COMPSCI 514 / ECE 558 - Lab2: BGP Simulation

Instructors: Xiaowei Yang and Shihan Lin
TA: Dingqi Zhang
Deadline: Nov. 1st, 2024 11:59 PM

1 Introduction

This lab requires you simulate the BGP propagation of the "Bad Gadget" topology in GNS3, a graphical network simulator. This document will instruct you to set up routers and the BGP sessions between them in GNS3. We provide you a basic BGP configuration of each router, but you will need to modify the configuration to run the Bad Gadget simulation. Finally, with the simulation results, you should write a report answering a few questions in § 4 and submit the report to Gradescope.

The directory of this lab includes some Python scripts for automatic simulation (`gns3_capture.py, gns3_deploy_topology_v1_1.py, gns3_show_appliances_v1_1.py` in `gns3_auto`), a router configuration file used by the scripts (`input/badgadget/badgadget.yaml`), and BGP configuration files for simulated routers (files in `input/badgadget/startcfgs`).

Contact the instructors or TAs if you have any question or need help.

2 Prerequisite

To work on this lab, you will need:

- Operating system: Ubuntu Desktop or MacOS
- Software: Wireshark and GNS3
- Programming language: Python3

2.1 Ubuntu or MacOS

If your local computer has Ubuntu or MacOS, then you can directly run this lab on your computer. Otherwise, you need a Virtual Machine(VM) with Graphical User Interface (GUI). You can

- Host a VM with Ubuntu on your local computer. You can use VMware or VirtualBox.
- Or reserve a free Duke VM[1], and choose Ubuntu as the OS. Use a Remote Desktop Connection (RDC)[2] client to connect the VM.

Since the lab will simulate 4 routers at the same time, we recommend to use a machine with least 4 CPU cores and 4 GiB memory. Duke VM may not support such a VM[3], so using your local computer directly or hosting a VM locally is preferred.

If you have trouble in obtaining a satisfied machine, please contact the instructors or TAs.

2.2 Wireshark and GNS3 Installation and Setup

Please download and install Wireshark and GNS3 on your machine according to the followings:

[1] https://vcm.duke.edu/
[2] https://vcm.duke.edu/help/15
[3] A Duke VM typically provides only 2 CPU cores. We did not test this lab on such a constrained environment, but it may also work for this lab, as the simulated routers are not CPU-intensive.

- Wireshark (MacOS): `https://www.wireshark.org/download.html`.
- Wireshark (Ubuntu): `sudo apt update && sudo apt install wireshark`
- GNS3: `https://www.gns3.com/software/download`

You will need to install a Cisco IOS image in GNS3 so that GNS3 can use this image to simulate a router. Please visit this link `https://github.com/hegdepavankumar/Cisco-Images-for-GNS3-and-EVE-NG` and download the router image (`c7200-adventerprisek9-mz.152-4.S6.image`) in the link.

For the detail of the image setup in GNS3. Please watch the video at this link.

For MacOS, you also need to create a soft link for `mergecap` after installing Wireshark:

```
sudo ln -sf /Applications/Wireshark.app/Contents/MacOS/mergecap /usr/local/bin/
```

2.3 Required Packages

You will need the following Python Packages to run the provided Python scripts:

```
pip3 install requests prettytable pexpect gns3fy netmiko
```

3 Network Model

This lab uses a Bad Gadget topology shown in Figure 1. If you are not familiar with the Bad Gadget topology, please refer to the lecture slides. For simplicity, each router also represents an Autonomous System (AS): R1 for AS1, R2 for AS2, etc. The links show the BGP sessions between routers, and the IP addresses of two connected interfaces are attached.

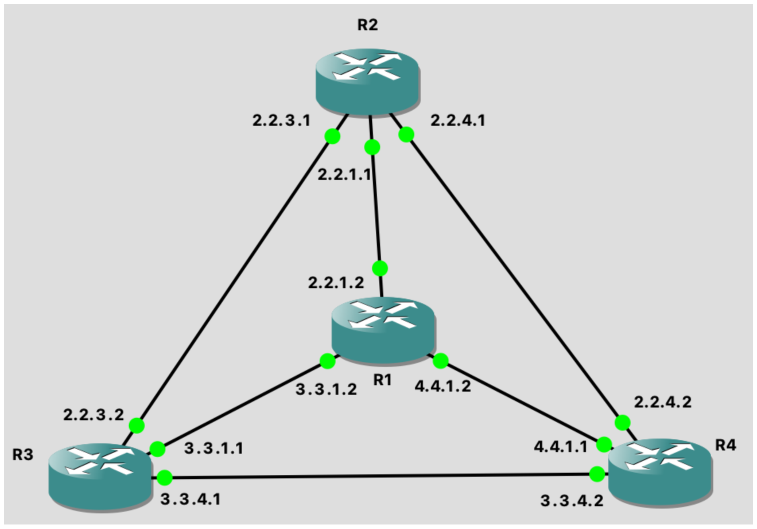
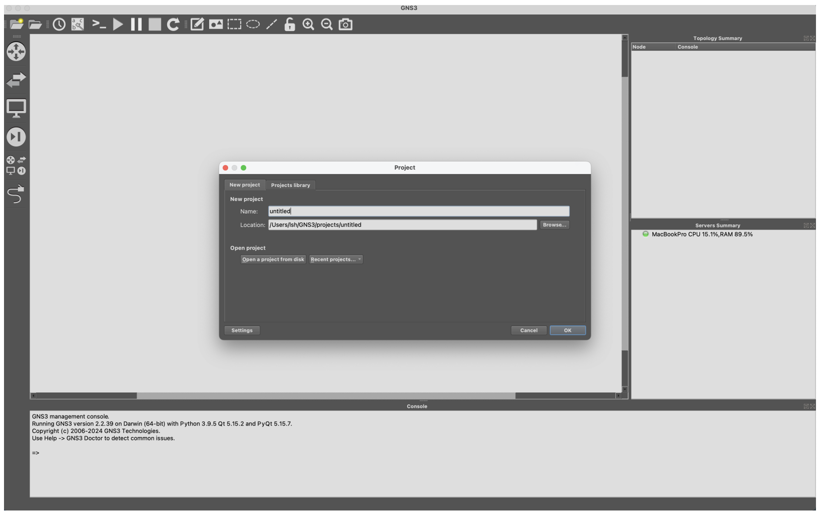


Figure 1: Network Model: Bad Gadget Topology

4 Simulation

Firstly, open the GNS3 software, and you should see a window like the picture below without errors.



New project

Name: untitled

Location: /Users/lsh/GNS3/projects/untitled

Open project

Servers Summary

MacBookPro CPU 15.1%,RAM 89.5%

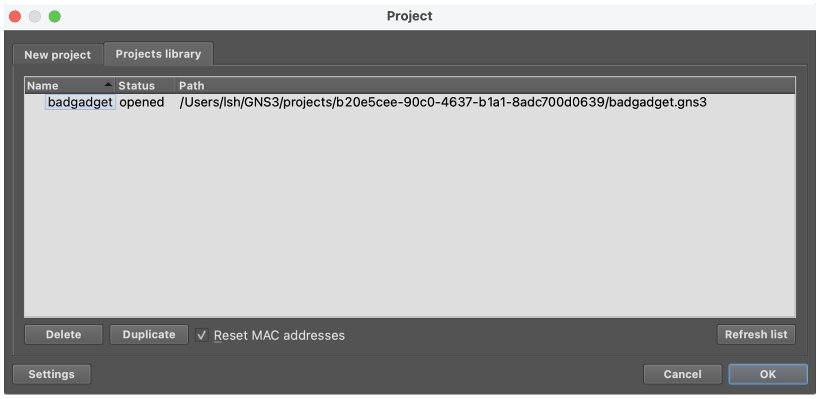
```
GNS3 management console.
Running GNS3 version 2.2.39 on Darwin (64-bit) with Python 3.9.5 Qt 5.15.2 and PyQt 5.15.7.
Copyright (c) 2006-2024 GNS3 Technologies.
Use Help -> GNS3 Doctor to detect common issues.

=>
```

You do not need to manually create the project. Instead, we provide a script `gns3_auto/gns3_deploy_topology_v1_1.py` to build it from the configuration files in `input/badgadget`. Thus, you should run the following command in a terminal:

```
python3 gns3_auto/gns3_deploy_topology_v1_1.py -f input/badgadget/badgadget.yaml
```

This script will run for about 10 minutes. After the script finished, you should be able to see a project named “badgadget” in GNS3, as shown below. You may need to click the “Refresh list” button to see it.



Name	Status	Path
badgadget	opened	/Users/lsh/GNS3/projects/b20e5cee-90c0-4637-b1a1-8adc700d0639/badgadget.gns3

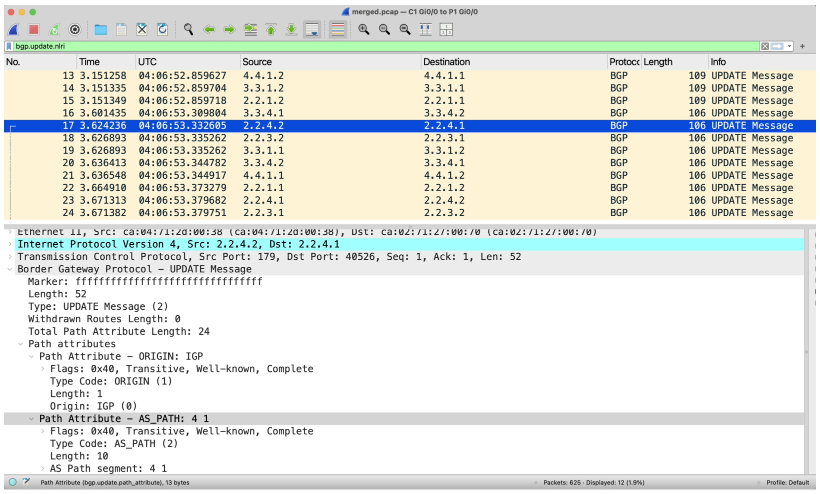
✓ Reset MAC addresses

You do not have to open the project manually, and you can use the script `gns3_auto/gns3_capture.py` to run the simulation.

```
python3 gns3_auto/gns3_capture.py badgadget
```

The script will run for about 5 minutes, and it may open a Wireshark window automatically. Please wait for the script stops, and the Wireshark window will be closed automatically. The script will produce some pcap files in `output/badgadget`. There is a `merged.pcap` which merges the other pcap files into one single file. You can focus on reading this file for this lab.

Use Wireshark to open `merged.pcap`, and use the filter command `bgp.update.nlri` in Wireshark to filter those BGP update messages. You will see something similar as below:


merged.pcap — C1 Gi0/0 to P1 Gi0/0

bgp.update.nlri

```
No.    Time      UTC              Source    Destination  Protocol Length Info
    13 3.151258  04:06:52.859627  4.4.1.2   4.4.1.1      BGP      109 UPDATE Message
    14 3.151335  04:06:52.859704  3.3.1.2   3.3.1.1      BGP      109 UPDATE Message
    15 3.151349  04:06:52.859718  2.2.1.2   2.2.1.1      BGP      109 UPDATE Message
    16 3.601435  04:06:53.309804  3.3.4.1   3.3.4.2      BGP      106 UPDATE Message
    17 3.624236  04:06:53.332605  2.2.4.2   2.2.4.1      BGP      106 UPDATE Message
    18 3.626893  04:06:53.335262  2.2.3.2   2.2.3.1      BGP      106 UPDATE Message
    19 3.626893  04:06:53.335262  3.3.1.1   3.3.1.2      BGP      106 UPDATE Message
    20 3.636413  04:06:53.344782  3.3.4.2   3.3.4.1      BGP      106 UPDATE Message
    21 3.636548  04:06:53.344917  4.4.1.1   4.4.1.2      BGP      106 UPDATE Message
    22 3.664910  04:06:53.373279  2.2.1.1   2.2.1.2      BGP      106 UPDATE Message
    23 3.671313  04:06:53.379682  2.2.4.1   2.2.4.2      BGP      106 UPDATE Message
    24 3.671382  04:06:53.379751  2.2.3.1   2.2.3.2      BGP      106 UPDATE Message
```

```
Ethernet II, Src: ca:04:71:2d:00:38 (ca:04:71:2d:00:38), Dst: ca:02:71:27:00:70 (ca:02:71:27:00:70)
Internet Protocol Version 4, Src: 2.2.4.2, Dst: 2.2.4.1
Transmission Control Protocol, Src Port: 179, Dst Port: 40526, Seq: 1, Ack: 1, Len: 52
Border Gateway Protocol - UPDATE Message
  Marker: ffffffffffffffffffffffffffffffff
  Length: 52
  Type: UPDATE Message (2)
  Withdrawn Routes Length: 0
  Total Path Attribute Length: 24
  Path attributes
    Path Attribute - ORIGIN: IGP
      Flags: 0x40, Transitive, Well-known, Complete
      Type Code: ORIGIN (1)
      Length: 1
      Origin: IGP (0)
    Path Attribute - AS_PATH: 4 1
      Flags: 0x40, Transitive, Well-known, Complete
      Type Code: AS_PATH (2)
      Length: 10
      AS Path segment: 4 1
```

Path Attribute (bgp.update.path_attribute), 13 bytes   Packets: 625 · Displayed: 12 (1.9%)   Profile: Default

Now, here are the questions you need to think about and answer in your report.

Q1. If you read the BGP configuration for each router in `input/badgadget/startcfgs`, you will find that we set all BGP neighbor with the same local preference 100 currently. For example, in the file `input/badgadget/startcfgs/R2`, you can see

```
neighbor 2.2.1.2 route-map MYPREF1 in
...
route-map MYPREF1 permit 10
    set local-preference 100
```

This configuration means setting the local preference of BGP routes from the neighbor with IP 2.2.1.2 (AS1) as 100.

Since all neighbors have the same local preference currently, the BGP propagation should converge. Thus, our question is: what is the AS path each router uses to reach R1? How do you confirm it from the pcap file?

Q2. Please the edit the local preference values in the configuration files of routers in `input/badgadget/startcfgs` to make this topology reflects the non-convergence behavior presented in the lecture slides. Then delete the current project in GNS3, rerun the Python scripts to create a new project, and rerun the simulation.

```
python3 gns3_auto/gns3_deploy_topology_v1_1.py -f input/badgadget/badgadget.yaml
python3 gns3_auto/gns3_capture.py badgadget
```

Please provide the screenshot of the new `output/badgadget/merged.pcap` in Wireshark in your report. Whether the BGP converge or not in the new configuration? How do you know its convergence from the pcap file? How do you edit the local preference values to make this happen?

Q3. Is there an time period for BGP update? How long is it? Why this period is necessary or helpful for BGP propagation?

Q4. According to what we learned in the lecture, how does ASes in reality solve such a non-convergence issue? Just explain the method and principles, and you do not need to implement it in this lab.

5 Submission

You only need to submit your report to Gradescope. You must finish this lab individually. It is okay to discuss with anyone, but please do not look at other students' solutions.