Software-Defined Networking

Lab 5

Midterm Lab

University of Colorado Boulder

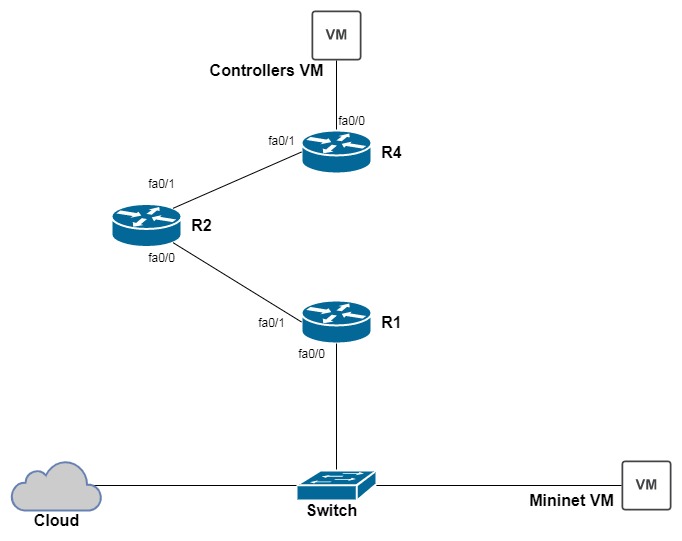
Department of Computer Science

Professor Levi Perigo, Ph.D.

# Summary

The objective of this lab is to recall and apply all the knowledge you have gained so far in this course. You will utilize the knowledge of traditional networking, software-defined networking, virtual switches, controllers, OpenFlow, packet capturing, GNS3 and Python. Students are encouraged to expand on the topics for additional learning and experiments.

# Objective 1 – Set up topology in GNS3



|  |  |  |
| --- | --- | --- |
| **Device** | **Interface** | **IP address** |
| **R1** | fa0/0 | 192.168.100.1/24 |
|  | fa1/0 | 192.168.200.1/24 |
| **R2** | fa0/0 | 192.168.200.2/24 |
|  | fa1/0 | 172.16.100.2/24 |
| **R4** | fa0/0 | 10.20.30.1/24 |
|  | fa1/0 | 172.16.100.1/24 |
| **Controllers VM** |  | 10.20.30.2/24 |

1. Configure the above topology in GNS3 interconnecting traditional routers, virtual machines, cloud and host laptop.
2. Configure IP addresses on the routers and the Controllers VM as given. Do not configure any IP address on the Mininet VM.
3. Configure DHCP server on R1 to provide an IP address to the Mininet VM.
4. Do not configure any routing commands manually.
5. Paste screenshot of the topology created in GNS3. [**15 points**]

A diagram of a network

Description automatically generated

1. Start the Ryu app simple\_switch\_13.py on the controllers VM.

# Objective 2 – Python script

1. Write a script in Python which runs on your laptop to achieve the following objectives (Please read all the objectives before you begin to write your script)-
   1. SSH into R1 and find the IP address leased out to the Mininet VM. Paste relevant screenshots. [**20 points**]

A computer screen shot of a program code

Description automatically generated

A computer screen shot of a computer code

Description automatically generated

* 1. SSH into the Mininet VM using the IP address found in the previous step and initialize the default Mininet topology (sudo mn). Paste relevant screenshots. [**20 points**]

A screenshot of a computer

Description automatically generated

A computer screen shot of a program code

Description automatically generated

* 1. Configure the OvS on the Mininet VM to connect to the controller. Paste relevant screenshots. [**20 points**]

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

* 1. SSH into the traditional routers R1, R2 and R3 to configure routing to establish the OpenFlow connectivity between the OvS and the controller. Paste relevant screenshots. [**20 points**]

A computer screen shot of a computer program

Description automatically generated

A computer code on a purple background

Description automatically generated

* 1. Verify and display that the successful OpenFlow connectivity between the OvS and the controller. Paste relevant screenshots. [**10 points**]

In objective C, I issued a pingall command to the mininet topology, here is the result of the ping being successful and the packet in messages displaying in the Ryu Controller

A screenshot of a computer

Description automatically generated

Now we can see the script I made to show the existence of the SDN working with ovs commands on the mininet VM:

A computer screen shot of a computer code

Description automatically generated

A computer screen with white and purple text

Description automatically generated

* 1. Capture the number of OpenFlow Packet\_In messages sent from the switch to the controller and visualize this through an interactive graph on a webpage. You can use your favorite Python web-framework like Flask, Django, etc. to set up the webpage. The graphs should be displayed in real-time i.e. they should get refreshed automatically after a periodic interval of time (say 5 seconds) without manually reloading the webpage. Paste relevant screenshots. [**30 points**]

Here is my method:

I have a flask webpage running with Flask-SocketIO that handles live updates without reloading the webpage. It is a simple flask webpage periodically reads a data.txt file every second. The chart was created through Chart.js. Here is the screenshot of the webpage running:

A screenshot of a computer

Description automatically generated

A computer screen shot of a program code

Description automatically generated

Also, I have a live.py file that does three things. First captures a pcap by SSHing into the Controller and starting a capture. It then stops and does a SCP to get the file to the NMS. The NMS then parses through to look for Packet\_IN packets and writes it to the data.txt file. This instance takes about 5 seconds and then the website looks for updates every second to get the data pushed visually.

A screenshot of a computer program

Description automatically generated

A computer screen shot of a program

Description automatically generated

A computer screen shot of a program code

Description automatically generated

So to make this all work, simply have two terminals open. Run live.py for getting data and app.py for the website.

* 1. At the end, push your Python script to a new private repo ‘SDN-Midterm' in your GitHub account. Paste relevant screenshots. [**5 points**]

A screenshot of a computer

Description automatically generated

Total Score = \_\_\_\_\_\_\_\_ / 140