Name: Lakshmi Sai Anmol Ramayanam Username: LR2947

Date: 05/15/2022 Section: 001

**Assignment 8: Final Project**

Part 3 functionality [Max 70 points]

Part 4 functionality [Max 25 points]

Style [Max 5 points]

Part 3 extra credit questions [Max 20 points (10 each)]

Total [Max 100 points + Extra Credit points] Total in points:

Total in extra credit points:

Professor’s Comments:

Affirmation of my Independent Effort: Lakshmi Sai Anmol Ramayanam

Table of Contents

[Introduction 3](#_Toc91070569)

[Proposed Solution/Architecture/Algorithms 4](#_Toc91070570)

[Simulation/Implementation 7](#_Toc91070571)

[Results & Conclusion 9](#_Toc91070572)

[Related Work 9](#_Toc91070573)

# **Introduction**

SDN is a networking method that communicates with underlying hardware infrastructure and directs traffic on a network using software-based controllers or application programming interfaces (APIs).

Traditional networks, on the other hand, require specialised hardware devices to regulate network traffic (e.g., routers and switches). SDN may control traditional hardware or establish and operate virtual networks.

While network virtualization allows organizations to segment different virtual networks within a single physical network or connect devices from different physical networks to form a single virtual network, software-defined networking allows for a new way of controlling data packet routing through a centralized server.

# **Proposed Solution**

Floodlight is used to create the SDN controller, which ensures that Dijkstra's shortest path routing method is implemented. Mininet, which can create multiple topologies like assign1 and someloops, is used to evaluate the simulation. Whenever a link is brought up or down, this implementation searches for the quickest path.

We can create various topologies like described below with SDN switch (denoted by s) and hosts (denoted by h) directly connected to the switches.

* linear,n:

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

* tree, n:

A screenshot of a computer screen

Description automatically generated with low confidence

* assign1:

Graphical user interface, application

Description automatically generated

* triangle:

A screenshot of a computer screen

Description automatically generated with low confidence

* mesh, n:

A picture containing text, electronics

Description automatically generated

* someloops:

Graphical user interface, text, application

Description automatically generated

# Simulation/Implementation

Here's a quick rundown of what some of the functions in the ShortestPathSwitching.java file do:

1. deviceAdded():

This function assists in obtaining information about the new host and iterating through all switches. Every time, the shortest path to this host's switch is computed.

We then use the shortest path information to determine the best next hop switch for each switch.

Then, with the host's MAC address and the port through which the present switch is connected to the next-hop switch, we may add a rule.

1. deviceMoved():

Similar to the added function, we would update the route in each node for the moved device.

1. deviceRemoved():

We can remove the rule linked with the deleted host by iterating through each switch.

1. switchRemoved():

When a switch is removed, the complete topology must be updated. Then go through each host and repeat step 1 for each one.

1. linkDiscoveryUpdate():

When a link is altered, the entire topology is updated.

Features working:

* To implement these paths, a global shortest-path switching table was created, with forwarding rules loaded on the switches.
* Based on global topology data gathered by the controller, this table was built on the controller.
* Based on a global view of the network topology, the application now builds route tables.
* Each SDN switch has a route table that directs packets.
* Created entries that match packets based on their destination IP address (and Ethernet type), then perform an output action to send the packet out a selected SDN port.
* The output action is the same as the interface field in a standard route table, and the match criteria are the same as the destination and mask fields.

LoadBalancer:

• A distributed load balancer is represented by the LoadBalancerInstance class.

• Each load balancer instance has a virtual IP address, virtual MAC address, and a list of hosts where TCP connections should be distributed.

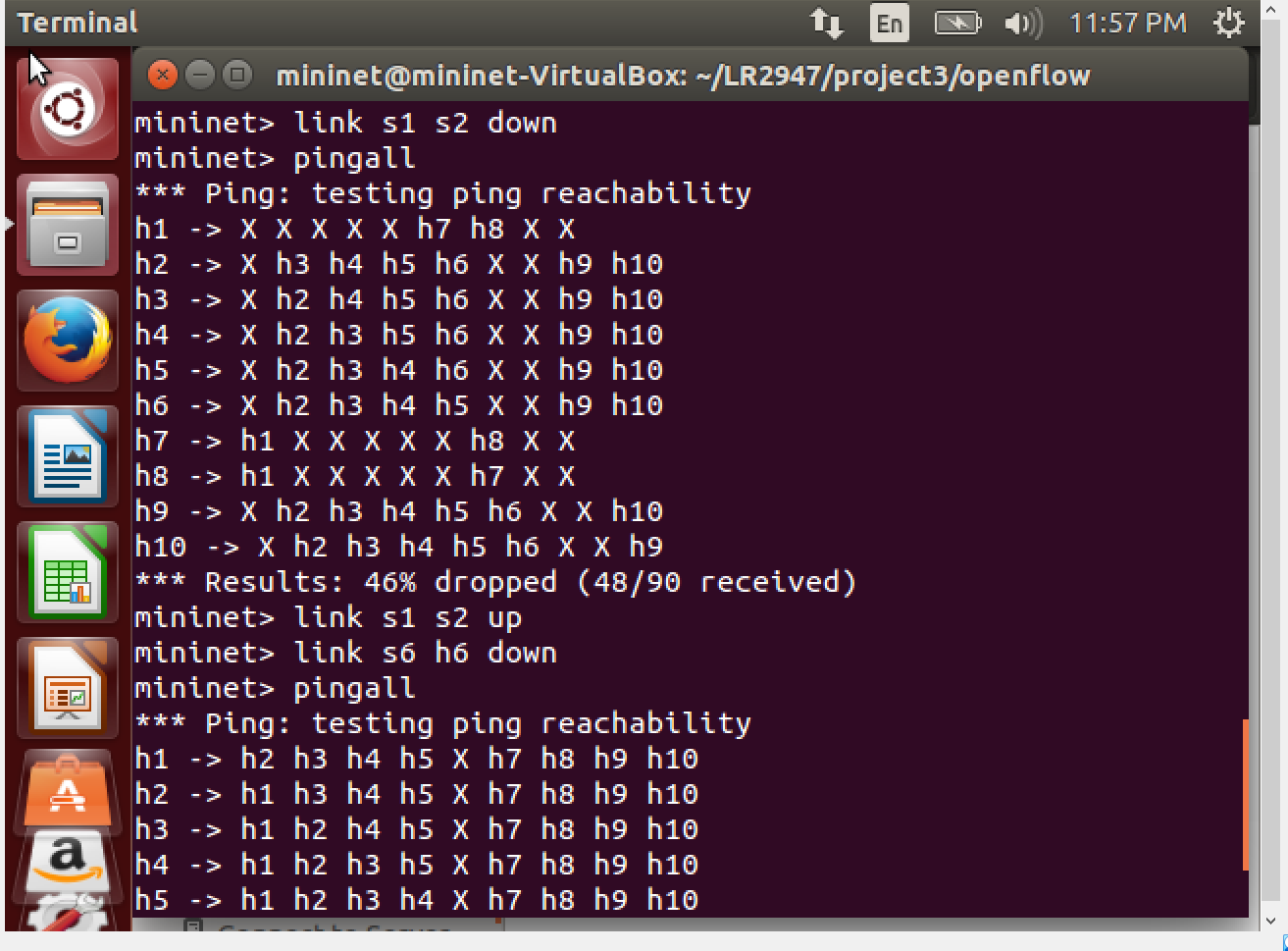
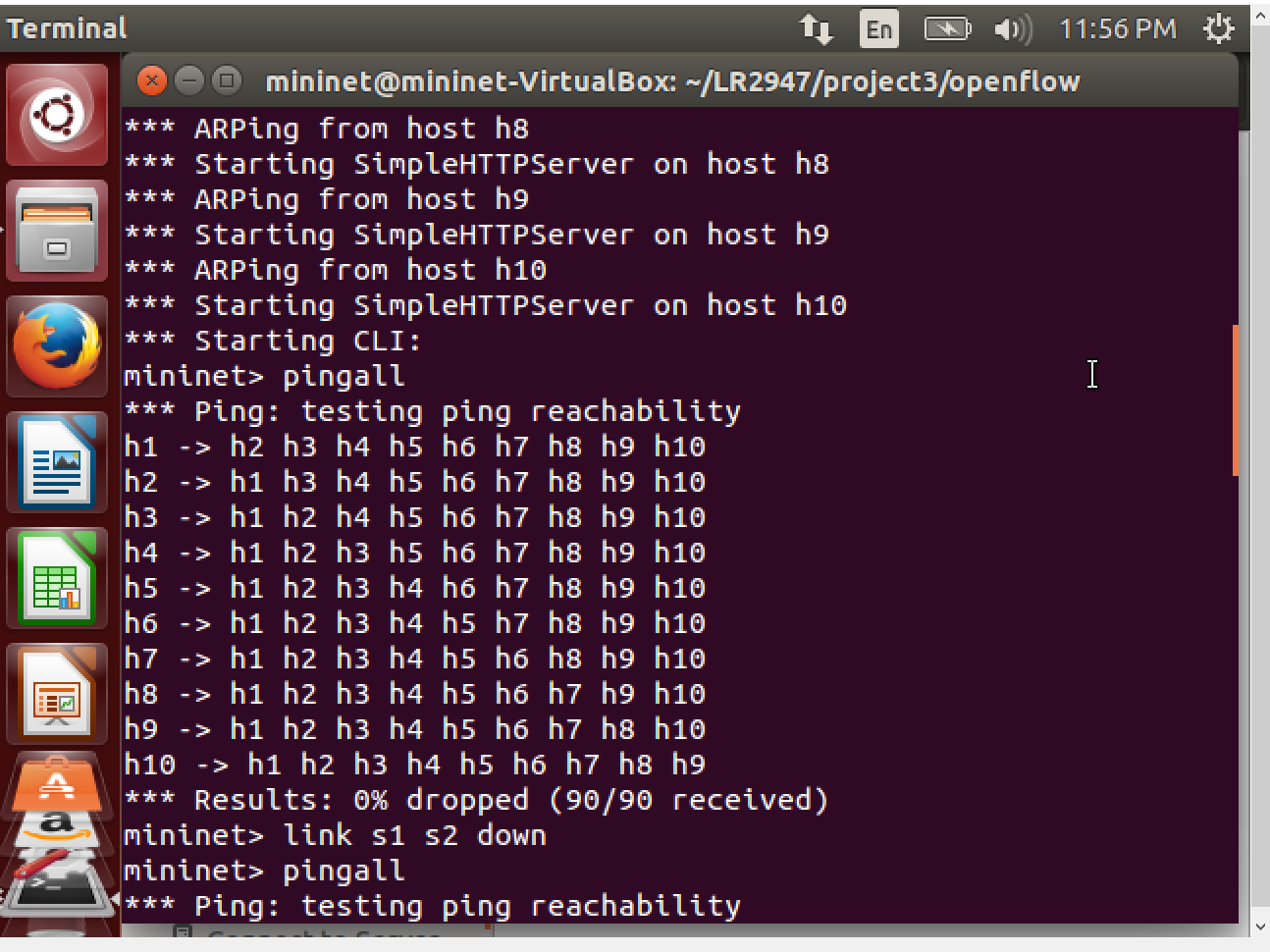
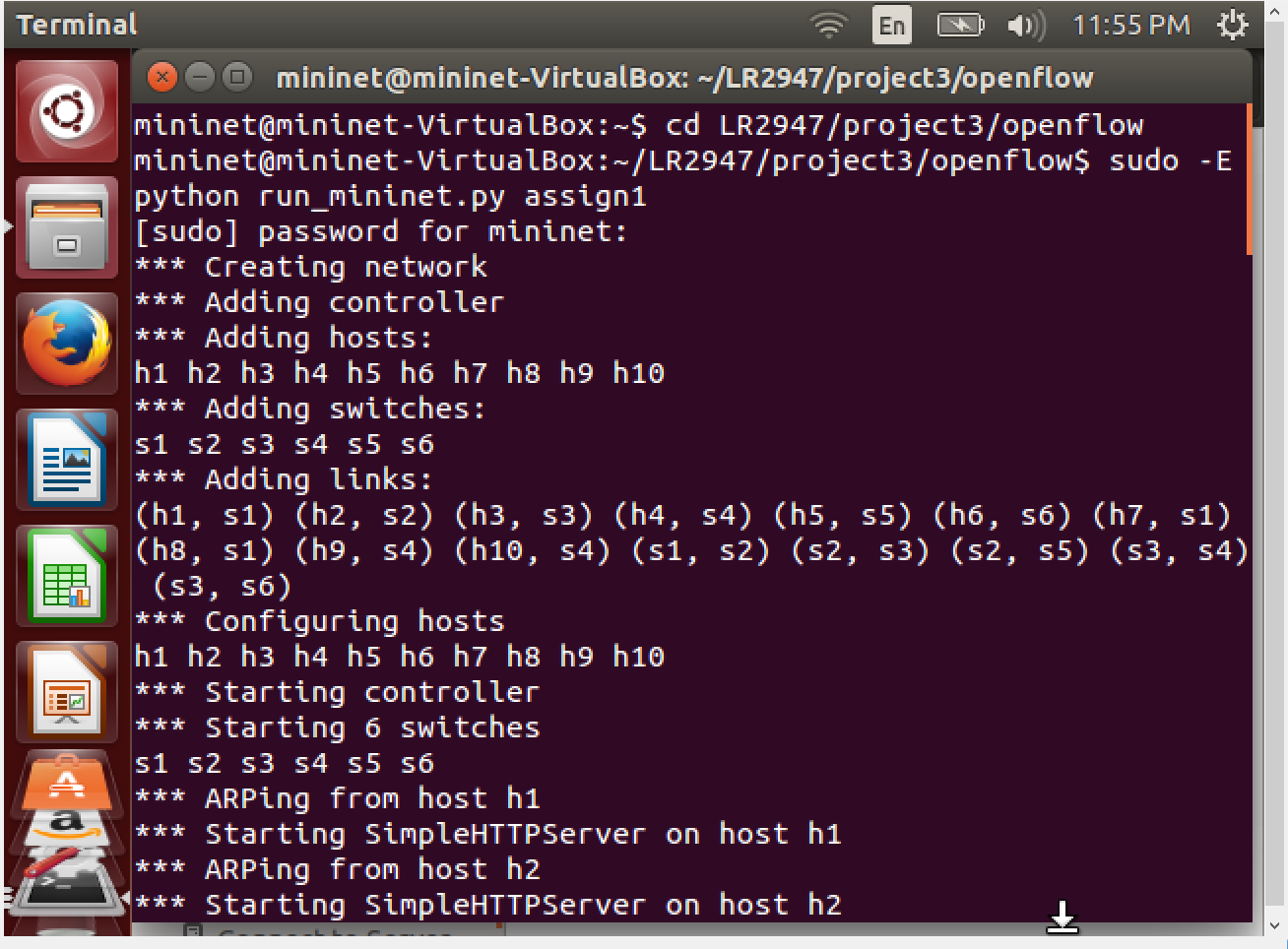
• The LoadBalancer class's instances class variable binds a virtual IP address to a specific load balancer instance.

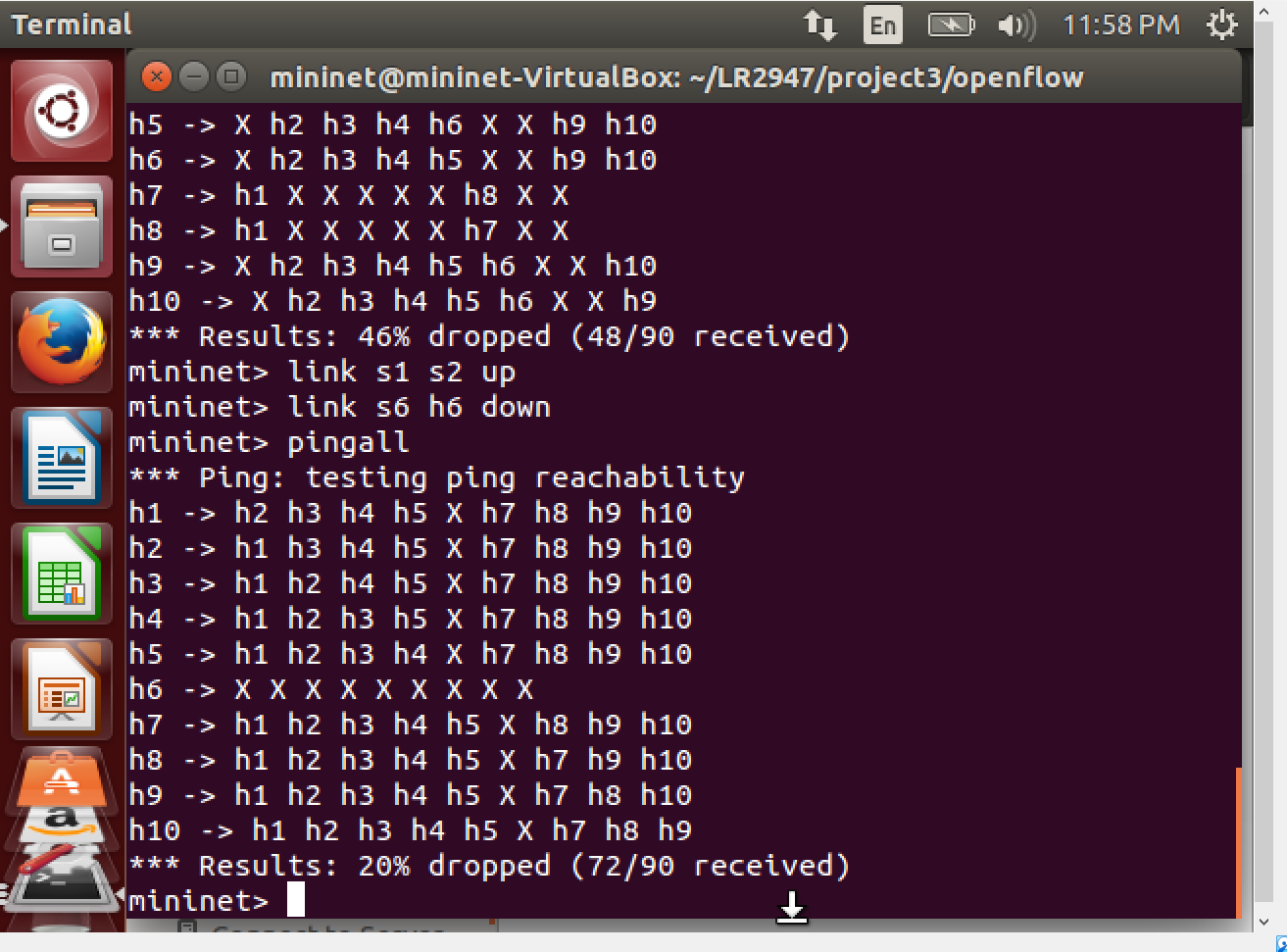
Code :

<https://github.com/ARamayanam097/SDN>

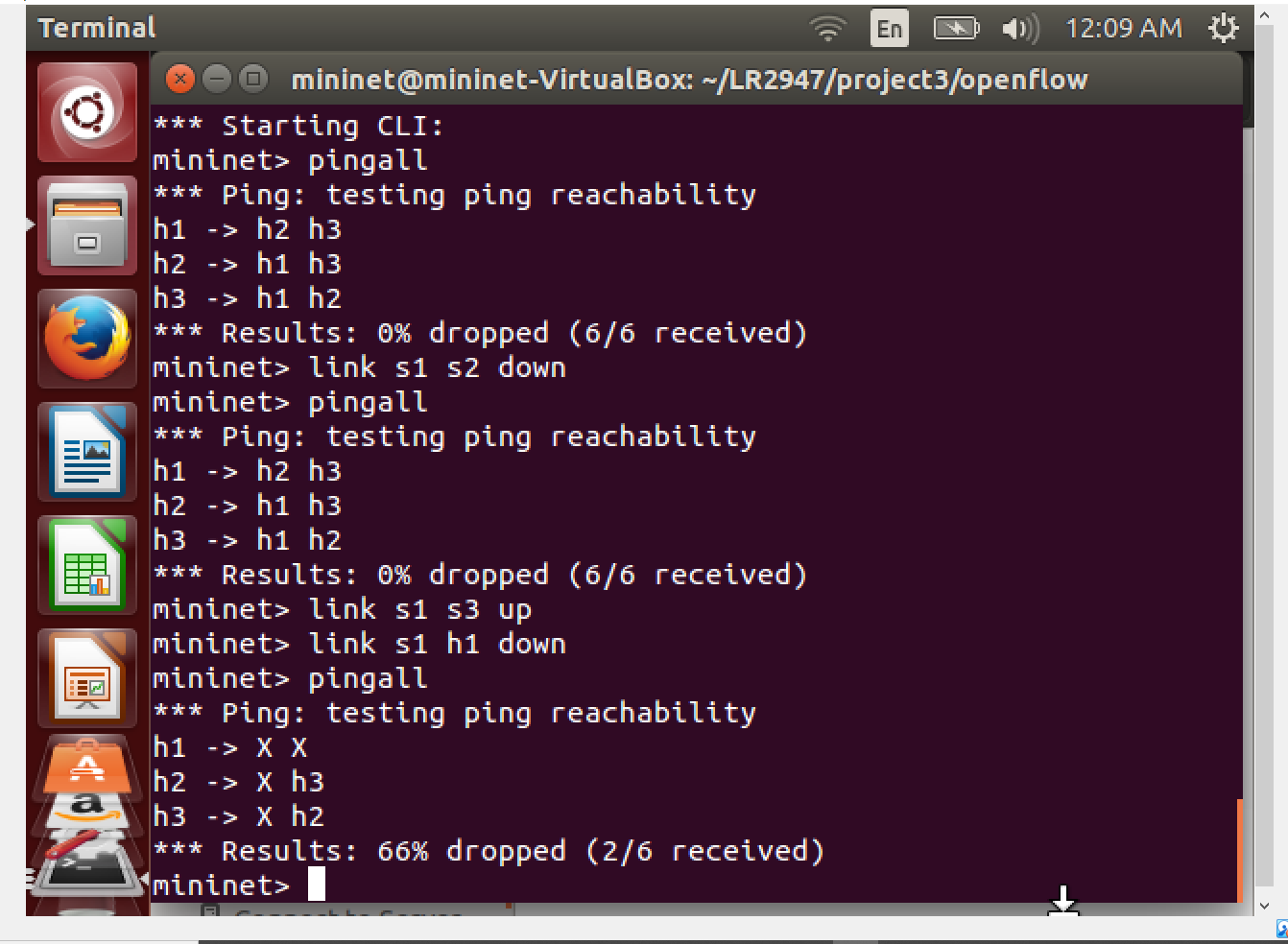
# Results & Conclusion

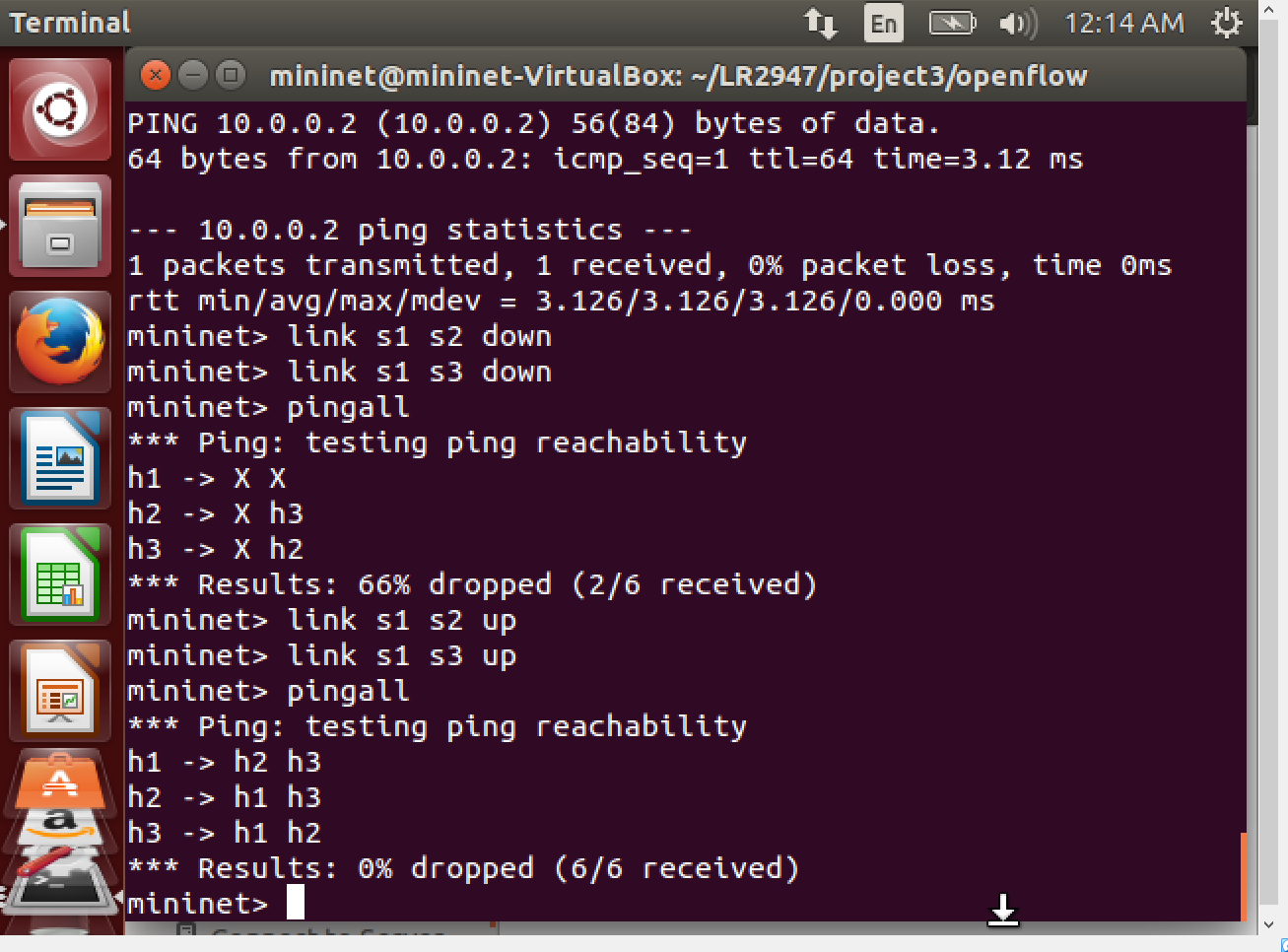
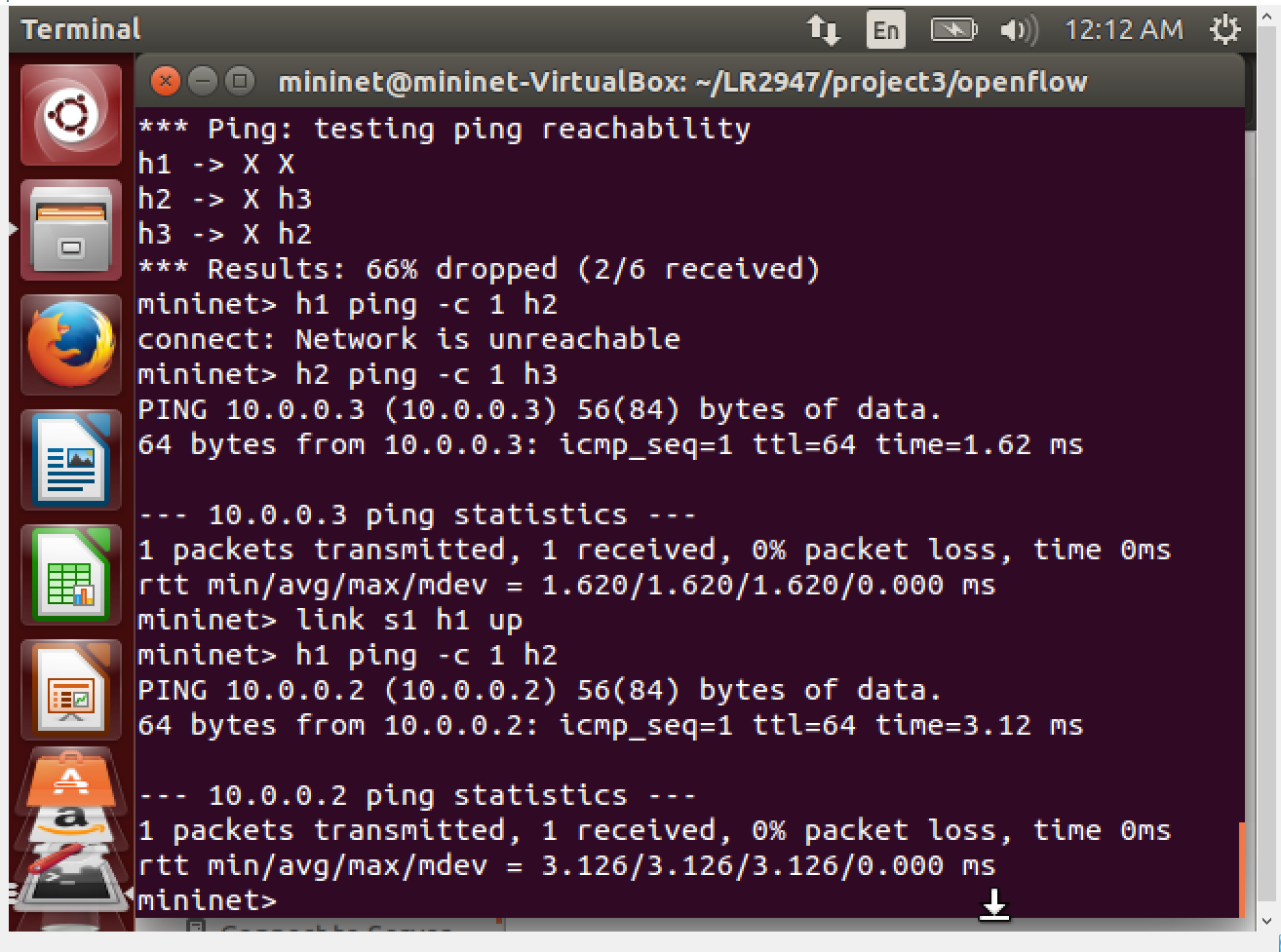
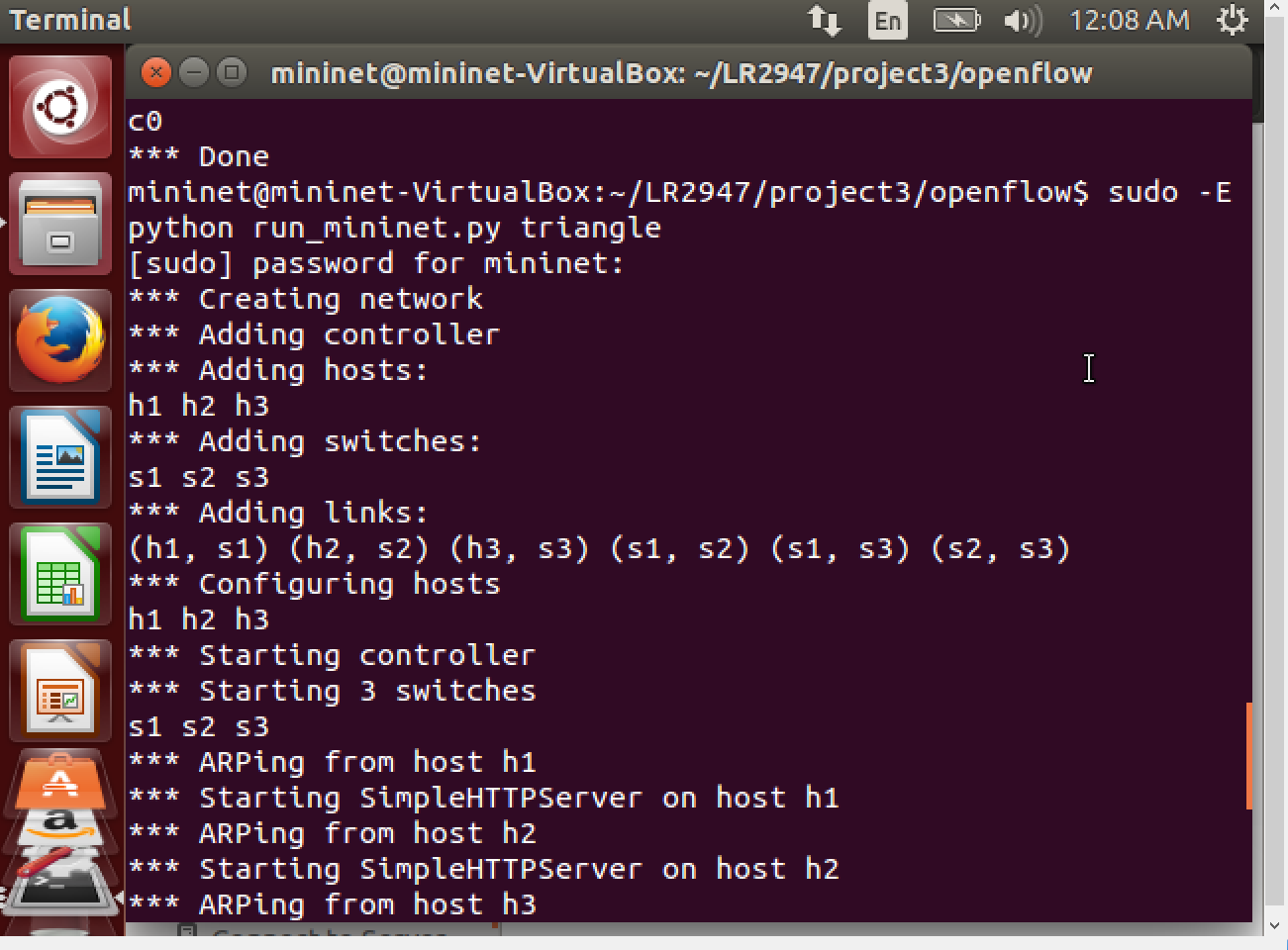
By implementing Dijkstra's shortest path algorithm, we can effectively conclude that our SDN implementation using mininet simulations for multiple topologies works as predicted.

Assign1:



Triangle Topology:





# Related Work

<http://mininet.org>

<https://www.opennetworking.org/sdn-resources/onf-specifications/openflow>

<http://openvswitch.org>

<https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview>

<https://www.vmware.com/topics/glossary/content/software-defined-networking.html>