Network Management

and Automation

Lab 5

Midterm Lab

University of Colorado Boulder

Network Engineering Program

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 Wireshark, TCPDUMP, GitHub, SNMP, DHCP, IPv6 and Python to complete the objectives of this lab. Students are encouraged to expand on the topics for additional learning and experiments.

## Prerequisite

If you do not already have the Cisco 7200 router images added in your course VM GNS3, download the Cisco 7200 image from - <https://drive.google.com/file/d/16iLceRNNAIhmsr9-UZnBUsVC0Ui9FtzX/view?usp=sharing>, and add it to GNS3 on the course VM using this link - <https://protechgurus.com/how-to-add-router-ios-image-in-gns3/>.

# Objective 1: Python Modules

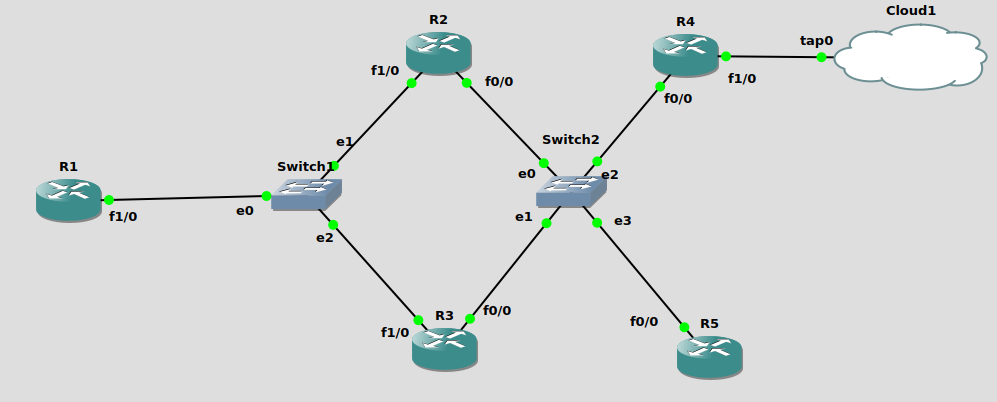
For this lab you will have to create modules (.py files) for each of the four Python files (NMtcpdump.py, NMdhcp.py, NMsnmp.py and NMgithub.py) you write in the next three objectives, which can be imported into a **NMmain.py** file. So, your NMmain.py file should start with –

import NMtcpdump, NMdhcp, NMsnmp, NMgithub

Submit the scripts. [Keep in mind that this code will be checked during your grading interview with the GSA’s.] [**10 points**]

# Objective 2: DHCPv4/v6, SLAAC, TCPDUMP

1. Import the topology topo\_midterm\_netman.gns3project in GNS3 of netman VM.



1. Paste the screenshot of imported GNS3 topology. [**1 point**]

A diagram of a network

Description automatically generated

1. Manually configure an IPv4 address on R4-F1/0. Provide relevant screenshot. [**2 points**]

A computer screen with white text

Description automatically generated

1. Manually enable IPv6 on R2-F0/0, R3-F0/0, and R5-F0/0 and have them obtain IPv6 addresses using SLAAC by manually configuring R4 to act as the local IPv6 router. Paste relevant screenshots. [**15 points**]

R4 Configuration:

A computer screen with white text

Description automatically generated

A screen shot of a computer

Description automatically generated

R2:

A computer code on a purple background

Description automatically generated

A screenshot of a computer

Description automatically generated

R3:

A computer screen shot of a number

Description automatically generated

A screenshot of a computer

Description automatically generated

R5:

A computer screen with white text

Description automatically generated

A screenshot of a computer screen

Description automatically generated

1. Manually enable R2-F0/0, R3-F0/0 and R4-F0/0 to obtain IPv4 addresses from DHCPv4 server. Provide relevant screenshot. [**2 points**]

R2:

A computer screen shot of white text

Description automatically generated

R3:

A computer screen shot of white text

Description automatically generated

R4:

A computer screen shot of white text

Description automatically generated

1. Run tcpdump on C1 on the tap interface while saving the packet captures to a .pcap file.
   1. Manually ping C1 from R2 and R3. Stop the captures once the pings are successful.
   2. Paste the screenshot of your saved .pcap file indicating the successful ping communication. [**10 points**]

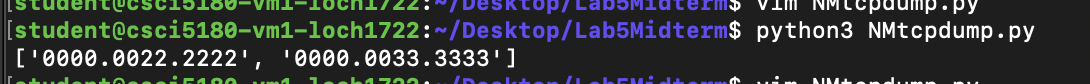
A screenshot of a computer screen

Description automatically generated

As you can see both R2 and R3 can reach C1 which has a ip of 4444::2. Associated EUI-64 addresses do represent MAC address on respective routers.

1. Write a Python script **(NMtcpdump.py)** which parses the saved .pcap file and extracts the MAC addresses of R2-F0/0 and R3-F0/0.

Paste relevant screenshots and submit your script. [**30 points**]

getIps() function grabs the Ipv6 Ips from the pcap ICMPv6 packets. EUI64ToMac(Ips) function takes those Ips and converts the EUI64 addresses to Mac addresses. 

1. Write a Python script **(NMdhcpserver.py)** to configureIPv4 Host DHCP pools on R5 for R2-F0/0, R3-F0/0, and R4-F0/0 (Remember, there is no IPv4 address on R4-F0/0) **[30 points]**

Your script does the following:

* SSH to R4, figures out the IPv6 address of the R5-F0/0 and SSH to R5.
* Using the MAC addresses of R2-F0/0 and R3-F0/0 extracted in the previous step R5 assigns host IPv4 addresses to R2-F0/0 and R3-F0/0 via DHCP, while assigning a dynamic IPv4 address to R4-F0/0 via DHCP.
* Returns a list of IP addresses of all the DHCPv4 clients. Paste the relevant DHCP screenshots from R4 (This can be done manually) **[5 points]**

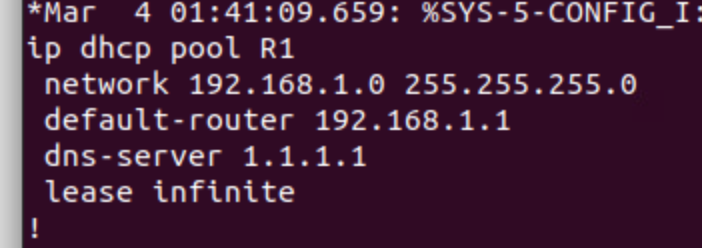
A screenshot of a computer

Description automatically generated

1. Manually configure R1 to act as both DHCPv4 as well as DHCPv6 (stateful) servers to assign v4 and v6 addresses to R2-F1/0 and R3-F1/0. [**10 points**]

These are configured using slaac, as per the comment on slack from Kiran informing us we could do SLAAC since the ‘ipv6 address dhcp’ command is not available on C7200 routers.

R1:



A computer screen with white text

Description automatically generated

R2:

A computer screen with white text

Description automatically generated

R3:

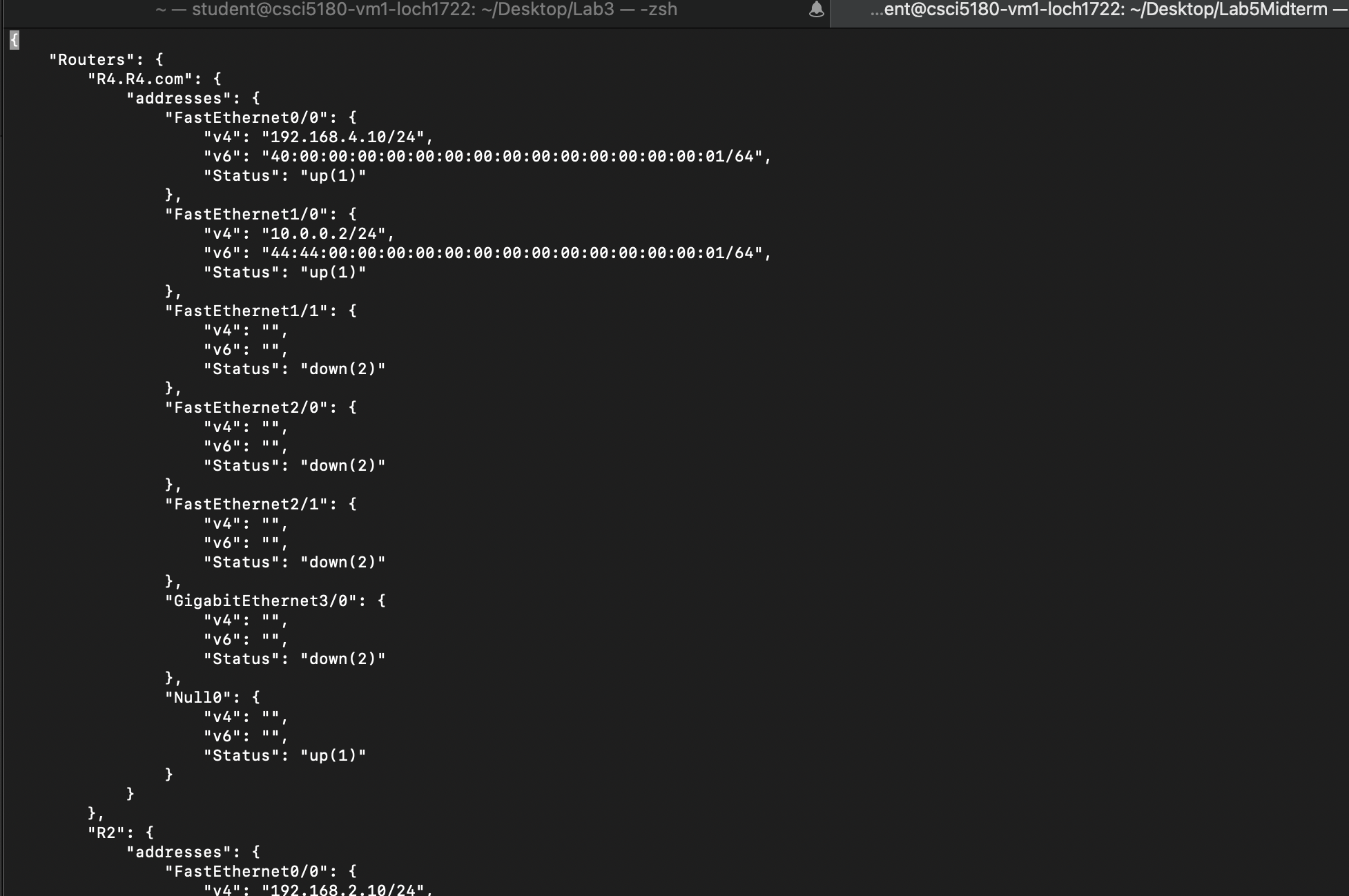
A computer screen shot of white text

Description automatically generated

# Objective 3: SNMP

Write a Python script **(NMsnmp.py)** using libraries for SNMP (like Easy SNMP or PySNMP) which-

1. Fetches both IPv4 and v6 addresses, and the interface status for all interfaces from the 5 routers (R1 to R5) using SNMP OIDs.
2. Stores the addresses (v4 and v6) in JSON format in a .txt file.

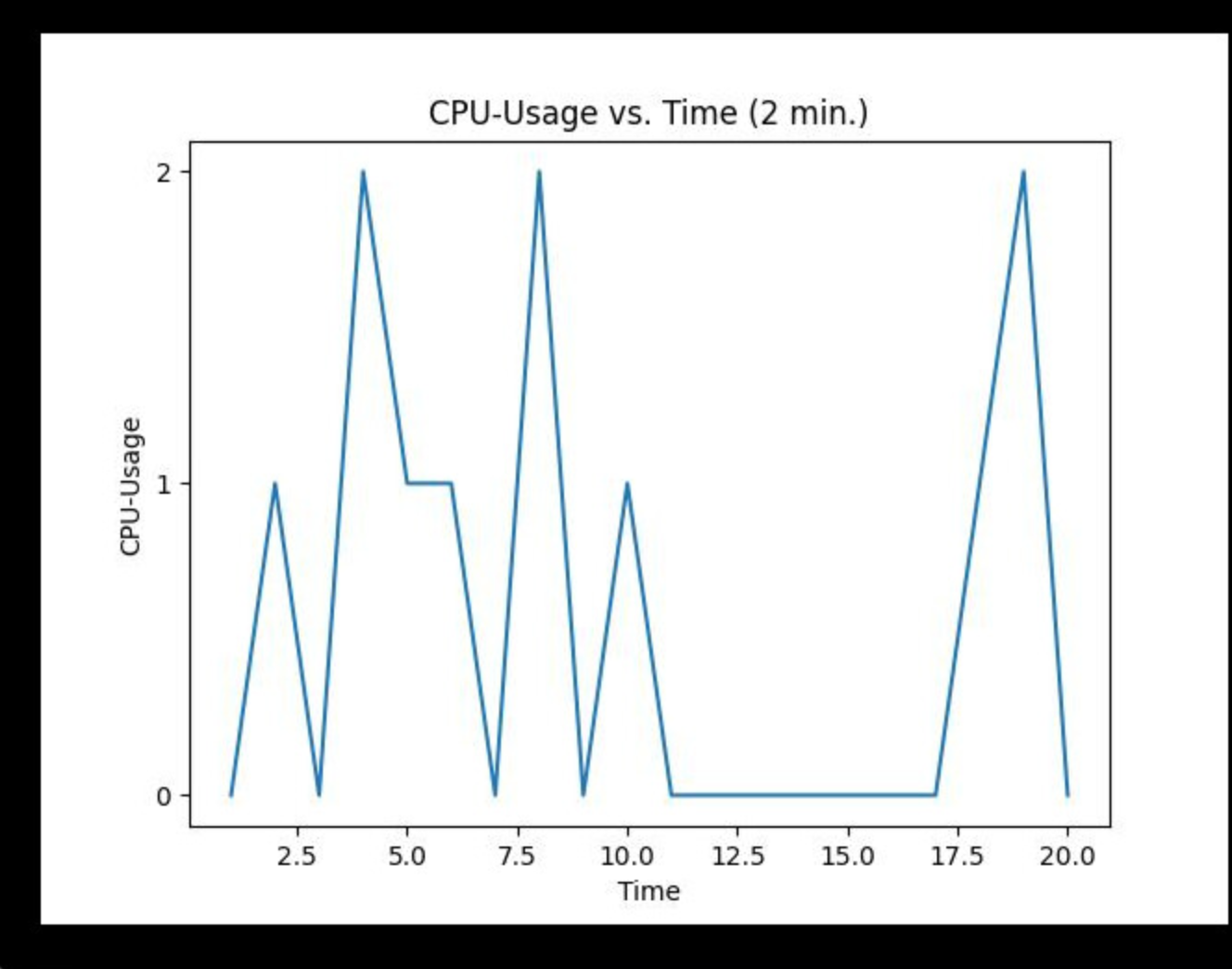
The format should be something like this:



1. Store the interface status (up/down) as a dictionary inside of another dictionary in the same .txt file.

The format should be something like this-



1. Fetches the CPU utilization of R1 continuously for 2 minutes in intervals of 5 seconds, plots this data as a line graph and saves the figure as .jpg file.

Paste relevant screenshots and submit the script. [**70 points**]

# Objective 4: GitHub

Write a Python script **(NMgithub.py)** which uses [GitPython](https://gitpython.readthedocs.io/en/stable/) –

1. Uses the previously created GitHub repository from lab 2 (NM\_lab) or creates a new GitHub repository.
2. Pushes the .txt and .jpg files from the previous objective to GitHub.
3. Pushes latest modified files to GitHub:
4. Compares modified files in the local repository against the GitHub repository.
5. Pushes those modified files to the GitHub repository.

Past relevant screenshots and submit the script. [**45 points**]

A screenshot of a computer

Description automatically generated

# Total Score = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/240