BÁO CÁO ĐỒ ÁN

**Môn học: QUẢN TRỊ MẠNG & HỆ THỐNG**

**Tên chủ đề: Python & Thư viện Netmiko**

*GVHD: Nguyễn Khánh Thuật*

1. **THÔNG TIN CHUNG:**

**Lớp: NT132.N23.MMCL**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Đặng Xuân Cường | 21521899 | 21521899@gm.uit.edu.vn |
| 2 | Nguyễn Văn Huấn | 21522108 | 21522108@gm.uit.edu.vn |

1. **ĐÁNH GIÁ KHÁC:**

|  |  |
| --- | --- |
| **Nội dung** | **Kết quả** |
| Nhận xét của giảng viên:  + Ưu điểm:  + Cải thiện: |  |
| Điểm |  |

**Phần bên dưới của báo cáo này là báo cáo chi tiết của nhóm thực hiện.**

Contents

[**I/ Giới thiệu khái quát về thư viện Netmiko trong Python** 3](#_Toc135592938)

[**II/ Cách thức hoạt động và ứng dụng của Netmiko** 3](#_Toc135592939)

[1/ Cấu hình thiết bị 3](#_Toc135592940)

[2/ Bắt đầu với Netmiko 7](#_Toc135592941)

[a/ Kết nối đến thiết bị mạng 7](#_Toc135592942)

[b/ Cấu hình địa chỉ IP lên các interface của router R1 7](#_Toc135592943)

[c/ Tìm số các interface bị shutdown của router R1 9](#_Toc135592944)

[3/ Tự động hoá với Netmiko (Automation) 11](#_Toc135592945)

[a/ Gửi lệnh hiển thị đến nhiều thiết bị 11](#_Toc135592946)

[b/ Gửi lệnh ở mode config đến nhiều thiết bị 14](#_Toc135592947)

[4/ TextFSM 15](#_Toc135592948)

[a/ TextFSM là gì? 15](#_Toc135592949)

[b/ Ứng dụng của TextFSM 16](#_Toc135592950)

BÁO CÁO CHI TIẾT

---

# **I/ Giới thiệu khái quát về thư viện Netmiko trong Python**

- Netmiko là thư viện được phát triển bởi Kirk Byers (một chuyên gia người Mỹ về Network Automation Training)

- Netmiko được thiết kế cho các kỹ sư mạng và cung cấp các giao diện, phương thức để kết nối và điều khiển các thiết bị mạng của Cisco như Router, Switch, Firewall, …

- Netmiko nhẹ, dễ sử dụng, phù hợp với nhiều mục đích khác nhau trong việc tương tác với thiết bị mạng.

***\* Cách cài đặt thư viện Netmiko trên Visual Studio Code: mở giao diện Terminal và nhập dòng lệnh ‘pip install netmiko’***

# **II/ Cách thức hoạt động và ứng dụng của Netmiko**

* Ứng dụng của Netmiko

- Đơn giản hoá thiết lập kết nối SSH và tương tác với thiết bị mạng bằng ngôn ngữ Python

- Hỗ trợ kết nối đến nhiều nền tảng thiết bị khác nhau của Cisco như IOS, IOS XE, NX-OS, IOS-XR, …

- Có thể kết nối đồng thời nhiều thiết bị và thực hiện cấu hình

- Được xây dựng dựa trên Python, hỗ trợ việc phát triển ứng dụng tự động hoá mạng

- Được phát triển và cập nhật liên tục nên ngày càng được cải tiến

## 1/ Cấu hình thiết bị

Để sử dụng được Netmiko thì phải có một sơ đồ mạng và một nền tảng mô phỏng các thiết bị mạng. Ở đây, ta sử dụng nền tảng mô phỏng EVE. Ví dụ một sơ đồ mạng trên EVE như sau:

A picture containing screenshot, diagram

Description automatically generated

*Hình 1: Sơ đồ mạng*

- Ở hình 1, ta sử dụng router R1 kết nối với cloud mạng với interface e0/0.

- Cấu hình cho router R1 như sau:

A screenshot of a computer screen

Description automatically generated with medium confidence

*Hình 2: Cấu hình router R1*

\* Các bước cấu hình lần lượt như sau:

+ Đặt hostname cho router là R1

+ Đặt username và password: username admin, password netmiko

+ Tạo IP domain-name: test1.com

+ Tạo tập khóa public-private để mã hõa dữ liệu bằng RSA

+ 512 là mặc định số bit mã hóa, nhưng SSH cần từ 768 đến 4096 (Giá trị càng cao thì key càng khó mở): ở ví dụ sẽ chọn 1024 bits

A screenshot of a computer

Description automatically generated with medium confidence

*Hình 3: Thêm mật khẩu để kích hoạt router R1*

- login local: Phương thức xác thực khi cấu hình ở vty, cho phép sử dụng cơ sở dữ liệu trên thiết bị để đăng nhập thay cho password ở hình 2

- transport input ssh: cho phép thực hiện cài đặt SSH lên router

- enable secret cisco: đặt mật khẩu để kích hoạt router

A screenshot of a computer

Description automatically generated

*Hình 4: Cài đặt SSH cho router R1*



*Hình 5: Dùng DHCP để cấp IP cho interface e0/0*

A screenshot of a computer screen

Description automatically generated with low confidence

*Hình 6: Kiểm tra IP của các interface router R1*

- Ở hình 6: địa chỉ IP 192.168.1.130 đã được cấp bằng giao thức DHCP lên interface e0/0

A screenshot of a computer program

Description automatically generated with medium confidence

*Hình 7: Kiểm tra kết quả*

- Ở hình 7: sử dụng CMD để ping đến IP của interface e0/0 và kiểm tra SSH đã được thiết lập trên R1 chưa và kết quả trả về là thành công qua việc yêu cầu password để truy cập vào R1

## 2/ Bắt đầu với Netmiko

### a/ Kết nối đến thiết bị mạng

A screen shot of a computer

Description automatically generated with medium confidence*Hình 8: Kết nối đến router R1*

- Ở hình 8: kết nối router R1 lần lượt các bước như sau:

+ Từ thư viện netmiko đã được cài sẵn trên máy (cách cài thư viện nêu rõ ở phần I) ta import phương thức ConnectHandler

+ Tạo biến R1 với kiểu dữ liệu là dictionary bao gồm các key là ‘device\_type’, ‘ip’,

‘username’, ‘password’, ‘secret’ và các value tương ứng như trên hình 8

+ Tạo biến connect = ConnectHandler(\*\*R1) để kết nối đến router R1

A screen shot of a computer

Description automatically generated with low confidence

*Hình 9: Một số lệnh thông dụng của Netmiko*

### b/ Cấu hình địa chỉ IP lên các interface của router R1

A screenshot of a computer program

Description automatically generated with medium confidence

*Hình 10: Cấu hình trên các interface khác của R1*

- Ở hình 10:

+ Tạo biến int lưu lại các lệnh để cấu hình interface e0/1

+ Dùng lệnh send\_config\_set(int) để gửi lệnh từ int ở mode config

+ Đây chỉ có thể cấu hình cho một interface, muốn cấu hình cho nhiều interface ta sẽ sử dụng cách dưới đây

A screen shot of a computer

Description automatically generated with medium confidence

*Hình 11: Cấu hình đồng thời nhiều interface*

- Ở hình 11: ta sử dụng vòng lặp for để cấu hình đồng thời nhiều interface

### c/ Tìm số các interface bị shutdown của router R1

A screenshot of a computer

Description automatically generated

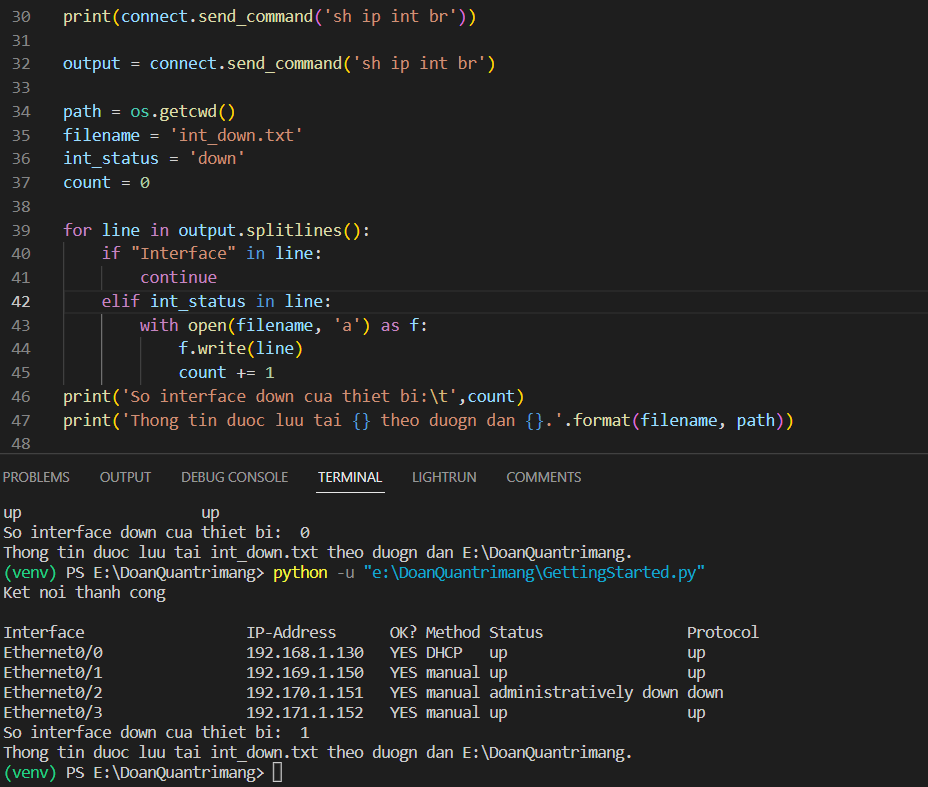
*Hình 12: Shutdown interface*

- Ở hình 12 ta thực hiện shutdown interface e0/2 trên terminal của router R1



*Hình 13: Import thêm phương thức os*

- Mục đích khi import thêm phương thức os: lấy đường dẫn hiện tại mà mình muốn lưu



*Hình 14: Kiểm tra các interface bị shutdown*

- Ở hình 14:

+ Tạo biến output sử dụng phương thức send\_command(‘sh ipi int br’) để cho ra các thông tin của interface router R1

+ Tạo biến path để lưu đường dẫn muốn lưu file ‘int\_down.txt’

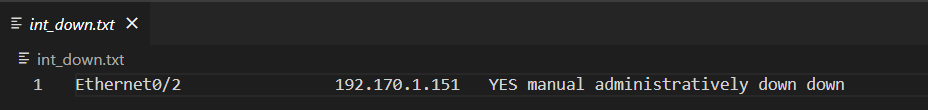
+ Tạo biến filename để lưu thông tin interface bị shutdown

+ Tạo biến int\_status = down là trạng thái của interface

+ Tạo biến count = 0 để đếm số interface bị shutdown

+ Sử dụng vòng lặp for, biến line được tạo ra để kiểm tra tuần tự dữ liệu xuất ra được gán ở biến output chia nhỏ ra: nếu gặp int\_status thoả mãn, dữ liệu của interface đó được ghi vào biến filename và count tăng lên 1 đơn vị

+ Xuất ra số interface bị shutdown của thiết bị và thông tin file được ghi vào thư mục trỏ đến trên bộ nhớ



*Hình 15: Thông tin xuất ra file int\_down.txt*

## 3/ Tự động hoá với Netmiko (Automation)

### a/ Gửi lệnh hiển thị đến nhiều thiết bị

A picture containing text, screenshot, font, software

Description automatically generated

*Hình 16: Cấu hình cho các thiết bị*

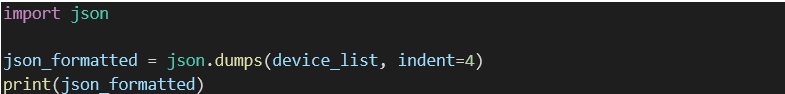
- Ở hình 16:

+ Tạo 1 mảng my\_device chứa các địa chỉ IP

+ Tạo 1 danh sách device\_list

+ Chạy vòng lặp for để thực hiện cấu hình các thiết bị và sau đó gán vào list device\_list

+ Xuất ra device\_list



*Hình 17: Import phương thức json*

- Ở hình 17: thực hiện việc thêm phương thức json để format lại device\_list với mục đích thẩm mỹ, nhìn đỡ rối rắm hơn, sau đó xuất ra json\_formatted

A screenshot of a computer screen

Description automatically generated with medium confidence

*Hình 18: Thông tin thiết bị sau khi được format phương thức json*



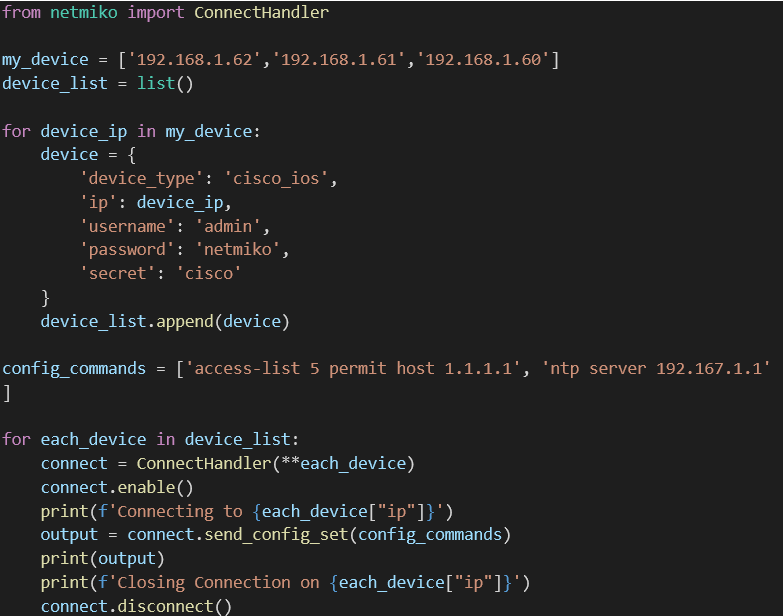
*Hình 19: Kiểm tra kết nối thiết bị*

- Ở hình 19:

+ Kế thừa kết quả từ hình 16, sử dụng thêm một vòng lặp for để hiển thị thông tin kết nối của các thiết bị

+ Ngắt kết nối bằng phương thức disconnect() sau khi đã sử dụng xong thiết bị thì mới được khởi tạo kết nối mới cho thiết bị khác trong vòng lặp

### b/ Gửi lệnh ở mode config đến nhiều thiết bị



*Hình 20: Thực hiện vào trạng thái config trên nhiều thiết bị*

- Ở hình 20: kế thừa kết quả từ hình 16 và tiếp tục các bước như sau:

+ Tạo biến config\_command để lưu lại lệnh muốn thực thi ở mode config

+ Tương tự như việc kết nối nhiều thiết bị, ta cũng sẽ dung vòng for để đi qua từng thiết bị, sau khi đã kết nối, ta sẽ dùng lệnh send\_config\_set để gửi lệnh đến thiết bị ở mode config và lưu quá trình đó ở biến output và in nó ra để kiểm tra

+ Ngắt kết nối sau khi thực hiện xong ở 1 thiết bị để tạo kết nối mới ở thiết bị khác trong vòng lặp

A computer screen shot of a computer

Description automatically generated with low confidence

*Hình 21: Terminal sau khi thực thi*

## 4/ TextFSM

### a/ TextFSM là gì?

- Khi bạn dùng lệnh CLI trên thiết bị mạng thì output nó hiện ra là một chuỗi semi-constructed data

- TextSFM là một module của Python cho phép phân tích cú pháp dữ liệu văn bản bán cấu trúc bằng Templates(Mẫu dữ liệu). Ban đầu, nó được phát triển để giúp kỹ sư mạng tự động hóa quá trình phân tích và trích xuất thông tin từ đầu ra qua các lệnh CLI của các thiết bị mạng.

- TextFSM nhận hai đầu vào - một tệp mẫu và một tệp văn bản (như đầu ra CLI) và trả về một danh sách các bản ghi chứa dữ liệu được phân tích từ văn bản đó

### b/ Ứng dụng của TextFSM

A screen shot of a computer

Description automatically generated with medium confidence

*Hình 22: Đoạn code CHƯA dùng bất kỳ biện pháp phân tích cú pháp*

A picture containing text, screenshot, font

Description automatically generated

*Hình 23: Kết quả của đoạn code CHƯA dùng bất kỳ biện pháp phân tích cú pháp*

A screen shot of a computer

Description automatically generated with medium confidence

*Hình 24: Sử dụng thêm biện pháp phân tích cú pháp*

- Ở hình 24: đã sử dụng thêm thư viện json và thêm use\_textfsm khi gửi lệnh ‘sh ip int br’

A screen shot of a computer

Description automatically generated with medium confidence

*Hình 25: Kết quả sau khi phân tích cú pháp*

- Bây giờ ta sẽ thử in ra các interface đang ở trạng thái administratively down và in ra

A screen shot of a computer

Description automatically generated with medium confidence

*Hình 26: In các interface ở trạng thái administrative down*

- Ở hình 26: Đoạn code trên đã thêm biến not\_connect\_interfaces lưu những interface đang ở trạng thái administratively down và in ra

A picture containing text, screenshot, font, line

Description automatically generated

*Hình 27: Kết quả in ra*

- Cấu hình thiết bị bằng key của TextFSM

A screen shot of a computer program

Description automatically generated with low confidence

*Hình 28: Cấu hình thiết bị bằng key TextFSM*

+ Ta dung vòng for để gửi lệnh đến từng thiết bị và thay vì nhập tay các interface, ta có thể

dung value của Key interface đã lấy được ở trên để gửi một cách tự động.

+ Biến config\_output sẽ lưu lại các lệnh đã thực hiện ở mode config của interface đó

A screen shot of a computer program

Description automatically generated with medium confidence

*Hình 29: Kết quả sau khi chạy*

A screenshot of a computer

Description automatically generated with medium confidence

*Hình 30: Sử dụng lệnh ‘sh ip int bri’ để kiểm tra kết quả*

- Tóm lại: ứng dụng của TextFSM là:

+ Phân tích cú pháp, định dạng lại thông tin được xuất ra

+ Sử dụng key TextFSM để cấu hình thiết bị

+ Hiển thị trạng thái của inteface

**\* Tài liệu tham khảo:**

**-** [**https://www.packetswitch.co.uk/netmiko-intro/**](https://www.packetswitch.co.uk/netmiko-intro/)

**-** [**https://www.packetswitch.co.uk/netmiko-par2/**](https://www.packetswitch.co.uk/netmiko-par2/)

**-** [**https://www.packetswitch.co.uk/netmiko-and-textfsm-example/**](https://www.packetswitch.co.uk/netmiko-and-textfsm-example/)

**-** [**https://www.packetswitch.co.uk/netmiko-intro/**](https://www.packetswitch.co.uk/netmiko-intro/)

**------ HẾT -----**