single,3` and `linear,4`

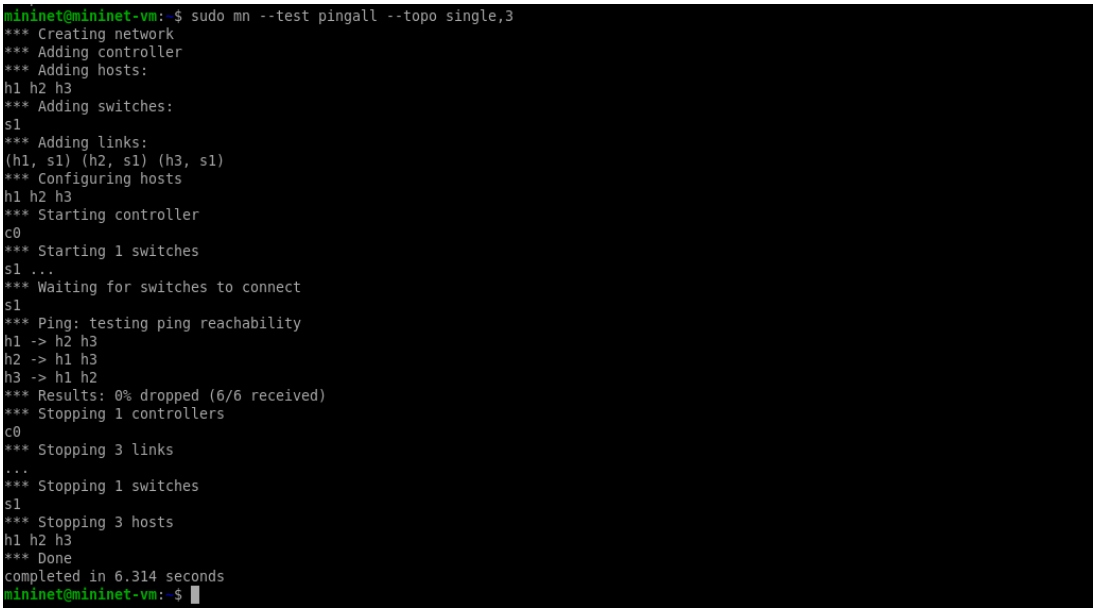


Figure 9: Single switch topology with 3 hosts (`single,3`)

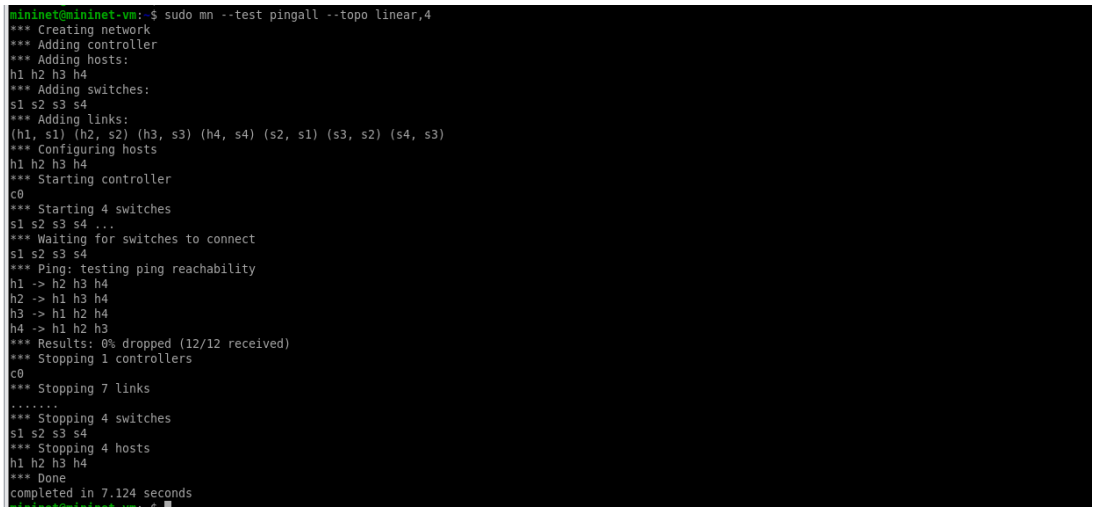


Figure 10: Linear topology with 4 switches (`linear,4`)

Mininet's built-in topologies include `minimal`, `single,n`, `linear,n`, and `tree,depth,fanout`. With the `-test pingall` option, Mininet creates the topology, runs the test, and automatically cleans up.

5.3 Custom Topology with Controller

This topology defines hosts (`h3`, `h4`, `h5`) connected to two switches (`s1`, `s2`) linked together and managed by a controller.

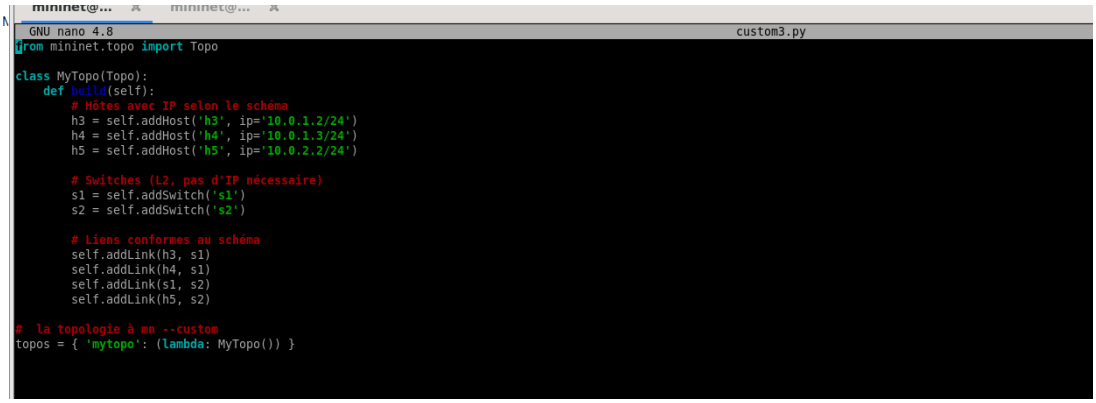


Figure 11: Custom topology script `custom.py`

5.4 POX Controller

The following command starts the POX controller with layer-2 learning and OpenFlow 1.0 support:

```
./pox.py log.level --DEBUG forwarding.l2_learning openflow.of_01 --port=663
```

- `log.level -DEBUG`: enables detailed logs.
- `forwarding.l2_learning` : *simplelearningswitchbehavior.*
- `openflow.of_01` : *OpenFlow*1.0*protocolsupport.*
- `-port=6633: TCP port used by switches to connect.`

The controller installs flow rules automatically, allowing communication between hosts (e.g. `h3 ping h4`).

5.5 Controller Results

Figure 12: Successful ping between h3 and h4 under POX controller

5.6 Fixed MAC Addresses

By default, hosts receive random MAC addresses, which complicates debugging. The `-mac` option assigns deterministic addresses (e.g., `h1 = 00:00:00:00:00:01`, `h2 = 00:00:00:00:00:02`).

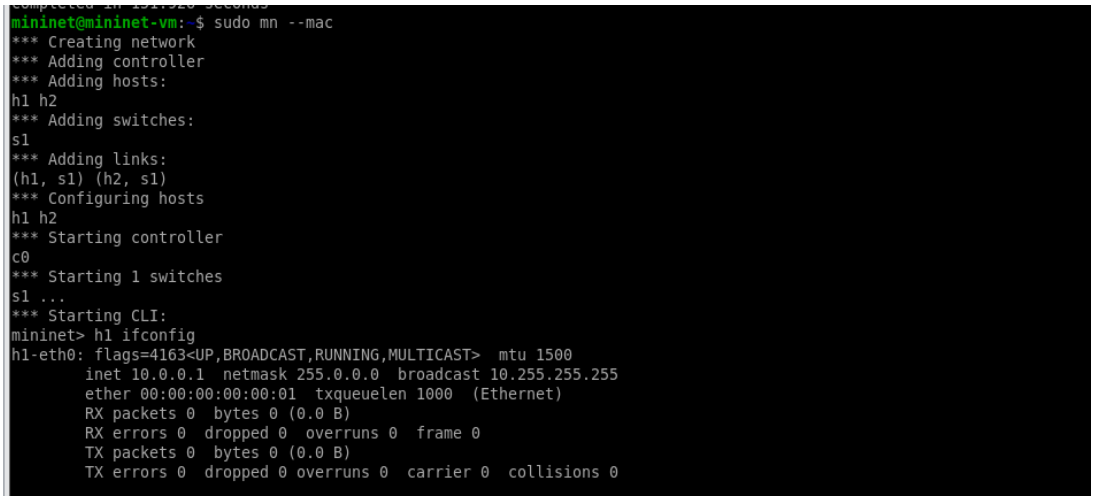


Figure 13: Deterministic MAC addresses with `-mac`

5.7 Link Parameter Variation (Part B.5)

The `b5` topology includes two hosts connected to a switch with constraints: `h1` limited to 20% CPU and `h2` connected via a 10 Mbps link with 50 ms delay.

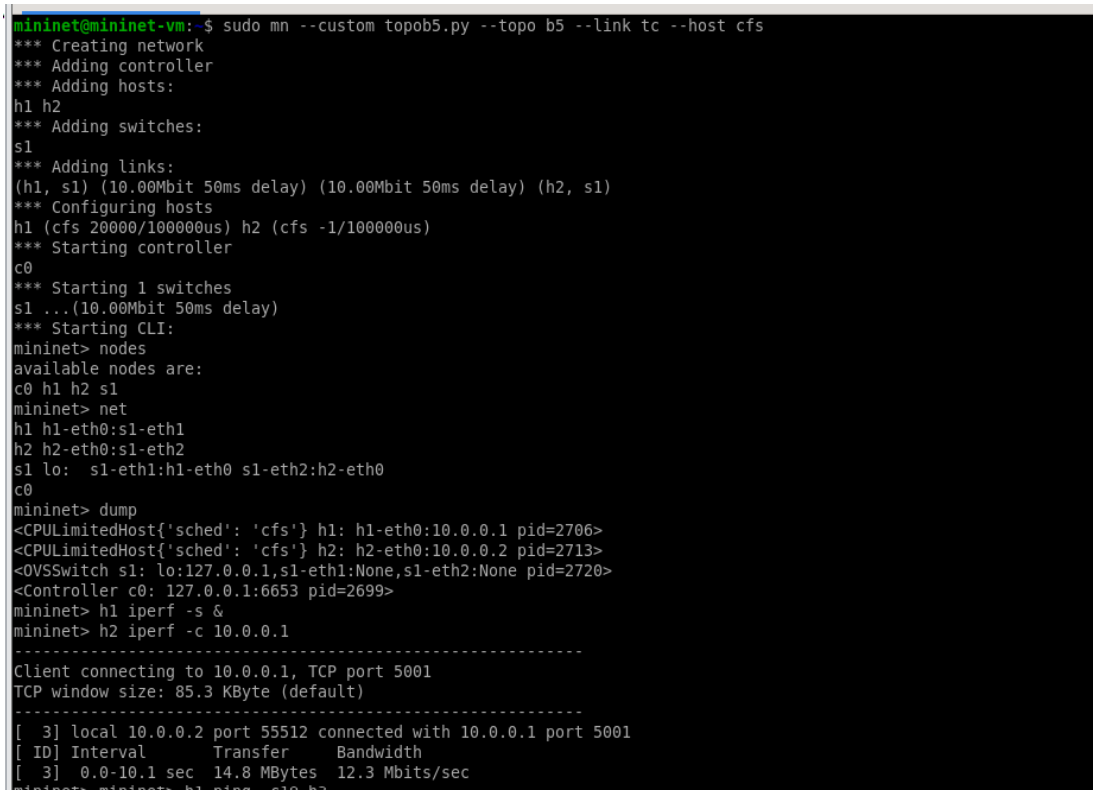


Figure 14: `iperf` result: bandwidth 10 Mbps

Bandwidth is limited to 10 Mbps, and the 50 ms link delay results in an RTT around 100 ms.

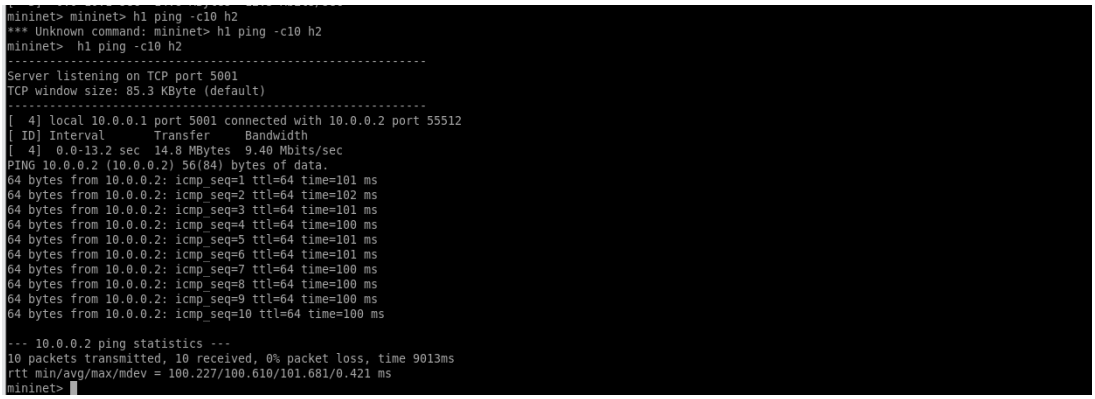


Figure 15: `ping` result: average RTT 100 ms

5.8 Topology Code B.5

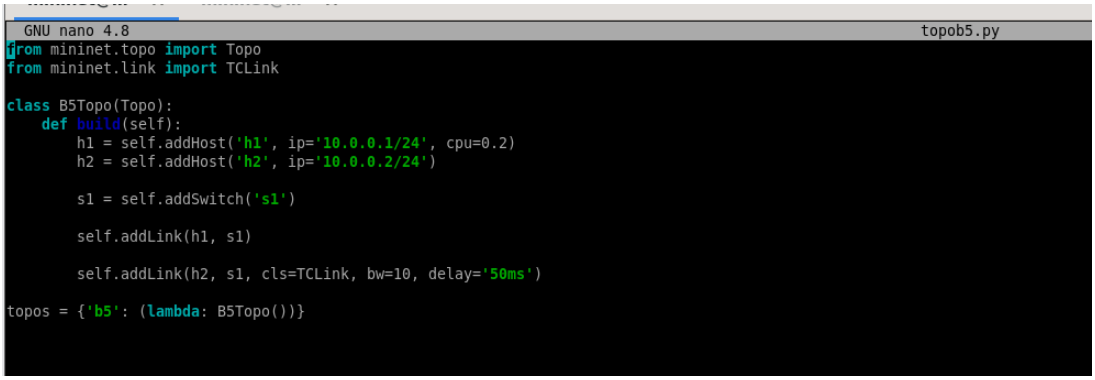


Figure 16: Python script `topob5.py`

5.9 B.6 – Verbosity Levels and Xterms

```
sudo mn -v debug   # detailed output
sudo mn -v output  # CLI output only
sudo mn -x         # open xterm for each host and switch
```

Before the ping, the flow table is empty. After communication starts, new entries appear.

5.10 B.7 – Python Interpreter

Commands starting with `py` are evaluated as Python code:

```
sudo mn
mininet> py 'hello ' + 'world'
mininet> py locals()
mininet> py dir(s1)
mininet> py help(h1)
mininet> py h1.IP()
```

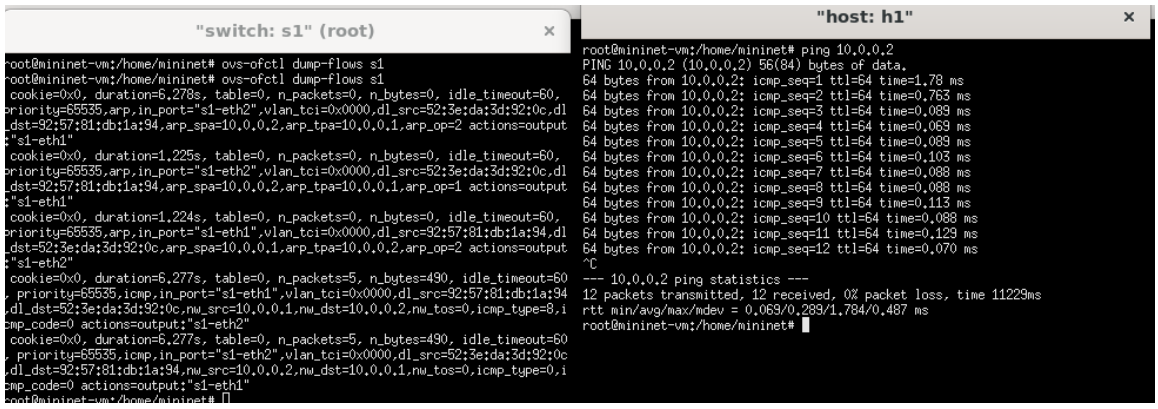


Figure 17: Flow table of s1 before and after pinging between h1 and h2

Figure 18: Examples of Python evaluation inside Mininet CLI

5.11 Reading Port Counters with `ovs-ofctl`

`ovs-ofctl dump-ports s1` shows activity on each switch port:

- **LOCAL**: internal management interface (drops here are normal).
- **s1-eth1, s1-eth2**: host interfaces. RX/TX counters increase when traffic flows.

Verification procedure:

```
mininet> net
mininet> h1 ping -c3 10.0.0.2
$ sudo ovs-ofctl dump-ports s1
```

Simulated failure:

```
mininet> link s1 h1 down
mininet> pingall
$ sudo ovs-ofctl dump-ports s1
mininet> link s1 h1 up
```

6 References

- Mininet releases: `https://github.com/mininet/mininet/releases`

- Official Mininet documentation: `http://mininet.org/`
- VMware Tools guide: `https://kb.vmware.com/s/article/1022525`
- Mininet custom topologies (GitHub): `https://github.com/mininet/mininet/tree/master/custom`
- Mininet API documentation: `http://mininet.org/api/annotated.html`