TLEN 5410 – Network Management and Automation

Lab 4

Midterm Lab

University of Colorado Boulder

Interdisciplinary Telecom 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, AWS, 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

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

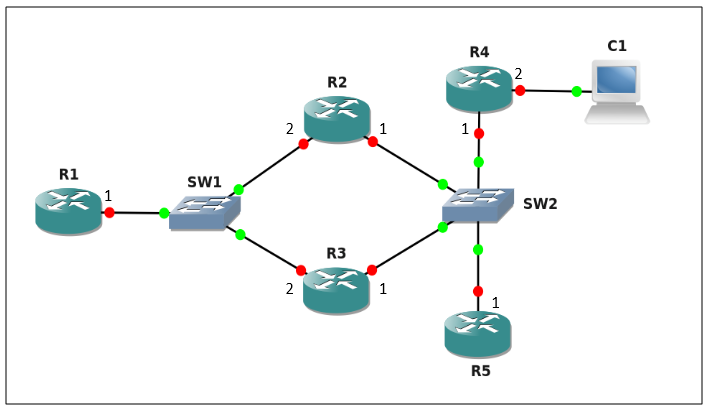
For this lab you will have to create modules (.py files) for each of the three Python files (NMtcpdump.py, NMsnmp.py and NMaws.py) you write in the next three objectives, which can be imported into a **NMmain.py** file. So, your NMmain.py file should start with –

import NMtcpdump, NMsnmp, NMaws

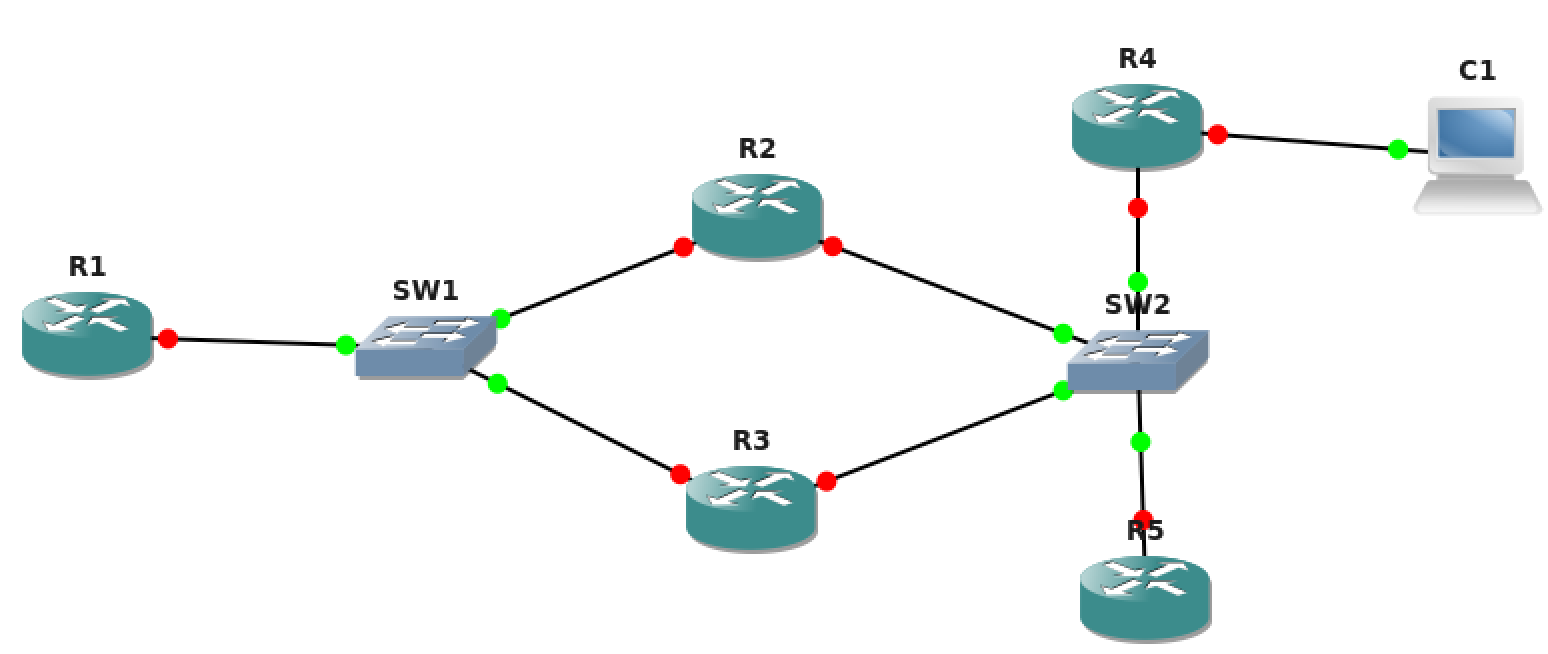
Submit the scripts. [Keep in mind that this code will be checked during your grading interview with the SA’s.] [**20 points**]

# Objective 2: DHCPv4/v6, SLAAC, TCPDUMP

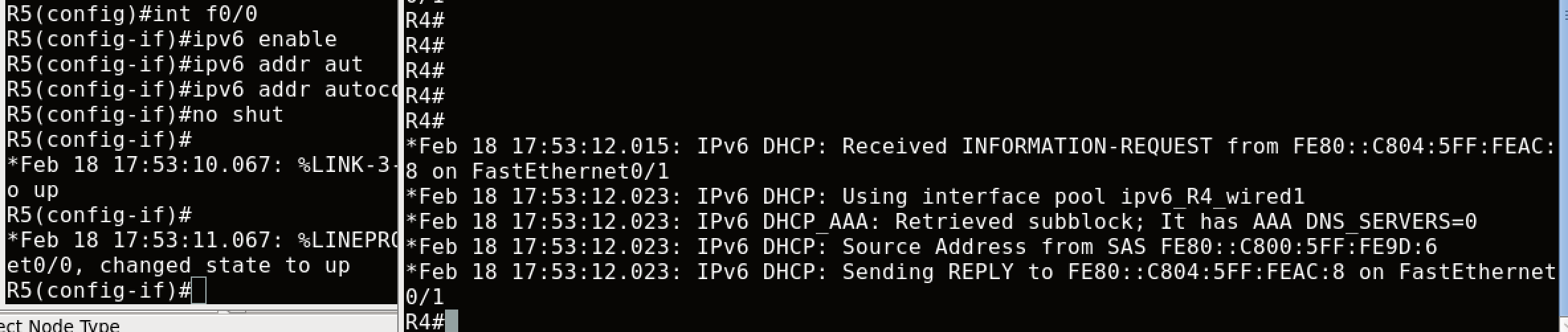
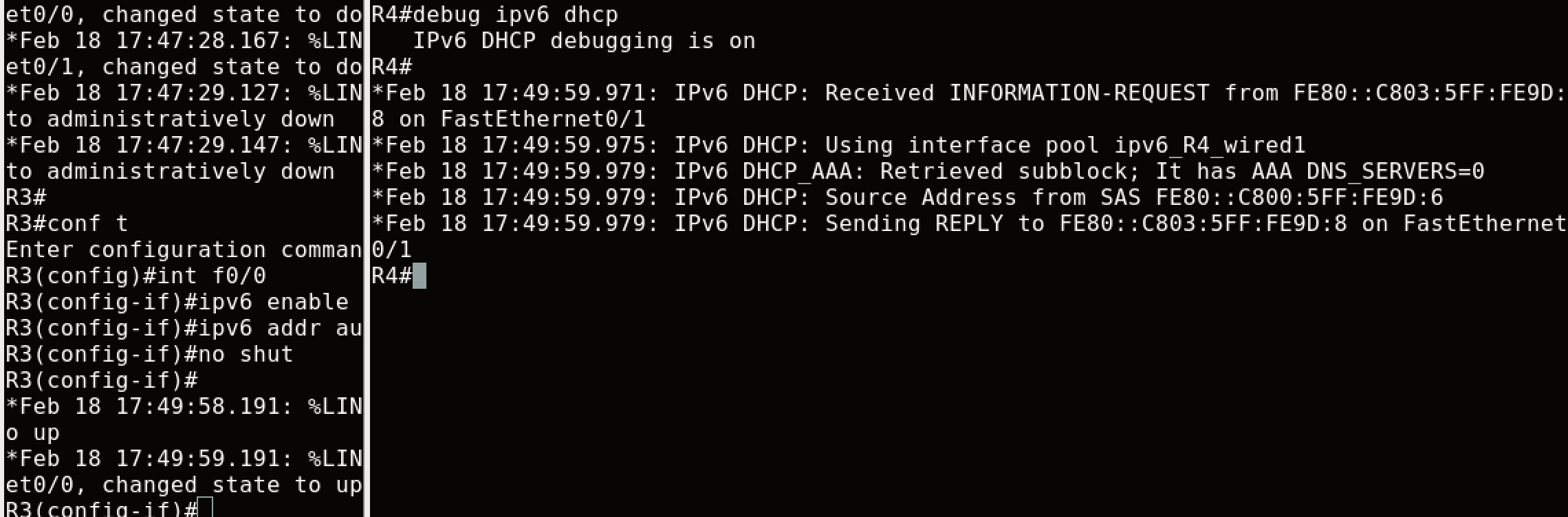
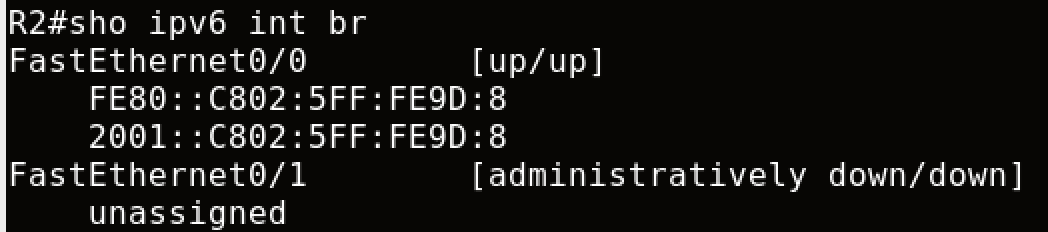
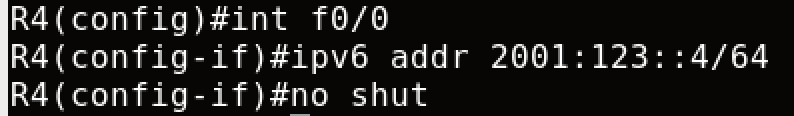
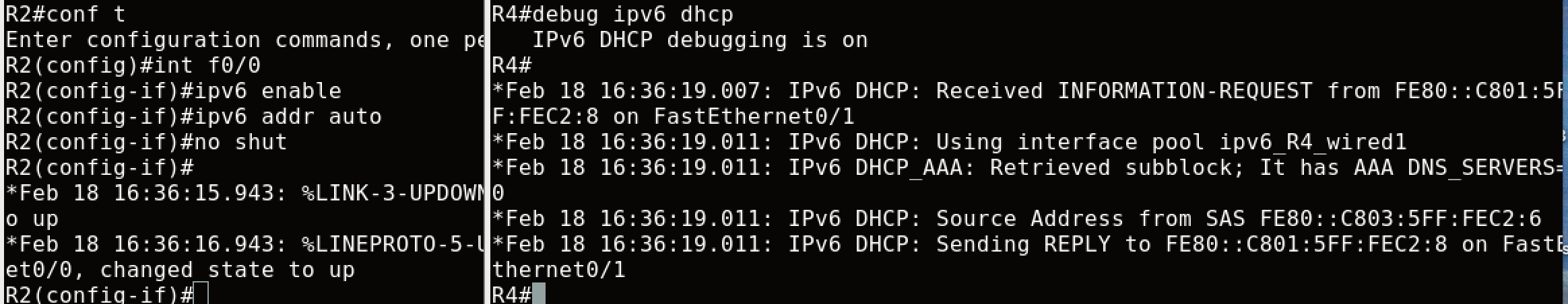
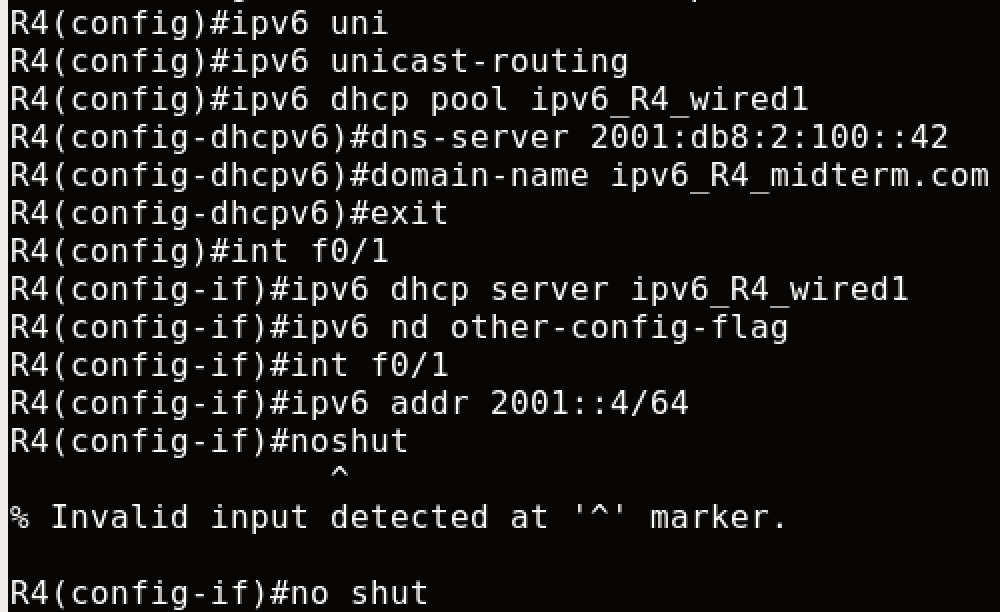
1. Create the topology in GNS3 as shown below. C1 is your host VM. Use the Cisco 7200 image for the routers.



1. Paste the screenshot of your GNS3 topology. [**5 points]**

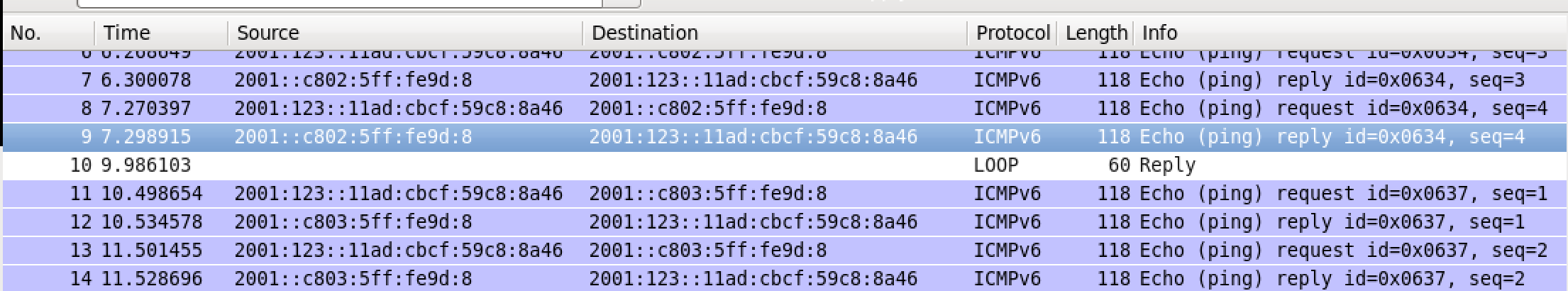


1. For the purpose of this lab, R1-1 refers to Router 1’s interface numbered 1, and similarily, R4-2 refers to Router 4’s interface numbered 2.
2. Enable IPv6 on R2-1, R3-1 and R5-1 and obtain IPv6 addresses using SLAAC with R4 acting as the local router. Paste relevant screenshots. [**15 points**]



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



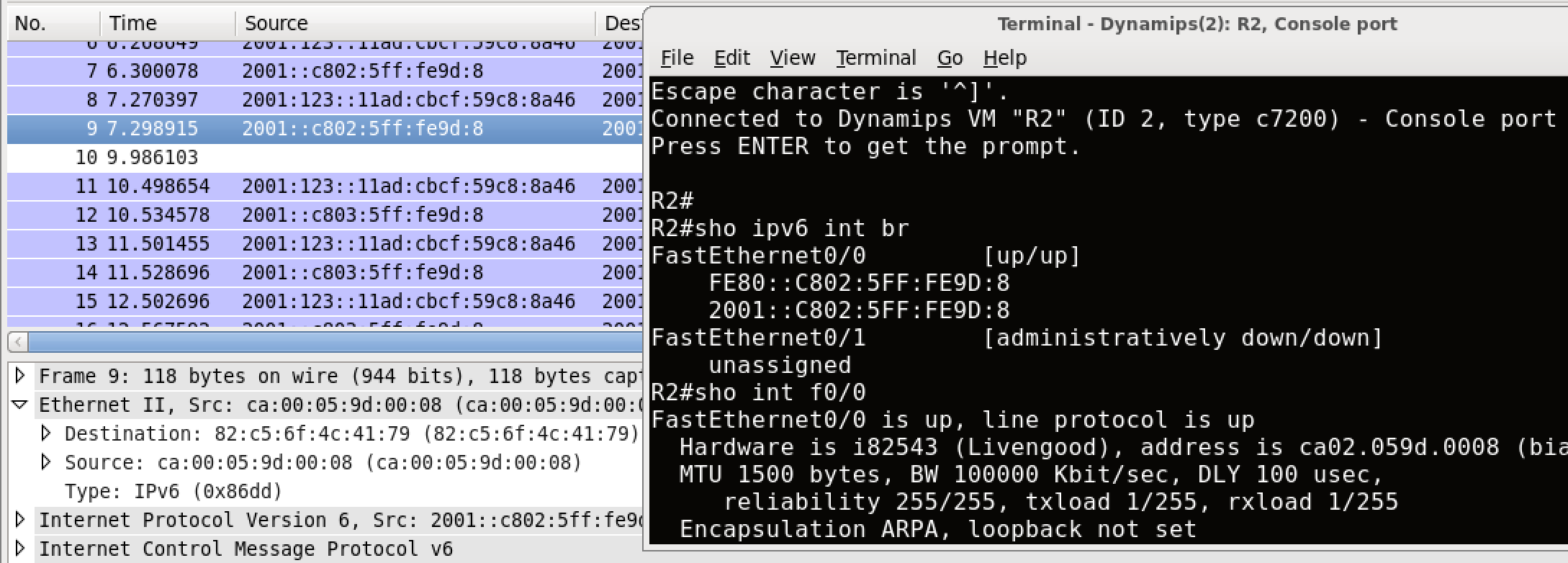


1. Write a Python script **NMtcpdump.py** which parses the saved .pcap file and extracts the MAC addresses of R2-1 and R3-1.

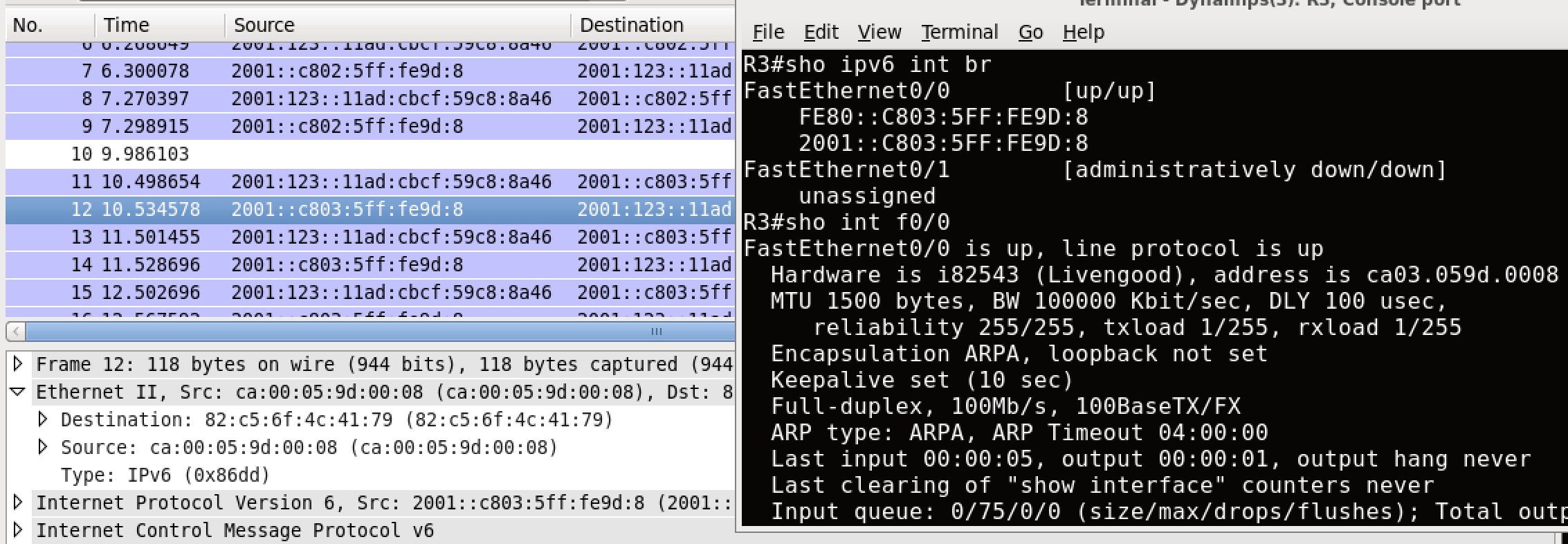
Paste relevant screenshots and submit your script. [**30 points**]

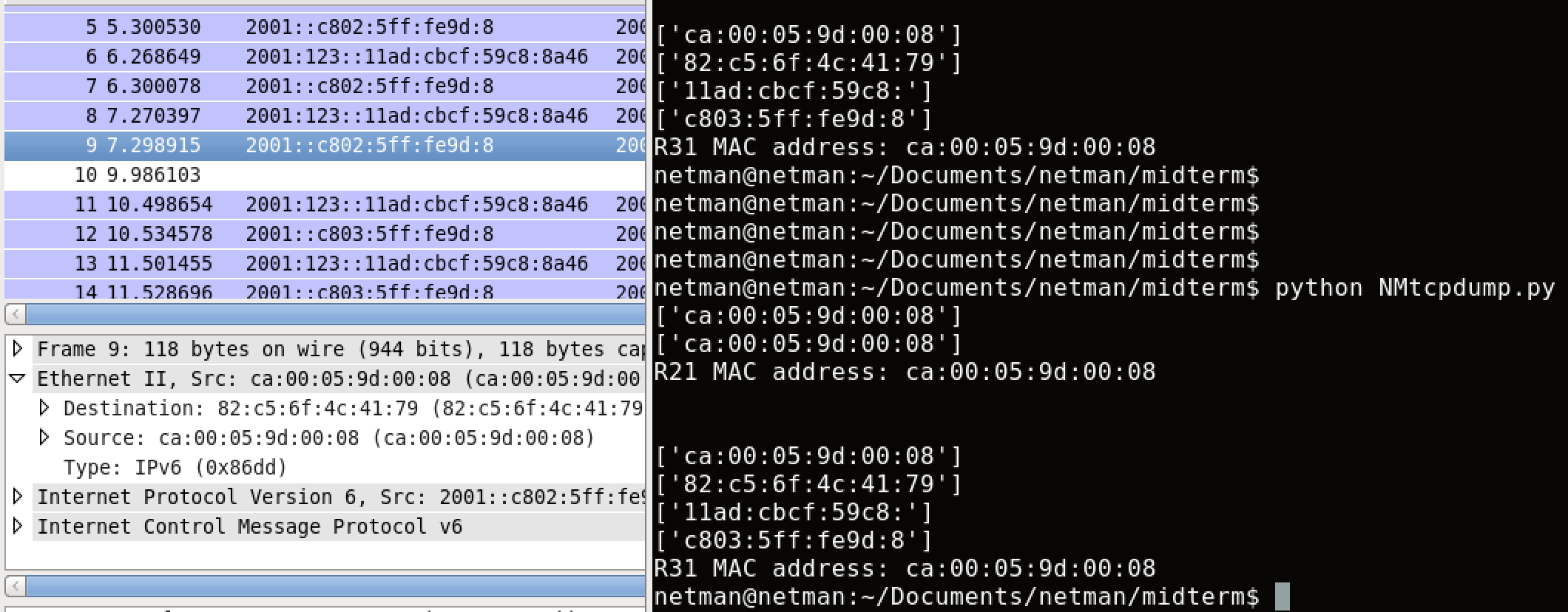
Note: I think there’s a bug, after I check the .pcap file, the MAC address are all the same, but it is different compared with the `show int <interface>` in router.

R21:



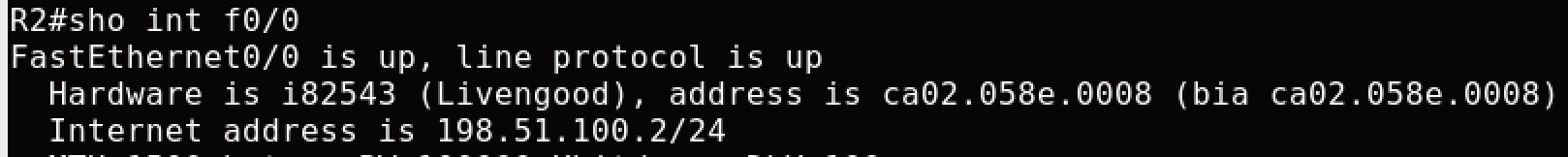
R31:



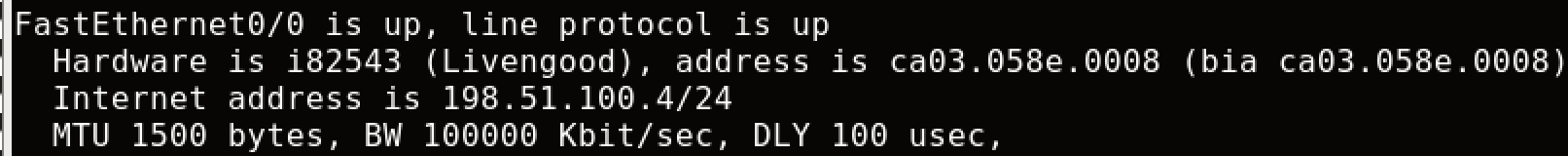


1. Using the MAC addresses of R2-1 and R3-1 extracted in the previous step, manually configure IPv4 Host DHCP pools on R5 to assign static addresses to R2-1 and R3-1, while assigning a dynamic IPv4 address to R4-1. Paste the relevant DHCP screenshots from R4. [**10 points**]

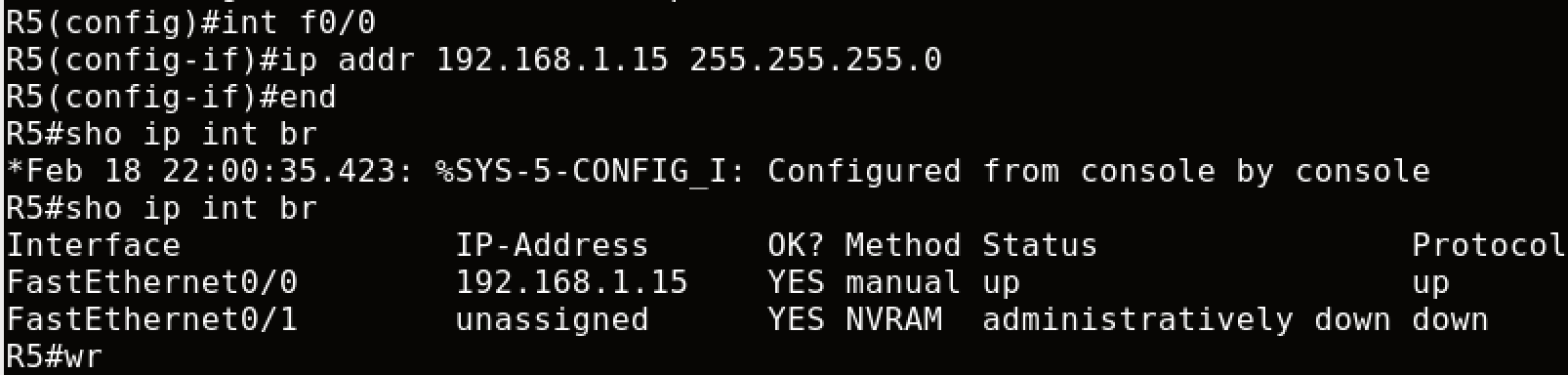
R21 MAC:



R31 MAC:

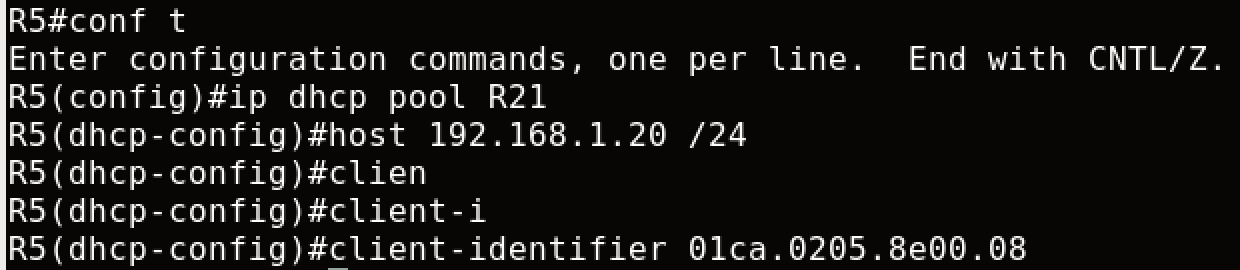


First set ip address on R5’s interface:

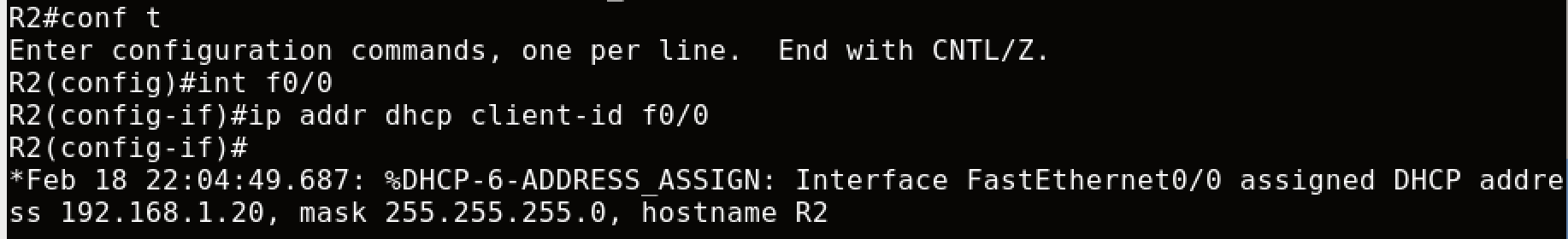


Set R21’s f0/0 MAC address to R5:

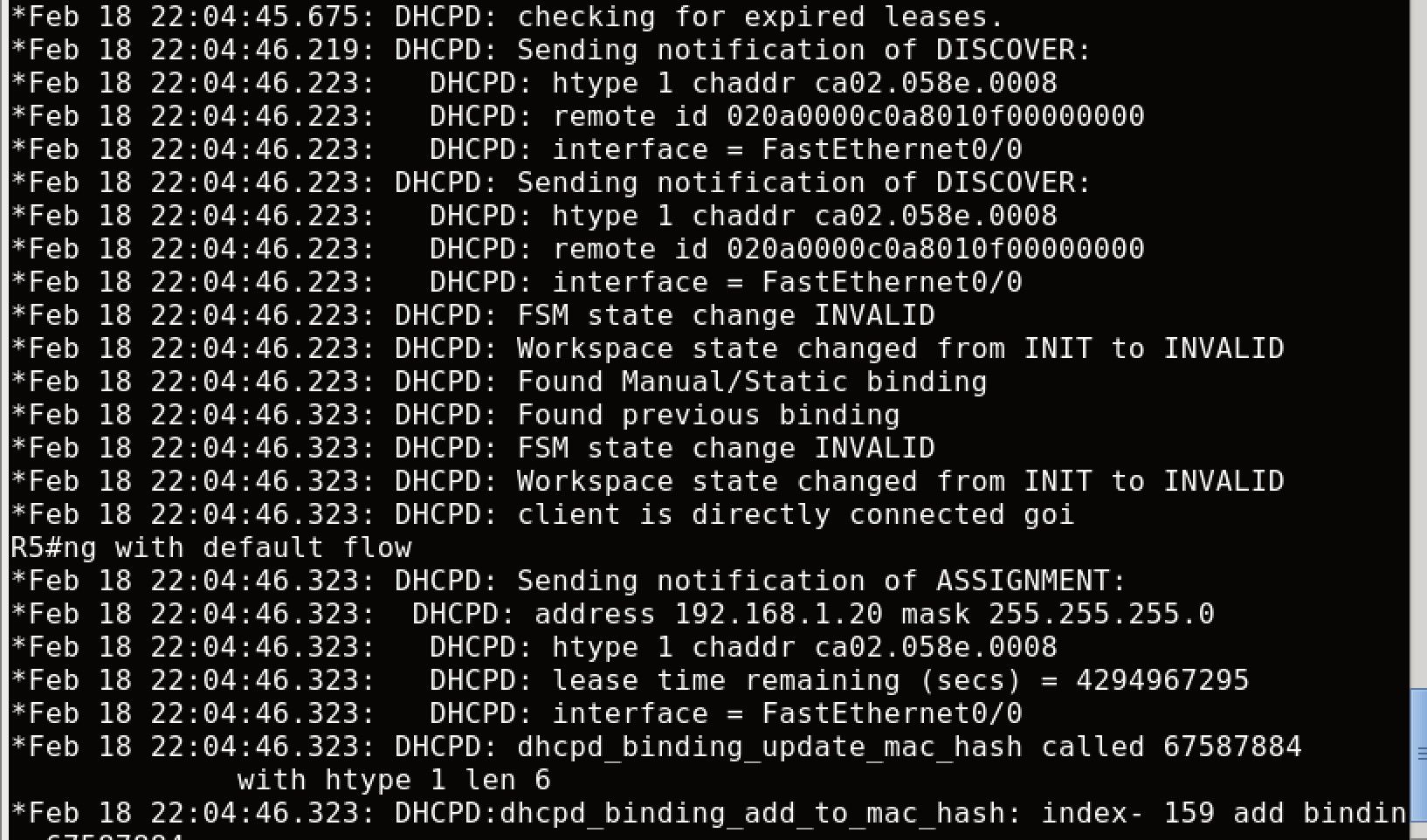
Note: R2 f0/0 MAC is: ﻿ca02.058e.0008, add 01 in front because it’s ethernet media type.



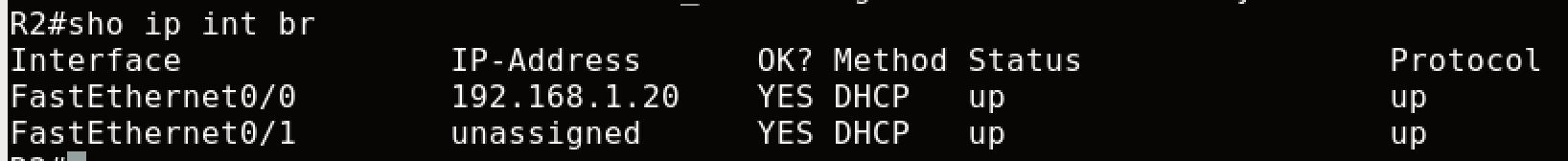
Configure R2 interface facing to R5:



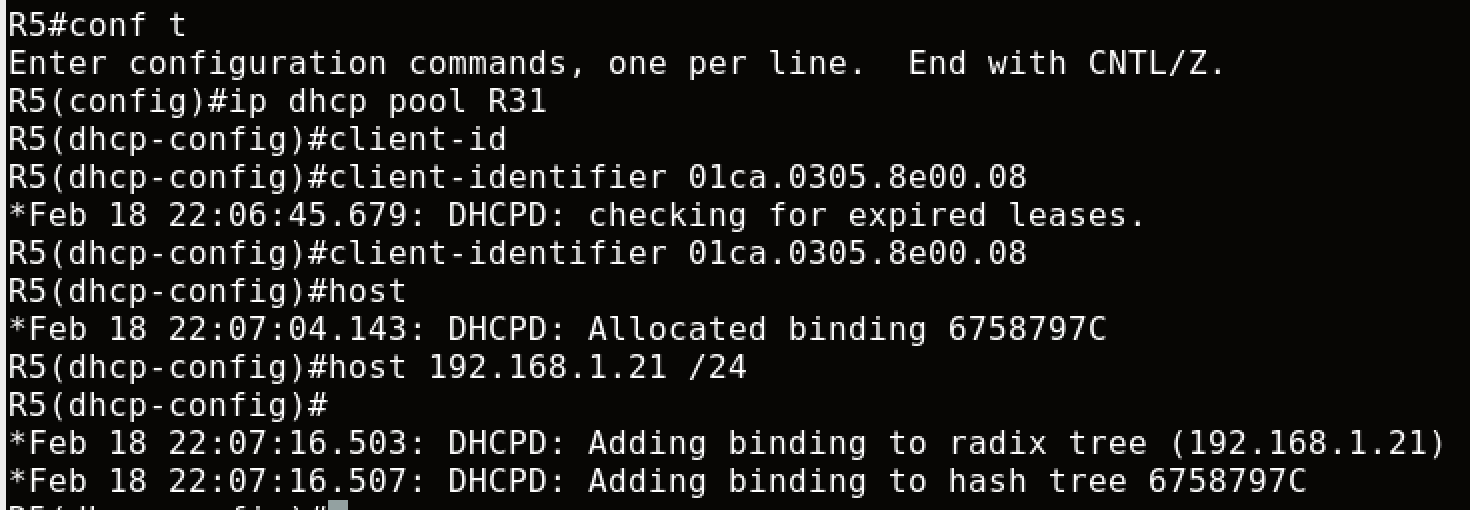
Then R5 distribute the ip address to R2:



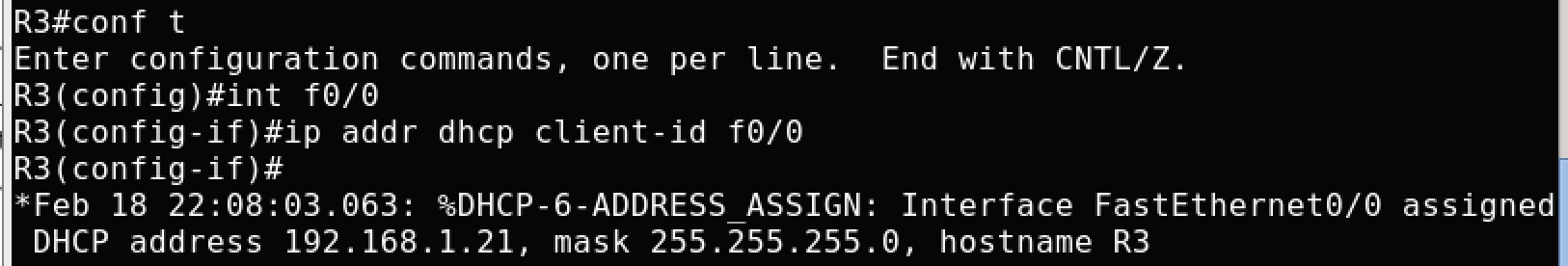
To verify on R2:



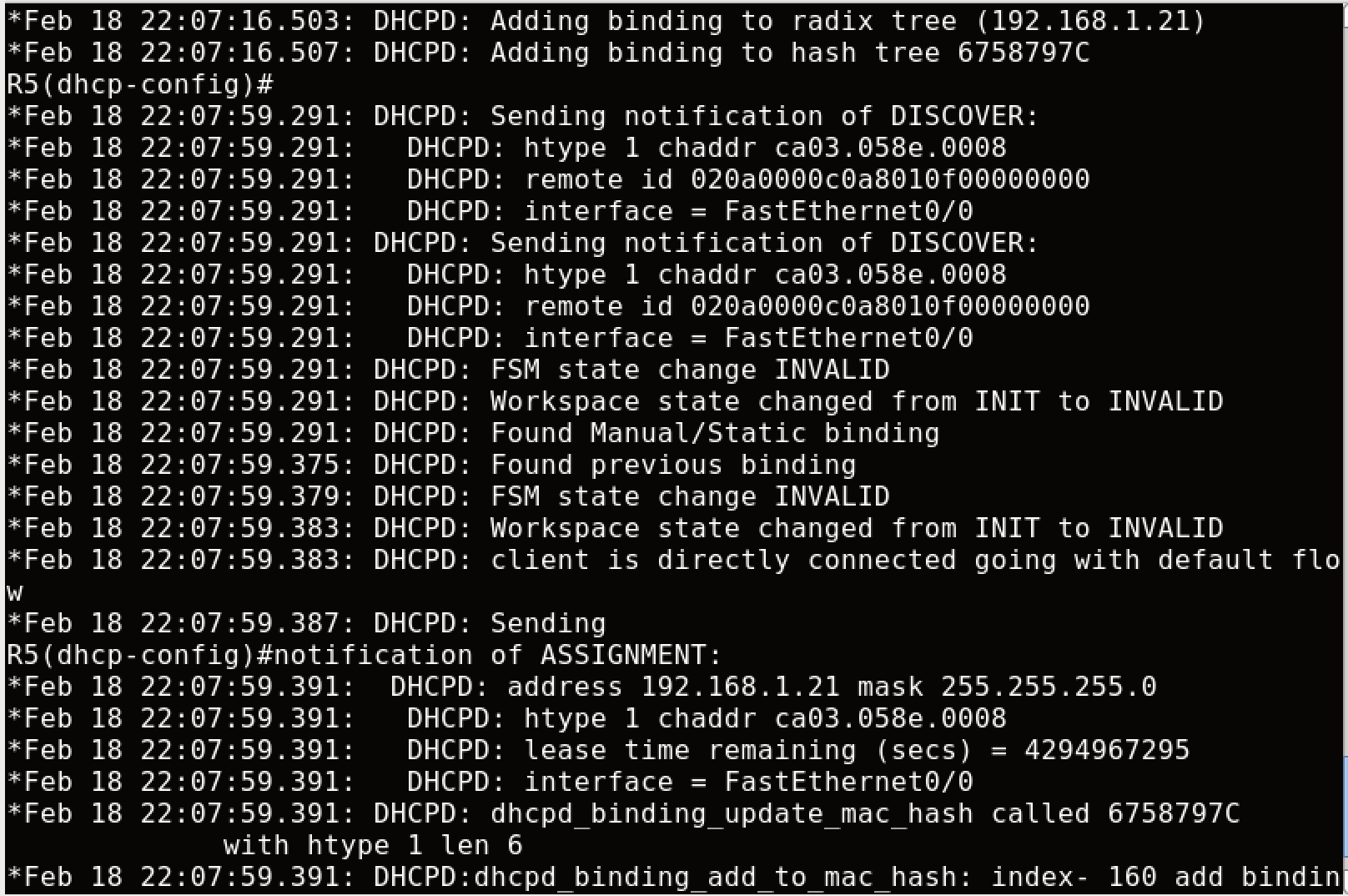
For R31:



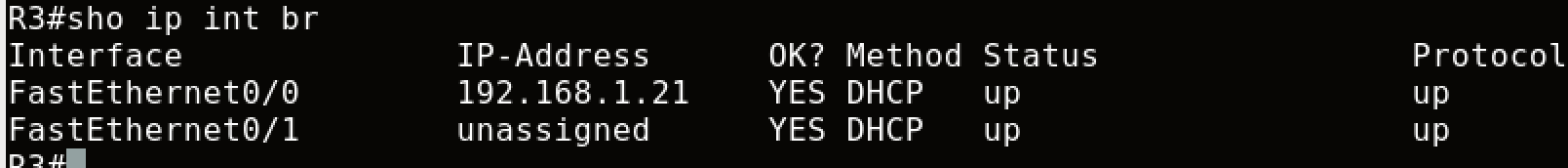
On R31:



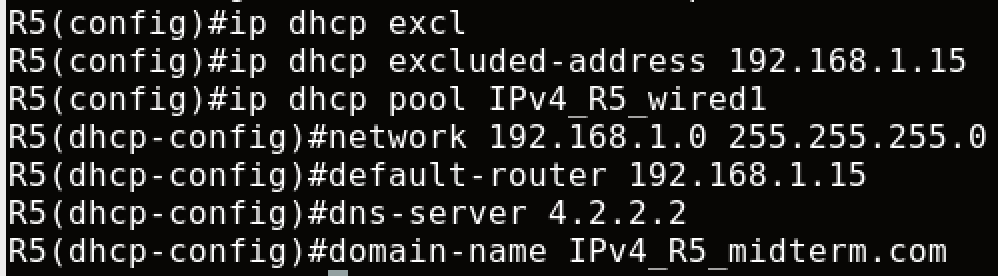
R5:



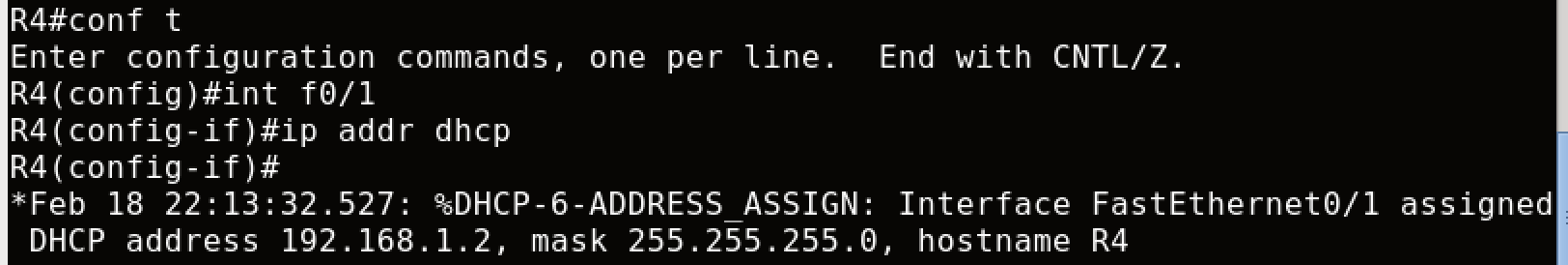
To Verify:



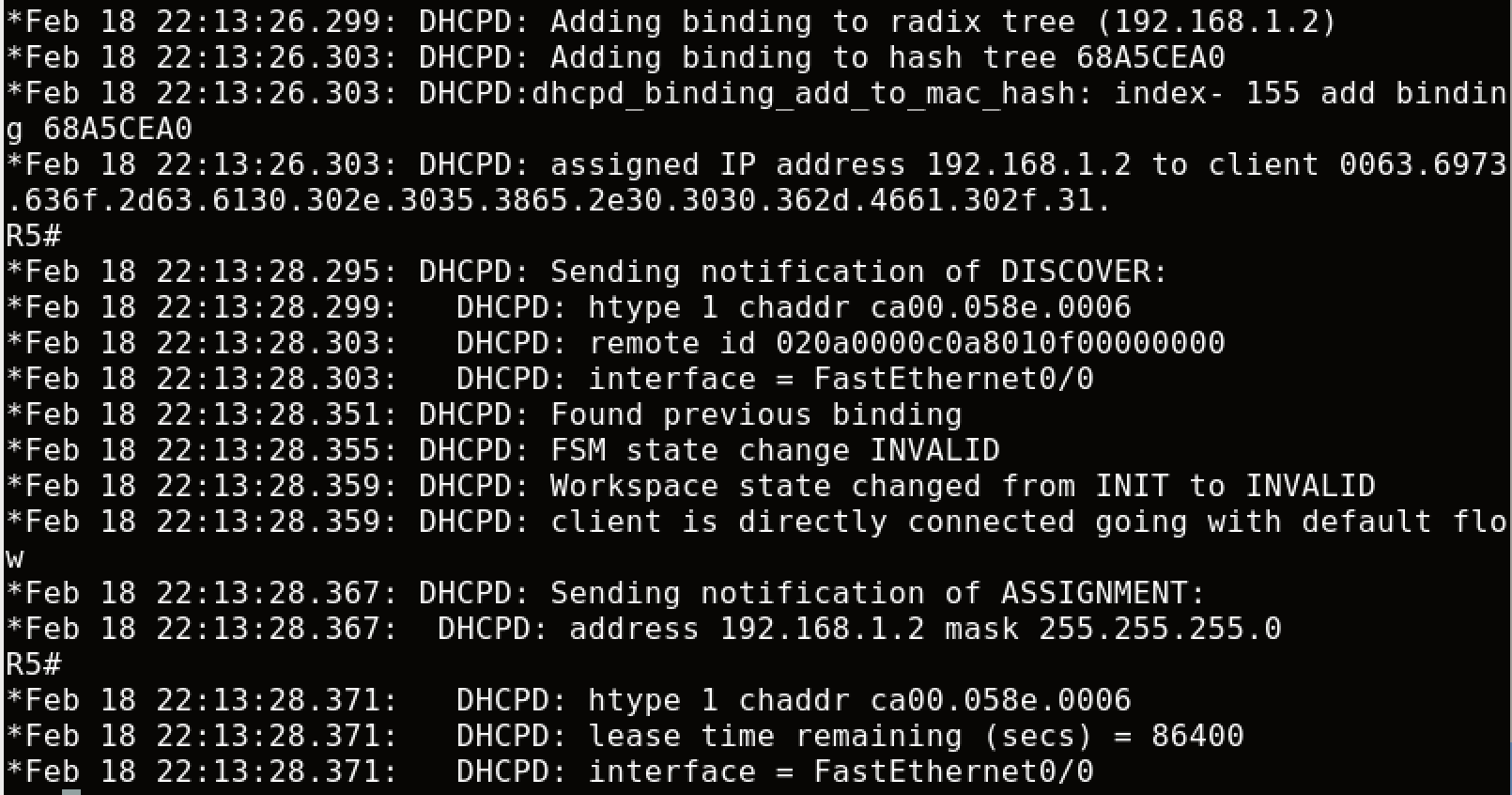
IPv4 Dynamic DHCP on R5:



IPv4 on R41:



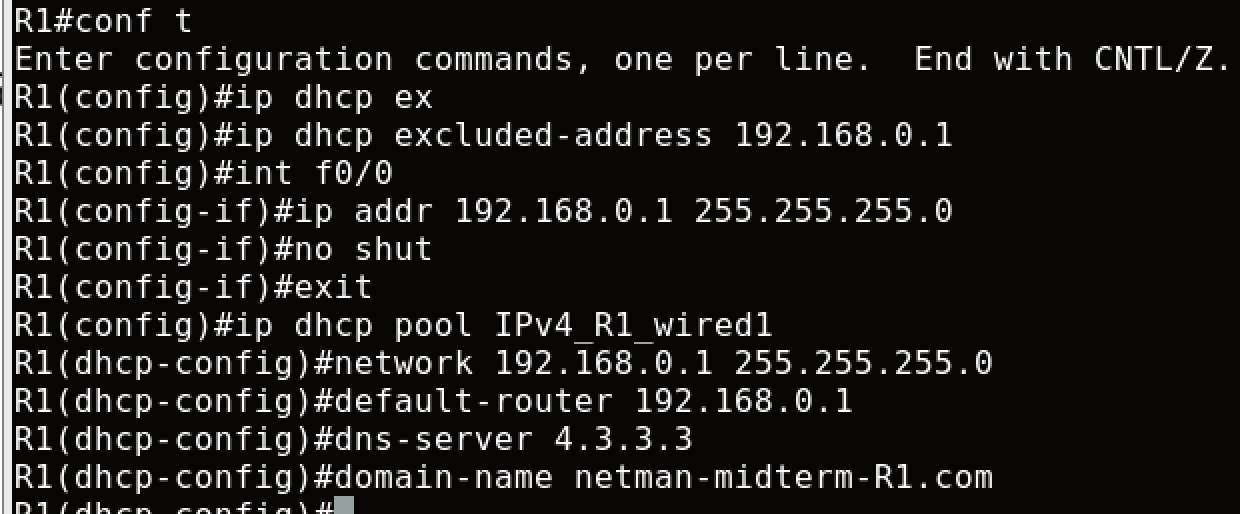
Log on R5:



1. Configure R1 to act as both DHCPv4 as well as DHCPv6 (stateful) servers to assign v4 and v6 addresses to R2-2 and R3-2. [**10 points**]

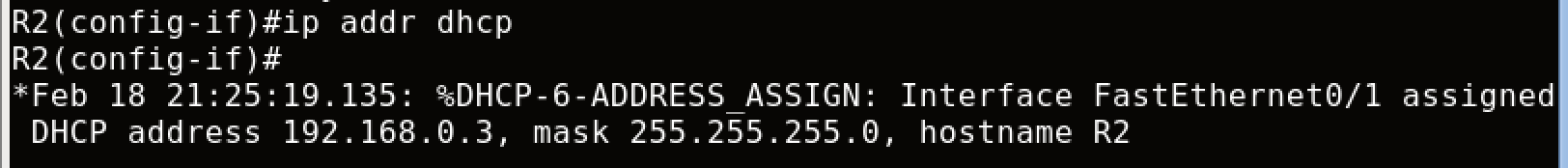
Note: c7200 doesn’t support stateful command (ipv6 address dhcp), and I confirm that we can use SLAAC for R2-2 and R3-2

DHCP for IPv4 on R1 as Server:

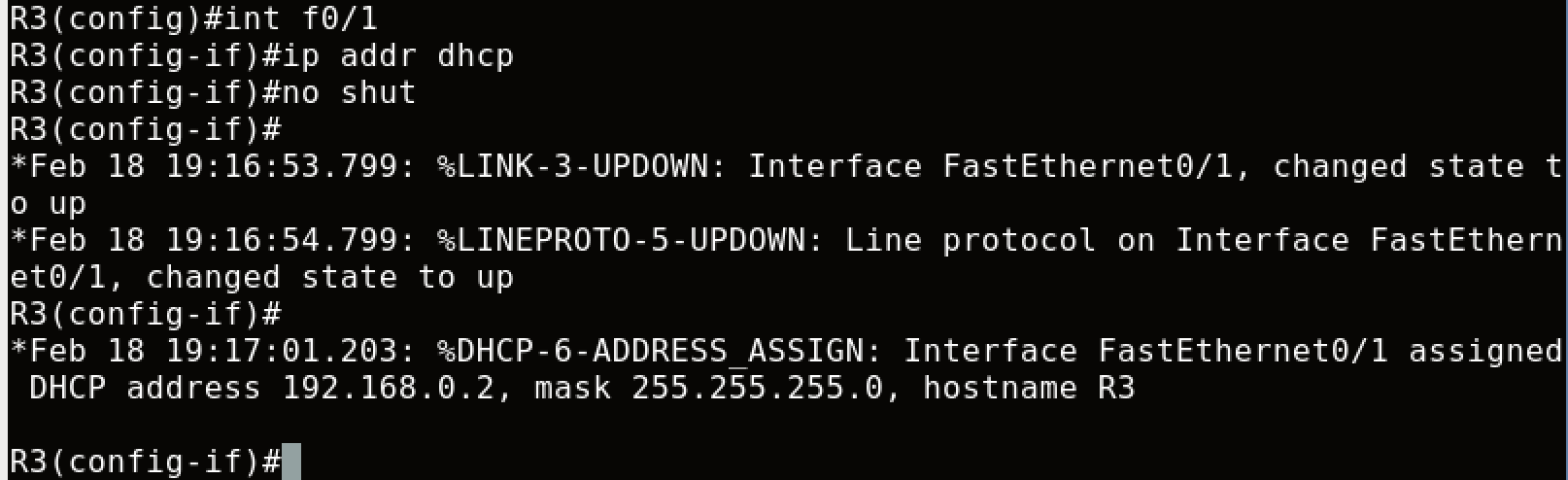


IPv4 on R22 as client:

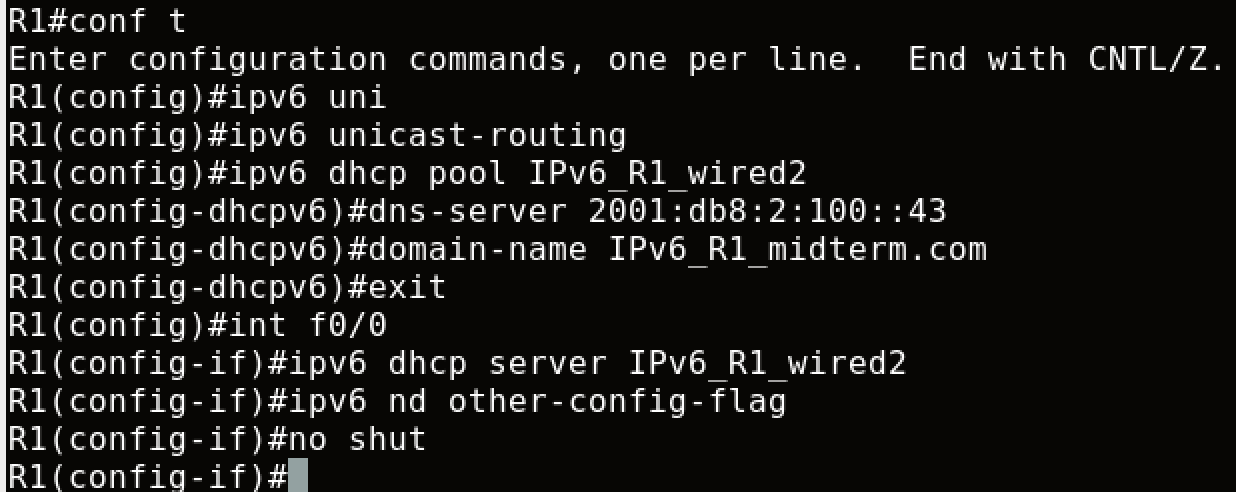


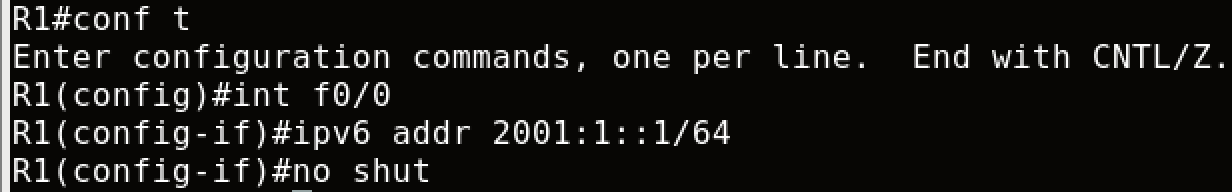


IPv4 on R32 as client:

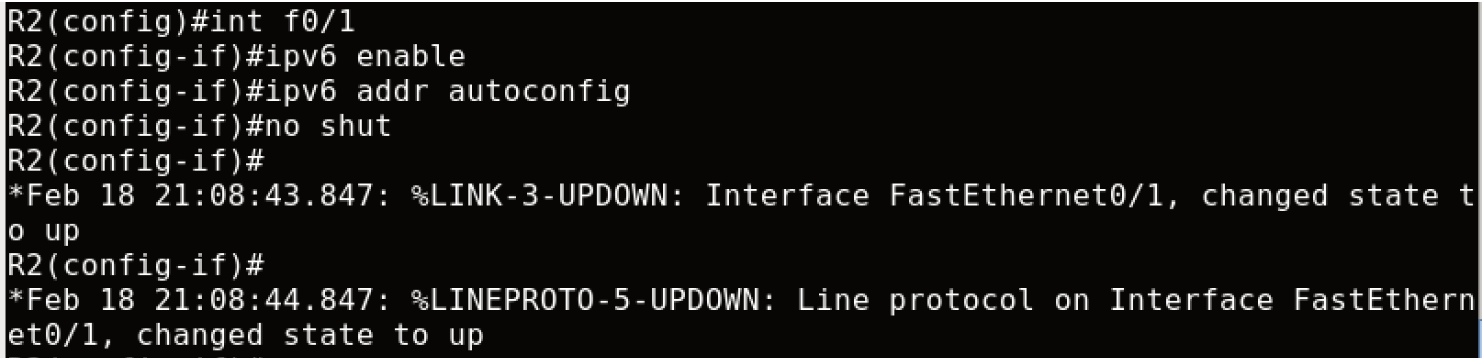


IPv6 DHCP as server on R1:

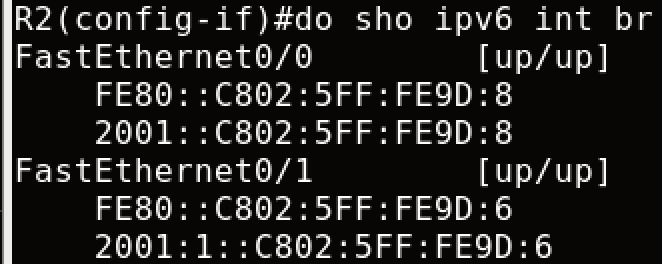




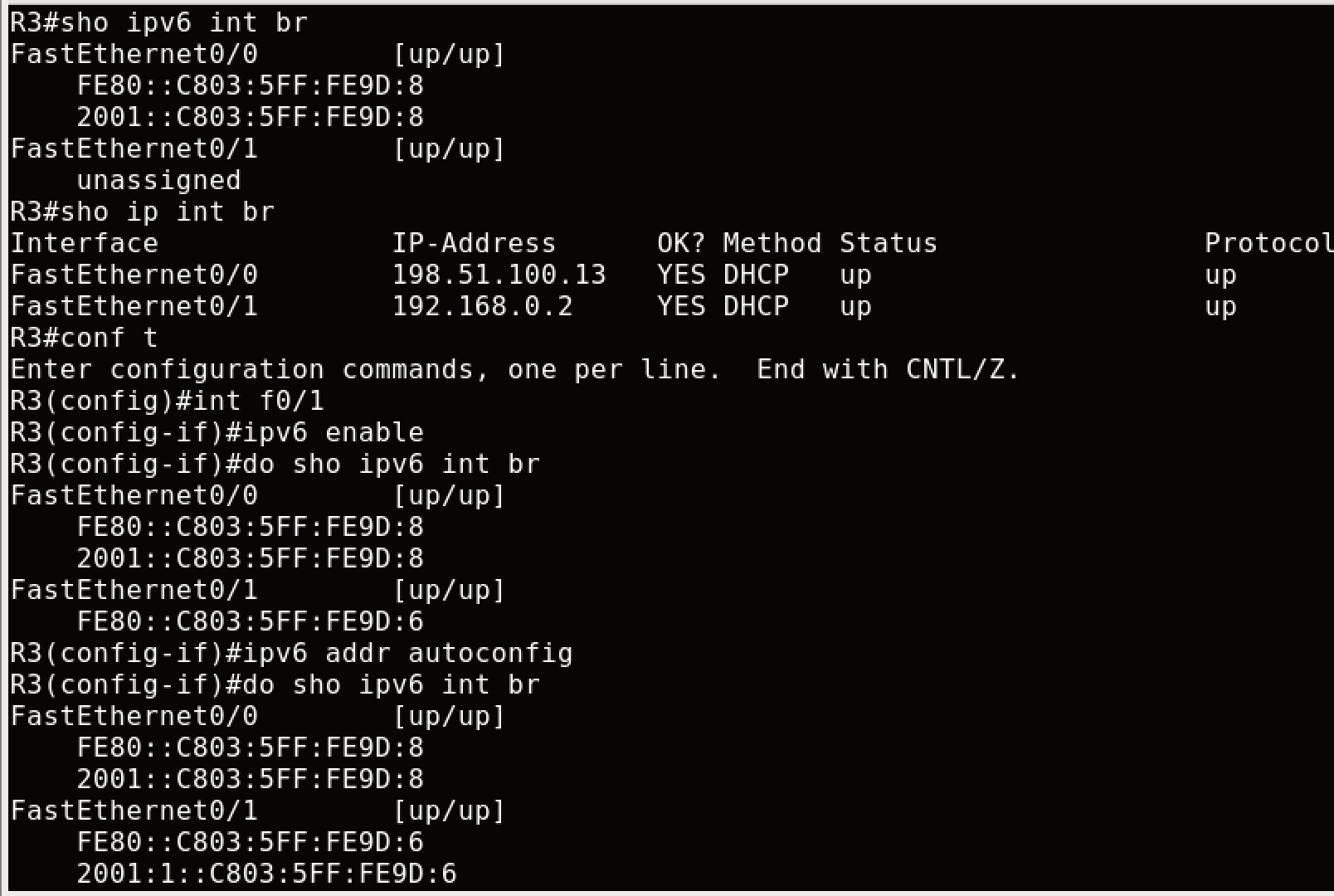
IPv6 DHCP as client on R22:



To verify on R22:



IPv6 DHCP as client on R32: (config and verify)



# 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:



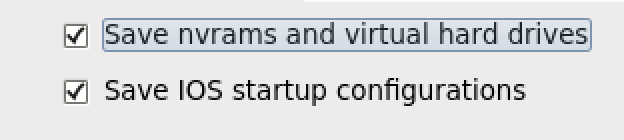
1. Store the interface status (up/down) as a dictionary inside of another dictionary in the same .txt file.

The format should be something like this-

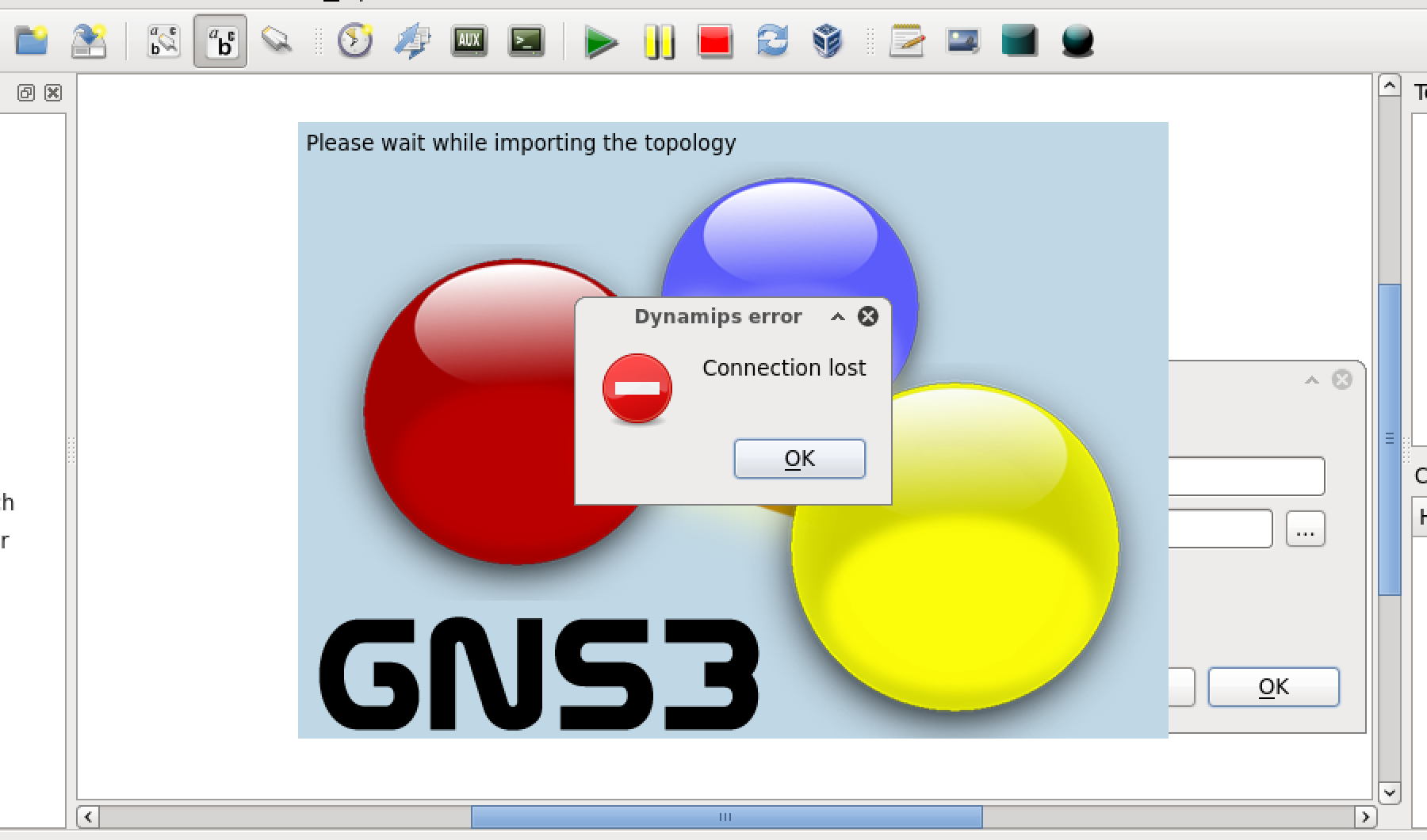


1. Fetches the CPU utilization of R1 continuously for 2 minutes in intervals of 5 seconds, plots this data as a line graph and saves the figure as .jpg file.

**Note: I have redone objective 2 and 3 for 3 times but my class VM kept crushing, so all my configurations are gone even if I have checked these two boxes every time I re-started:**



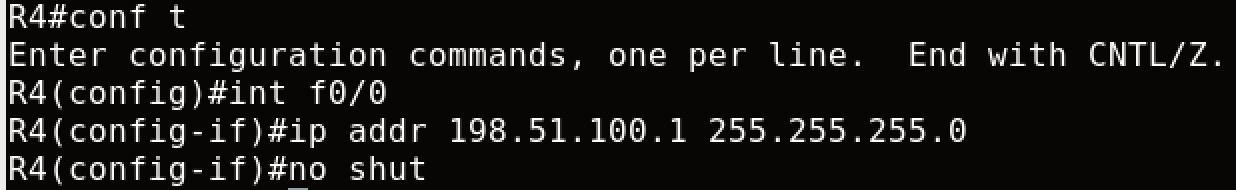
**Every time when I tried to get back where my snapshot was taken, I always got this:**



**So I had to restart from beginning… Thus, I didn’t do CPU utilization…**

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

First, I configure the same subnet ip address on the interface facing C1:



Add default route in C1 (Ubuntu)



Then, on R2, R3, R5, I added default route to R4 (interface facing R4). This way my C1 can ping R2, R3, R5.









Check if SNMP is running by `show snmp host`

Enable snmp traps:

1. (config)#snmp-server enable traps

2. (config)# snmp-server host 198.51.100.2 public

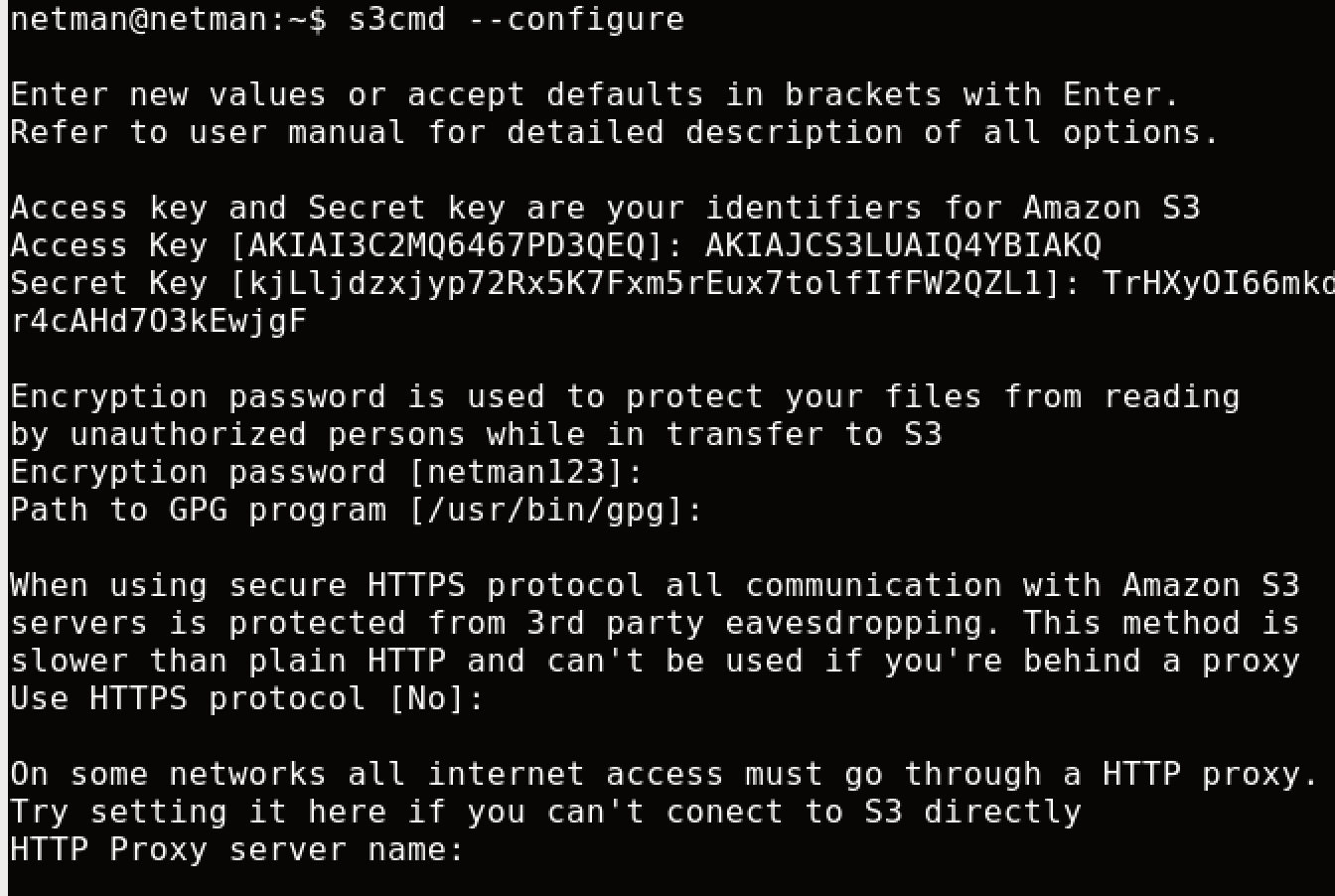
3. (config)# snmp-server community public rw

# Objective 4: AWS

Write a Python script **NMaws.py** which uses AWS SDK for Python (Boto3) which-

1. Creates a new S3 bucket called netman-lab5-<your-identity-key> [Example: netman-lab5-raga1184].

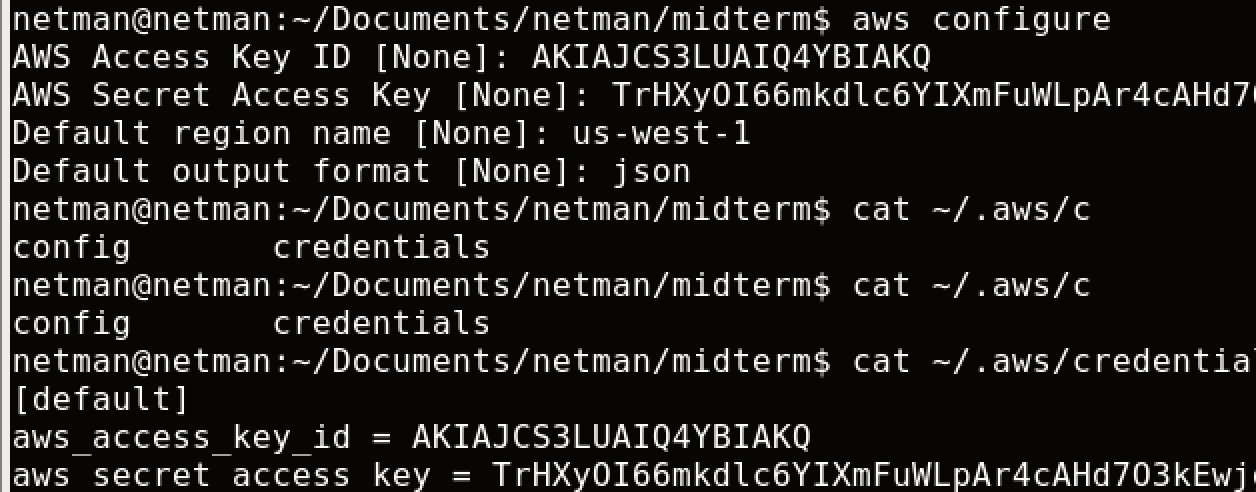
First, go to Amazon to get new Access Key and Secret Key. And do the followings:



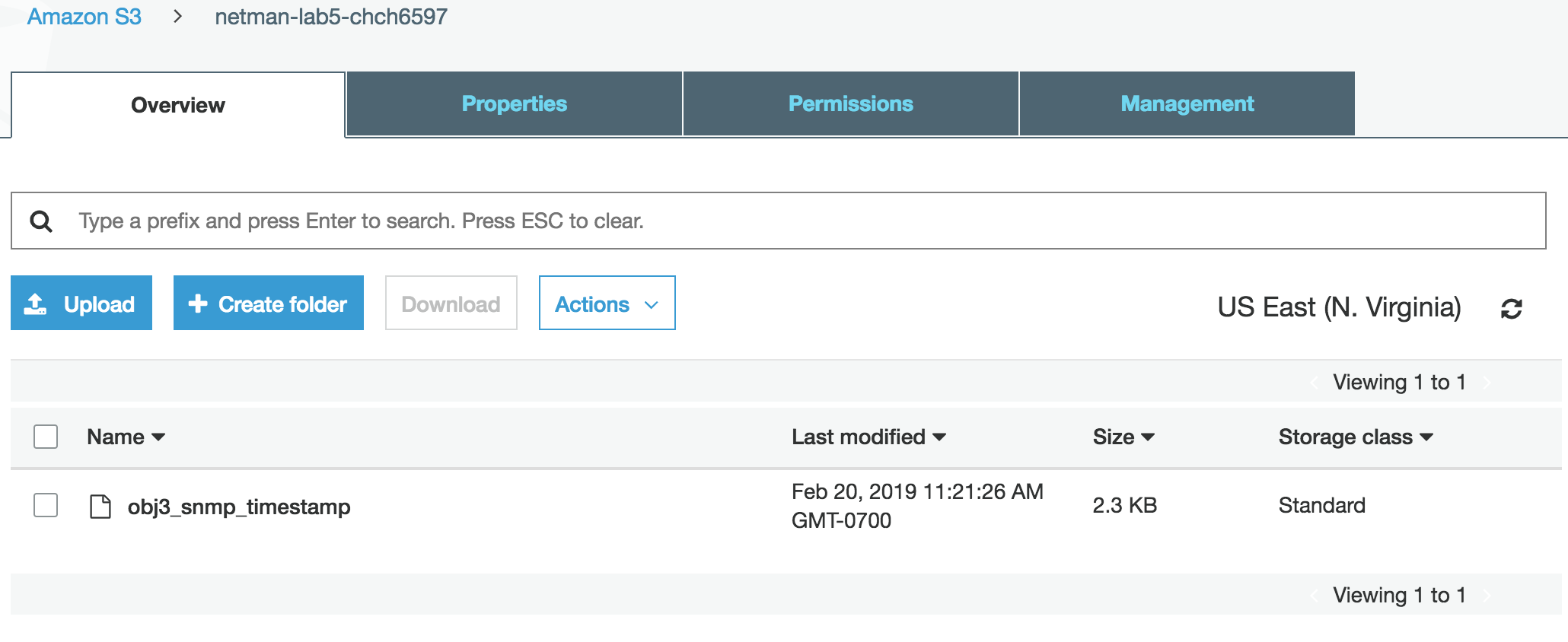
And install aws cli



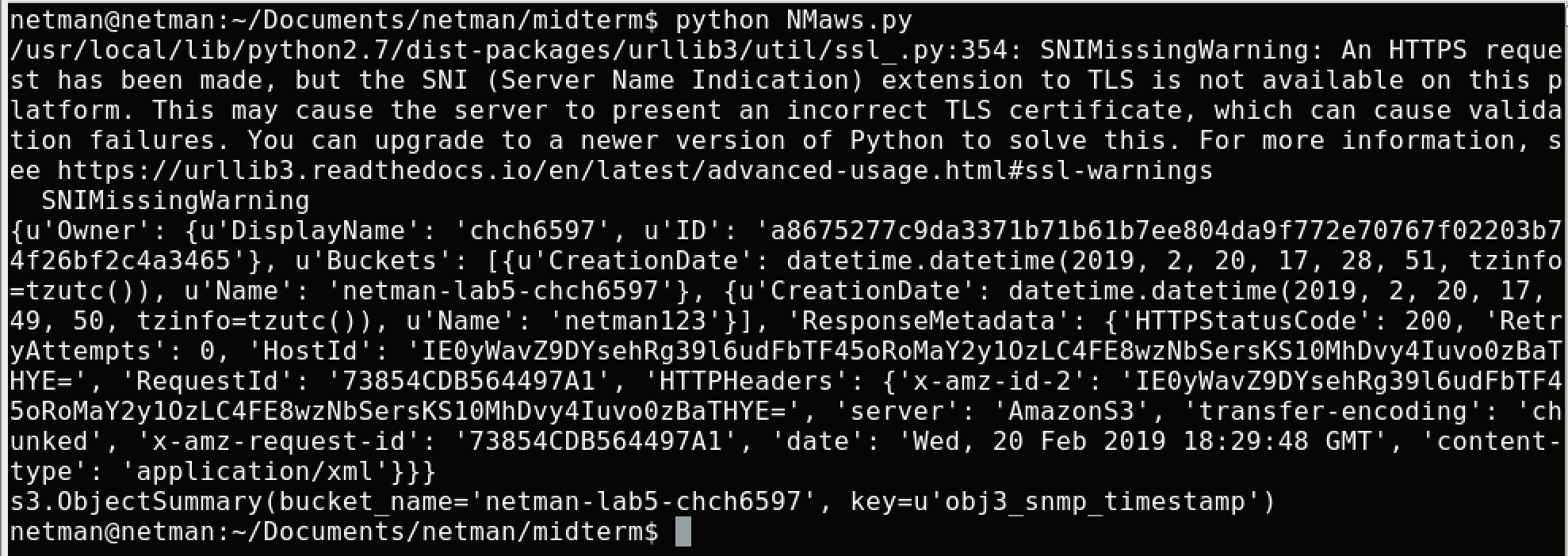
Configure default aws cli: Note: You need to install the aws cli, or you cannot upload files



1. Pushes the .txt and .jpg files from the previous objective to the bucket and saves the files with the name as: filename\_timestamp

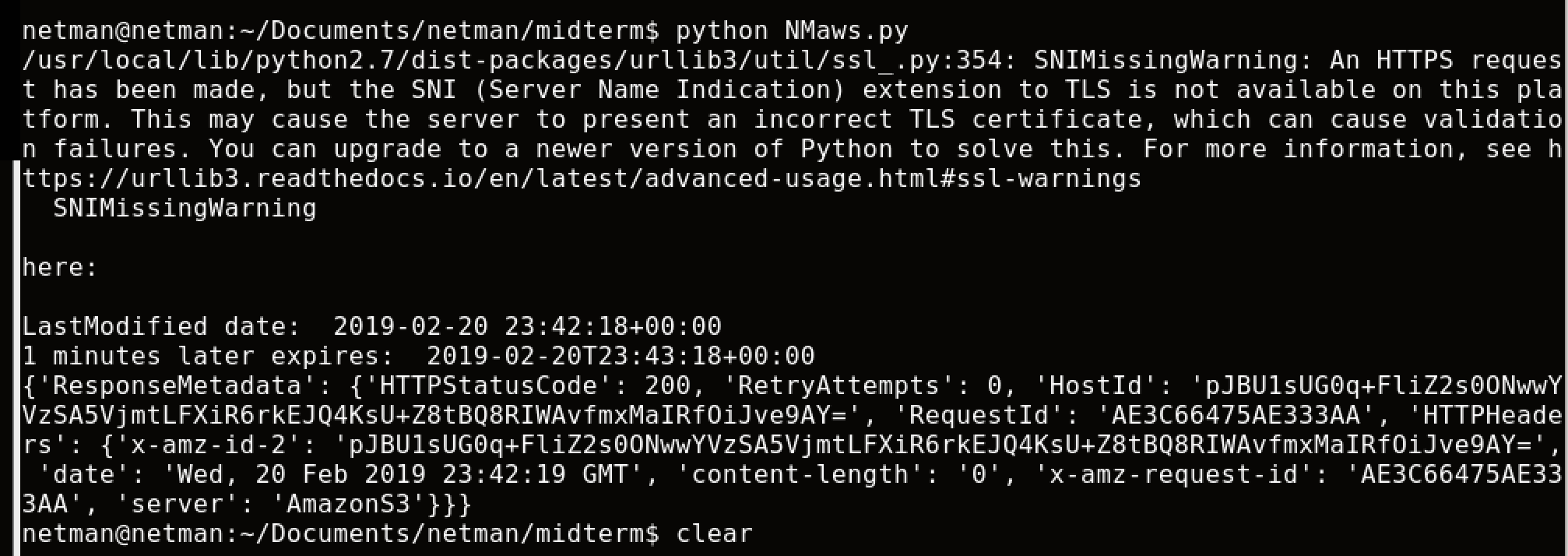


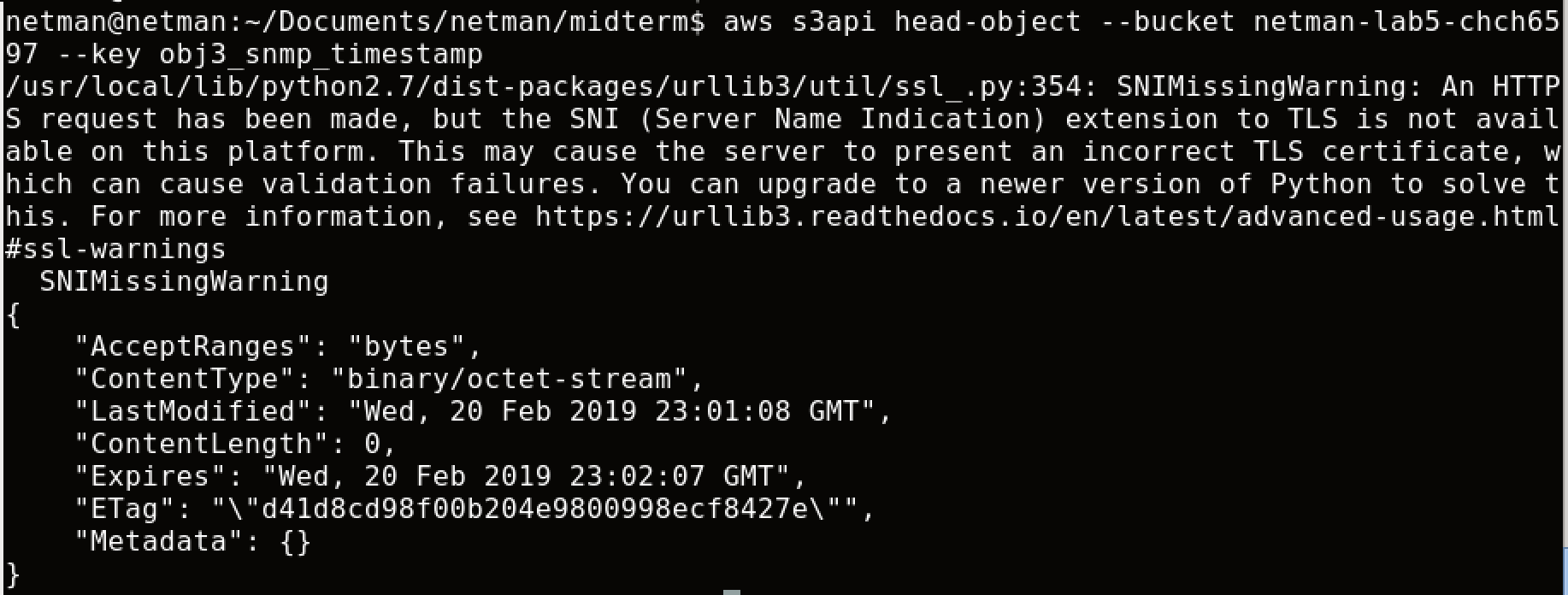
1. Lists the contents of the bucket.

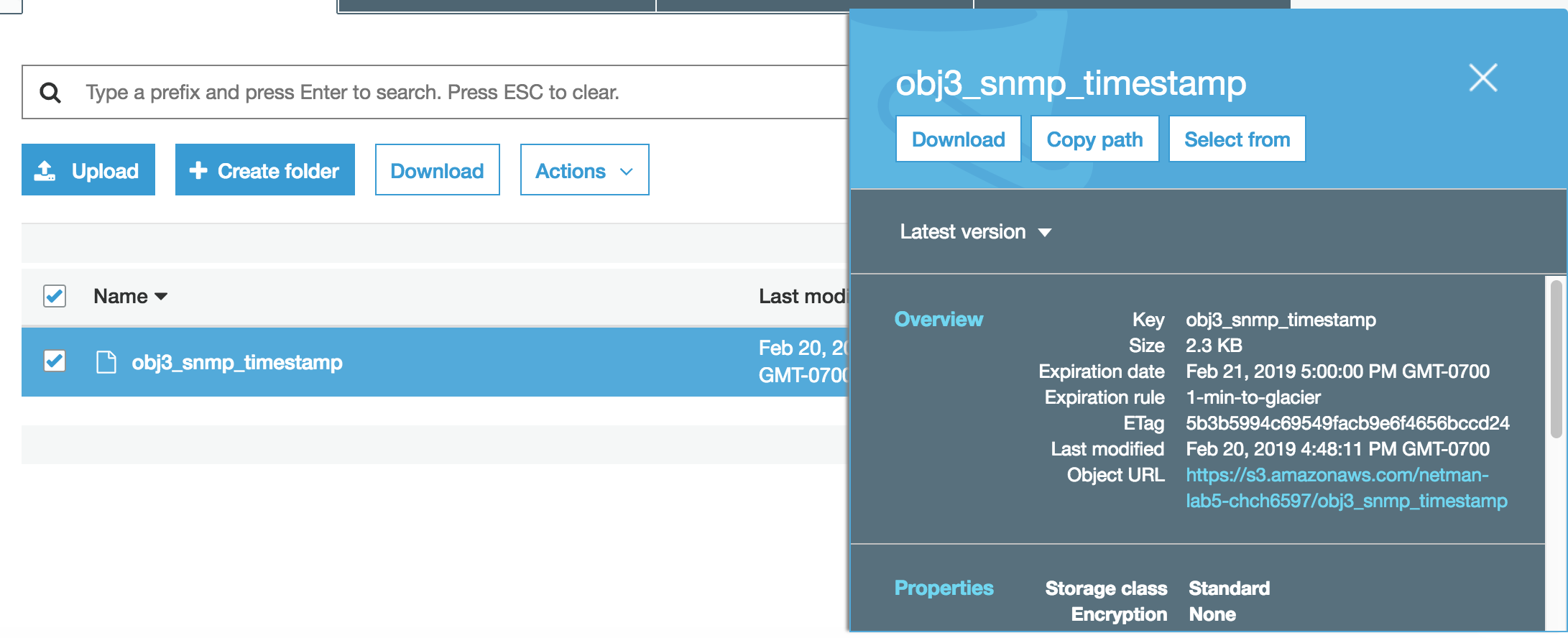


1. Checks the last-modified time of the bucket contents and removes files after a certain age (say 5 minutes for testing purposes).

Run program and print result:







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

# Total Score = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/215