**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**

A logo with arrows in a white background

Description automatically generated

**PBL4: DỰ ÁN HỆ ĐIỀU HÀNH & MẠNG MÁY TÍNH**

**Đề tài: Xây dựng tiện ích quét IP để phục vụ quản trị mạng**

**SINH VIÊN THỰC HIỆN:**

**Đặng Văn Cường LỚP: 22T\_DT1 NHÓM: 22Nh13A**

**Siphanthong Xanakone LỚP: 22T\_DT5 NHÓM: 22Nh13A**

**GIẢNG VIÊN HƯỚNG DẪN: Th.S Nguyễn Văn Nguyên**

**Đà Nẵng 10/ 2024**

MỤC LỤC

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 6](#_Toc182395829)

[I. Giới thiệu về giao thức ARP và cách thức hoạt động 6](#_Toc182395830)

[1. Khái niệm ARP 6](#_Toc182395831)

[2. Cách thức hoạt động của ARP 6](#_Toc182395832)

[3. Cấu trúc gói tin ARP 7](#_Toc182395833)

[4. Vai trò của ARP trong mạng LAN 7](#_Toc182395834)

[II. Phương pháp quét IP bằng ARP và các ứng dụng 8](#_Toc182395835)

[1. Quét ARP là gì? 8](#_Toc182395836)

[2. Quy trình quét ARP 8](#_Toc182395837)

[3. Ưu điểm và hạn chế của quét ARP 8](#_Toc182395838)

[CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 9](#_Toc182395839)

[I. Phân tích yêu cầu 9](#_Toc182395840)

[1. Yêu cầu chức năng 9](#_Toc182395841)

[2. Yêu cầu phi chức năng 9](#_Toc182395842)

[II. Thiết kế hệ thống 9](#_Toc182395843)

[1. Các module chính 9](#_Toc182395844)

[CHƯƠNG 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ 10](#_Toc182395845)

[I. Triển khai 10](#_Toc182395846)

[1. Các công cụ triển khai 10](#_Toc182395847)

[2. Triển khai chương trình 10](#_Toc182395848)

[II. Kết quả 16](#_Toc182395849)

[1. Kết quả sau khi chạy chương trình 16](#_Toc182395850)

[2. Kết quả sau khi quét một dải IP 17](#_Toc182395851)

[3. Kết quả của phần quản lý RAM&CPU 17](#_Toc182395852)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 18](#_Toc182395853)

[I. Đạt được 18](#_Toc182395854)

[II. Chưa đạt được 18](#_Toc182395855)

[III. Hướng phát triển 18](#_Toc182395856)

[PHỤ LỤC 20](#_Toc182395857)

DANH SÁCH HÌNH VẼ

[Hình 1: Mô tả cách giao thức ARP hoạt động 6](#_Toc182395858)

[Hình 2: Mô tả các trường của một gói tin ARP 7](#_Toc182395859)

[Hình 3: Minh họa sự hoạt động của các luồng 14](#_Toc182395860)

[Hình 4: Kết quả sau khi chạy chương trình 16](#_Toc182395861)

[Hình 5: Kết quả sau khi quét một dải IP 17](#_Toc182395862)

[Hình 6: Kết quả của phần quản lý RAM & CPU 17](#_Toc182395863)

DANH SÁCH CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| Thuật ngữ | Định nghĩa |
| IP | Internet Protocol |
| ARP | Address Resolution Protocol |
| MAC | Media Access Control |
| LAN | Local Area Network |
| ICMP | Internet Control Message Protocol |
| TCP | Transmission Control Protocol |

GIỚI THIỆU ĐỀ TÀI

Đề tài "Xây dựng tiện ích phục vụ quản trị mạng," tập trung vào việc phát triển một công cụ hỗ trợ cho quản trị và giám sát hệ thống mạng. Tiện ích này có các chức năng chính như quét địa chỉ IP, ping, giám sát CPU và giám sát RAM.

1. **Mục đích:**

* Xây dựng chức năng **quét địa chỉ IP** để xác định các thiết bị đang kết nối trong mạng.
* Tạo công cụ **ping** để giám sát trạng thái kết nối và thời gian phản hồi của các thiết bị.
* Phát triển **giám sát CPU** để hiển thị mức sử dụng CPU theo thời gian, giúp quản trị viên nắm bắt tình trạng tải hệ thống.
* Phát triển **giám sát RAM** để hiển thị mức sử dụng RAM theo thời gian, từ đó giúp tối ưu hoá và đảm bảo hiệu suất ổn định.

1. **Phạm vi:**

* **Quét IP và Ping**: Chỉ áp dụng cho các thiết bị trong cùng một mạng nội bộ hoặc các thiết bị kết nối qua các phương pháp cho phép.
* **Giám sát CPU và RAM**: Được thực hiện trên hệ điều hành Windows hoặc Linux và chỉ giới hạn ở các hệ thống đơn giản, không hỗ trợ các hệ thống phức tạp hoặc cơ sở hạ tầng lớn.

# CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

## Giới thiệu về giao thức ARP và cách thức hoạt động

### Khái niệm ARP

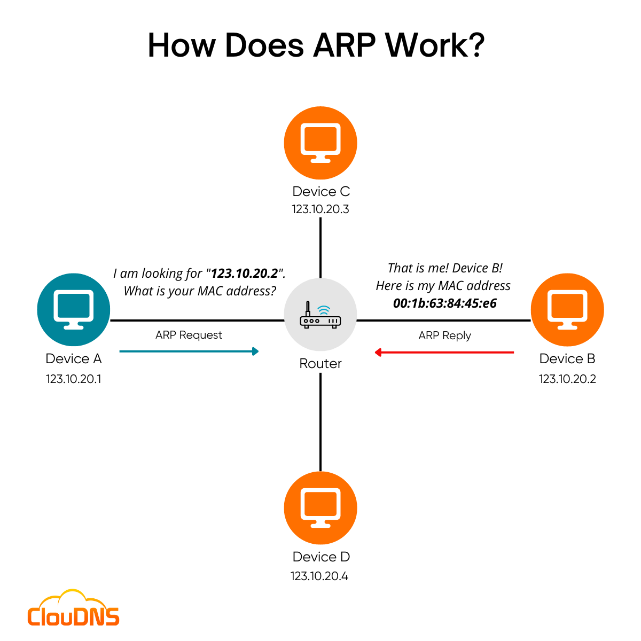
Address Resolution Protocol (ARP) là một giao thức được sử dụng trong mạng máy tính để ánh xạ địa chỉ IP (Internet Protocol) thành địa chỉ MAC (Media Access Control). ARP hoạt động ở lớp liên kết dữ liệu trong mô hình OSI và chủ yếu được sử dụng trong mạng LAN (Local Area Network). Mỗi thiết bị trong mạng LAN đều có địa chỉ IP và MAC riêng, và giao thức ARP giúp ánh xạ giữa hai loại địa chỉ này để các thiết bị có thể truyền thông tin với nhau.

Khi một thiết bị cần gửi gói tin đến một thiết bị khác trong mạng LAN, nó cần biết địa chỉ MAC của thiết bị đích. Tuy nhiên, các gói tin chỉ chứa địa chỉ IP. ARP sẽ giải quyết vấn đề này bằng cách tìm địa chỉ MAC tương ứng với địa chỉ IP đích.

### Cách thức hoạt động của ARP

Quá trình hoạt động của ARP bao gồm hai bước chính:

* **ARP Request**: Khi một thiết bị muốn biết địa chỉ MAC của một thiết bị khác trong mạng, nó sẽ gửi một gói tin ARP Request tới tất cả các thiết bị trong mạng (broadcast). Gói tin ARP Request chứa địa chỉ IP của thiết bị đích và yêu cầu bất kỳ thiết bị nào có địa chỉ IP đó phản hồi với địa chỉ MAC của nó.
* **ARP Reply**: Thiết bị đích khi nhận được gói tin ARP Request sẽ gửi lại một gói tin ARP reply chứa địa chỉ MAC của nó cho thiết bị yêu cầu. Thông tin này sau đó sẽ được lưu trữ trong ARP cache của thiết bị gửi yêu cầu để sử dụng trong các lần truyền dữ liệu tiếp theo.

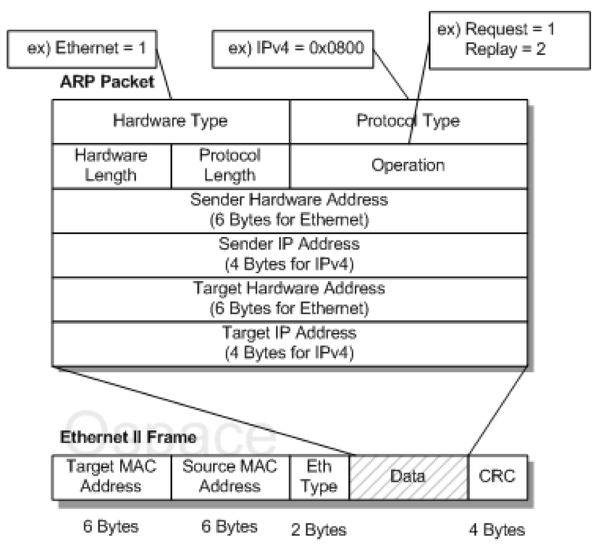


Hình 1: Mô tả cách giao thức ARP hoạt động

### Cấu trúc gói tin ARP

Gói tin ARP bao gồm các trường sau:

* **Hardware Type**: Xác định loại phần cứng, thường là Ethernet (1).
* **Protocol Type**: Loại giao thức, thường là IPv4 (0x0800).
* **Hardware Address Length**: Độ dài của địa chỉ MAC (6 byte).
* **Protocol Address Length**: Độ dài của địa chỉ IP (4 byte).
* **Operation Code**: Phân biệt giữa ARP Request (1) và ARP Response(2).
* **Sender MAC Address**: Địa chỉ MAC của thiết bị gửi.
* **Sender IP Address**: Địa chỉ IP của thiết bị gửi.
* **Target MAC Address**: Địa chỉ MAC của thiết bị nhận (để trống trong ARP Request).
* **Target IP Address**: Địa chỉ IP của thiết bị nhận.



Hình 2: Mô tả các trường của một gói tin ARP

### Vai trò của ARP trong mạng LAN

Trong mạng LAN, các thiết bị chỉ có thể giao tiếp với nhau nếu biết địa chỉ MAC của thiết bị đích. ARP giúp thiết bị xác định địa chỉ MAC dựa trên địa chỉ IP, giúp quá trình giao tiếp trong mạng được diễn ra suôn sẻ. Không có ARP, việc truyền dữ liệu trong mạng LAN sẽ không thể thực hiện được, vì thiết bị gửi không thể biết được địa chỉ phần cứng của thiết bị nhận để định tuyến gói tin.

## Phương pháp quét IP bằng ARP và các ứng dụng

### Quét ARP là gì?

Quét ARP (ARP Scan) là một kỹ thuật dùng để xác định các thiết bị đang hoạt động trong mạng LAN bằng cách gửi các gói tin ARP Request tới các địa chỉ IP trong phạm vi mạng và nhận phản hồi từ các thiết bị có địa chỉ MAC tương ứng. Kỹ thuật này chủ yếu được sử dụng trong các mạng nội bộ (LAN) vì ARP không hoạt động qua các router, do đó nó chỉ có thể quét được các thiết bị trong cùng một phân đoạn mạng.

### Quy trình quét ARP

Quy trình quét ARP bao gồm các bước sau:

1. **Gửi ARP Request**: Chương trình quét sẽ gửi hàng loạt gói tin ARP Request đến các địa chỉ IP trong phạm vi mạng được chỉ định. Mỗi gói tin sẽ chứa một địa chỉ IP đích cụ thể trong mạng LAN.
2. **Nhận ARP Reply**: Các thiết bị có địa chỉ IP khớp với địa chỉ được yêu cầu sẽ phản hồi bằng gói tin ARP Response, chứa địa chỉ MAC của chúng.
3. **Lưu trữ kết quả**: Dựa trên các phản hồi nhận được, chương trình sẽ liệt kê các địa chỉ IP đang hoạt động cùng với địa chỉ MAC tương ứng. Các thiết bị không trả lời ARP Request có thể được coi là không hoạt động hoặc không có trong mạng.

### Ưu điểm và hạn chế của quét ARP

* **Ưu điểm**:
  + **Tốc độ nhanh**: Quét ARP chỉ cần thực hiện trong phạm vi mạng LAN nên thường có tốc độ nhanh hơn so với các phương pháp khác như ICMP hay TCP Scan.
  + **Đáng tin cậy trong mạng LAN**: ARP là một giao thức cơ bản được sử dụng trong tất cả các mạng LAN, do đó kỹ thuật này đảm bảo được độ chính xác trong việc xác định thiết bị.
* **Hạn chế**:
  + **Giới hạn phạm vi**: ARP chỉ hoạt động trong mạng LAN và không thể sử dụng để quét các mạng bên ngoài.
  + **Phụ thuộc vào cấu hình mạng**: Trong một số trường hợp, mạng có thể được cấu hình để chặn các yêu cầu ARP hoặc giới hạn thời gian phản hồi, gây khó khăn cho việc quét.

# CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## Phân tích yêu cầu

### Yêu cầu chức năng

* Quét được các IP thuộc về thiết bị đang hoạt động trong mạng LAN hoặc một địa chỉ IP cụ thể
* Hiển thị danh sách các IP gồm các thông tin như: Địa chỉ IP, địa chỉ MAC, tên thiết bị, thời gian ping trung bình
* Lưu trữ kết quả quét được vào tệp .xlsx
* Hiển thị mức độ sử dụng RAM và CPU của máy chủ (máy quét)

### Yêu cầu phi chức năng

* Tối ưu thời gian quét và thời gian hiển thị bằng lập trình đa luồng, đa tuyến
* Giao diện thân thiện, dễ sử dụng
* Dải IP đầu vào linh hoạt, nhiều biến thể
* Chuẩn hóa dải IP đầu vào

## Thiết kế hệ thống

### Các module chính

Hệ thống chia thành 3 module chính: module quét IP, module quản lý RAM&CPU, module giao diện

**Module quét IP**: Sử dụng thư viện Scapy để thực hiện quá trình quét mạng thông qua giao thức ARP. Thành phần này có nhiệm vụ gửi gói tin ARP Request đến các địa chỉ IP trong dải mạng cần quét và nhận lại phản hồi ARP Reply từ các thiết bị hoạt động trong mạng.

**Module quản lý RAM&CPU**: Sử dụng thư viện Psutil để thực hiện quá trình lấy thông tin phần trăm CPU&RAM đang sử dụng

**Module giao diện người dùng**: Được xây dựng với thư viện Tkinter và CustomTkinter để cung cấp một giao diện đồ họa đơn giản cho người dùng. Phần này giúp hiển thị kết quả quét IP trong một bảng dữ liệu trực quan, cho phép người dùng dễ dàng theo dõi các thiết bị đang kết nối trong mạng LAN. Vẽ đồ thị biểu diễn sự thay đổi của mức sử dụng RAM&CPU theo phần trăm

# CHƯƠNG 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

## Triển khai

### Các công cụ triển khai

* + Ngôn ngữ lập trình: Python
  + Thư viện hỗ trợ chính
    - Scapy: Giúp tạo, đóng gói các gói tin ARP và gửi, nhận request
    - Psutil: Lấy các thông số CPU&RAM của hệ điều hành
    - Tkinter & CustomTkinter: Để tạo giao diện đồ họa cho người dùng (GUI), hỗ trợ nhập dữ liệu và hiển thị kết quả.

### Triển khai chương trình

1. **Cài đặt các thư viện cần thiết: Scapy, Tkinter, …**

pip install scapy, tkinder, …

1. **Import các thư viện vào chương trình**

import customtkinter as ctk

import tkinter as tk

from tkinter import ttk, messagebox

from tkinter import font as tkFont

from scapy.all import ARP, Ether, srp

import ipaddress

import threading

from queue import Queue

import time

import socket

import struct

import platform

import subprocess

from openpyxl import Workbook

import psutil

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

import matplotlib.pyplot as plt

1. **Thực hiện**
2. **Thuật toán quét một ip đơn lẻ**

**Mã:**

def scan\_single\_ip(ip):

if stop\_flag.is\_set():

return []

arp = ARP(pdst=ip)

ether = Ether(dst="ff:ff:ff:ff:ff:ff")

packet = ether/arp

result = srp(packet, timeout=1, verbose=0)[0]

devices = [{'ip': received.psrc, 'mac': received.hwsrc} for sent, received in result]

return devices

**Đầu vào:** Một địa chỉ IP

**Đầu ra:** Thông tin (địa chỉ ip, địa chỉ MAC) của thiết bị có IP đó

**Xử lý:**

* Trước hết, cần phải kiểm tra đã có tín hiệu dừng quét không, nếu có thì không thực hiện hàm này
* arp = ARP(pdst=ip)
  + Dòng này tạo ra một gói tin ARP
  + ARP() là một lớp của Scapy để tạo gói tin ARP
  + Tham số pdst=ip chỉ định địa chỉ IP đích mà bạn muốn gửi gói tin ARP tới. Đây là địa chỉ IP mà bạn muốn kiểm tra xem nó có đang hoạt động trong mạng hay không
* ether = Ether(dst="ff:ff:ff:ff:ff:ff")
  + Tạo ra một gói tin Ethernet.
  + Ether() là lớp của Scapy để tạo gói tin Ethernet.
  + dst="ff:ff:ff:ff:ff:ff" là địa chỉ MAC broadcast, có nghĩa là gói tin này sẽ được gửi tới tất cả các thiết bị trong mạng cục bộ. Trong ARP, khi cần tìm thông tin về một địa chỉ IP, gói tin sẽ được gửi tới tất cả các thiết bị để tìm ra thiết bị nào sở hữu địa chỉ IP đó.
* packet = ether/arp
  + Dòng này kết hợp (xâu chuỗi) gói tin Ethernet với gói tin ARP để tạo ra một gói tin hoàn chỉnh sẽ được gửi đi.
  + Dấu / trong Scapy được dùng để nối các lớp giao thức khác nhau (ở đây là Ethernet và ARP) thành một gói tin duy nhất
* result = srp(packet, timeout=1, verbose=0)[0]
  + srp() trả về một danh sách các cặp (sent\_packet, received\_packet), trong đó mỗi cặp tương ứng với một gói tin đã được gửi và gói tin phản hồi nhận được. Ở đây, [0] lấy kết quả đầu tiên trong danh sách (là các gói tin đã nhận).
* devices = [{'ip': received.psrc, 'mac': received.hwsrc} for sent, received in result]
  + Dòng này tạo ra một danh sách các thiết bị được tìm thấy trong mạng.
  + Kết quả là một danh sách các từ điển, mỗi từ điển chứa thông tin về một thiết bị trong mạng.

1. **Thuật toán quét toàn bộ IP của mạng**

**Mã:**

def scan\_network(ip\_list, update\_progress):

devices = []

try:

queue = Queue()

for ip in ip\_list:

queue.put(ip)

def worker():

while not queue.empty() and not stop\_flag.is\_set():

ip = queue.get()

if stop\_flag.is\_set():

break

devices.extend(scan\_single\_ip(ip))

queue.task\_done()

update\_progress() # Update progress after scanning each IP

num\_threads = 20

threads = []

for \_ in range(num\_threads):

thread = threading.Thread(target=worker)

thread.start()

threads.append(thread)

for thread in threads:

thread.join()

except Exception as e:

print(f"Failed to scan network: {e}")

return devices

**Đầu vào:** Một dải địa chỉ IP

**Đầu ra:** Các thiết bị đang hoạt động trong mạng

**Xử lý:**

* Để đáp ứng yêu cầu phi chức năng là tối ưu về mặt thời gian nên chương trình này áp dụng lập trình đa luồng để quét các địa chỉ ip có trong mạng nhanh hơn
* Vì các luồng phải làm việc với một dải IP nên để tránh sự xung đột ta phải dùng một hàng đợi để lưu trữ các IP, các IP này sẽ được lấy ra tuần tự theo quy tắc (First In First Out) nhưng sẽ được các luồng quét song song với nhau, luồng nào xong thì có thể lấy IP ở đầu hàng đợi để quét tiếp
* dùng hàm join() của thread để chờ cho tất cả các luồng hoàn công việc rồi tiếp tục

A diagram of a process

Description automatically generated

Hình 3: Minh họa sự hoạt động của các luồng

1. **Thuật toán giám sát CPU và RAM**

**Mã:**

def update\_cpu\_ram\_plot():

*for* widget *in* monitor\_tab.winfo\_children():

        widget.destroy()

    cpu\_data = []

    ram\_data = []

    fig, ax = plt.subplots(2, 1, *figsize*=(5, 3))

    fig.suptitle('CPU and RAM Usage')

    canvas = FigureCanvasTkAgg(fig, *master*=monitor\_tab)

    canvas.get\_tk\_widget().pack(*side*=tk.TOP, *fill*=tk.BOTH, *expand*=1)

    def animate():

*while* True:

            cpu\_data.append(psutil.cpu\_percent())

            ram\_data.append(psutil.virtual\_memory().percent)

*if* len(cpu\_data) > 60:

                cpu\_data.pop(0)

                ram\_data.pop(0)

            ax[0].clear()

            ax[1].clear()

            ax[0].plot(cpu\_data, *label*=f'CPU %: {cpu\_data[-1]:.1f}%' *if* cpu\_data *else* 'CPU %')

            ax[0].set\_ylim(0, 100)

            ax[0].set\_ylabel("CPU Usage (%)")

            ax[0].legend()

            ax[1].plot(ram\_data, *label*=f'RAM %: {ram\_data[-1]:.1f}%', *color*='green')

            ax[1].set\_ylabel("RAM Usage (%)")

            ax[1].set\_ylim(0, 100)

            ax[1].legend()

            canvas.draw()

            time.sleep(1)

    threading.Thread(*target*=animate, *daemon*=True).start()

**Đầu vào:** Không có đầu vào cụ thể; thuật toán này sẽ giám sát CPU và RAM tự động.

**Đầu ra:** Biểu đồ hiển thị mức sử dụng CPU và RAM theo thời gian.

**Xử lý:**

* Dùng thư viện *psutil* để lấy thông tin phần trăm sử dụng của CPU và RAM trong mỗi giây
* Dùng thư viện *matplotlib* để tạo đồ thị với trục x là trục thời gian và trục y là mức phần trăm của CPU và RAM, giới hạn của khung thời gian là từ 1 đến 60 giây

## Kết quả

### Kết quả sau khi chạy chương trình

**A screenshot of a computer

Description automatically generated**

Hình 4: Kết quả sau khi chạy chương trình

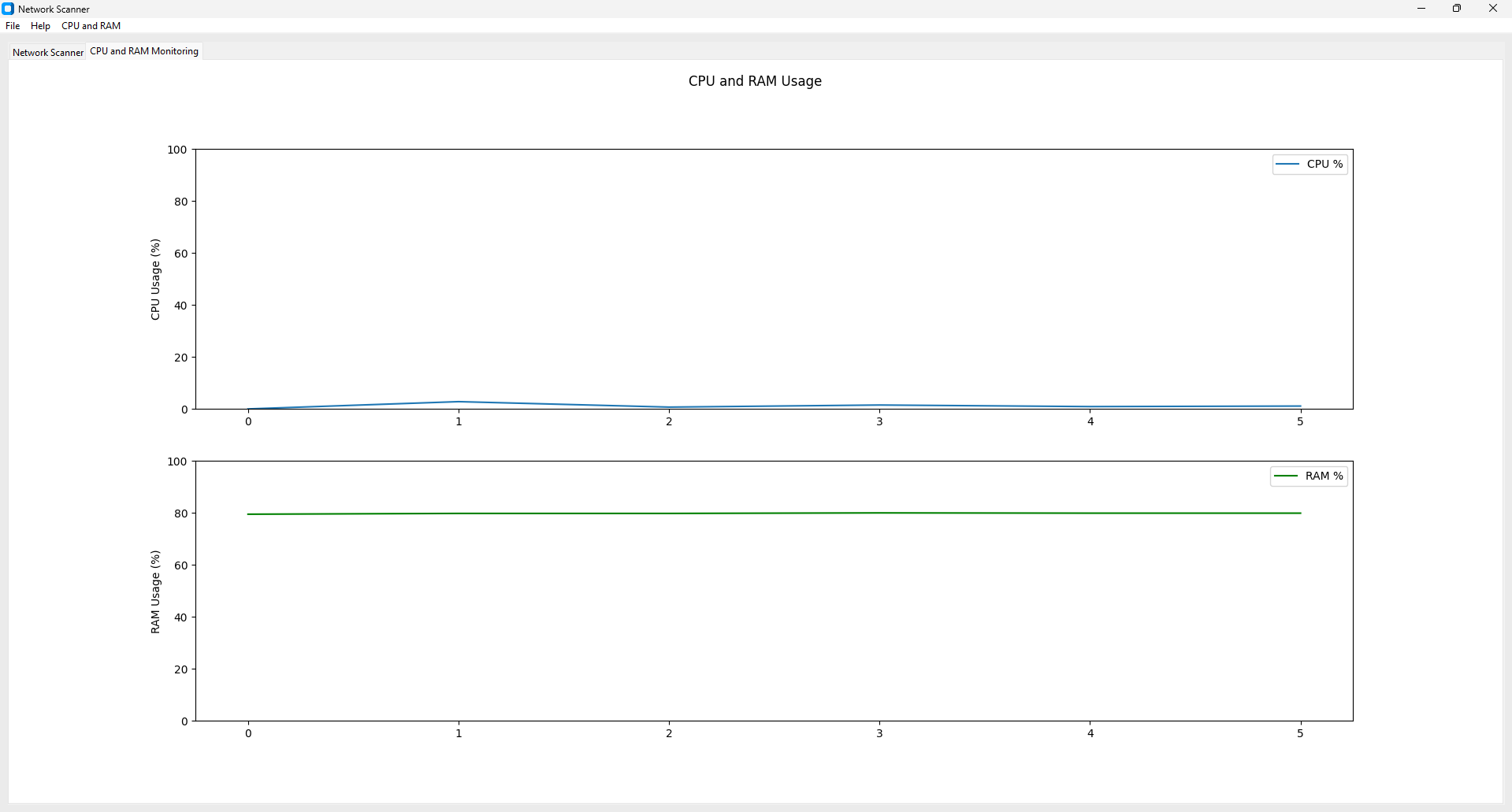
### Kết quả sau khi quét một dải IP

**A screenshot of a computer

Description automatically generated**

Hình 5: Kết quả sau khi quét một dải IP

### Kết quả của phần quản lý RAM&CPU

****

Hình 6: Kết quả của phần quản lý RAM & CPU

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Đạt được

* Trong PBL4 , nhóm chúng em đã trình bày một ứng dụng quét IP sử dụng giao thức ARP (Address Resolution Protocol) để xác định và quản lý các thiết bị trên mạng LAN. Ứng dụng không chỉ giúp người dùng dễ dàng xác định các địa chỉ IP và MAC của các thiết bị kết nối mà còn cung cấp khả năng theo dõi hiệu suất của CPU và RAM của hệ thống thông qua biểu đồ trực quan.
* Việc kết hợp giữa chức năng quét IP và quản lý tài nguyên hệ thống đã tạo ra một công cụ mạnh mẽ cho việc giám sát mạng, hỗ trợ quản trị viên mạng trong việc duy trì và tối ưu hóa hiệu suất hoạt động của hệ thống.

## Chưa đạt được

* Hệ thống vẫn còn một số vấn đề cần giải quyết là hệ thống vẫn chưa thể quét được các tiến trình đang chạy trên các thiết bị đang truy cập vào mạng, cung cấp các thông tin về thiết bị chưa chi tiết, chưa sử dụng các thuật toán để dự đoán và ngăn chặn tình trạng quá tải tài nguyên

## Hướng phát triển

**Mở rộng tính năng quét IP:**

* Tích hợp khả năng quét các địa chỉ IP trong các dải địa chỉ lớn hơn, hỗ trợ quét đa subnet.
* Cung cấp thông tin chi tiết hơn về các thiết bị, bao gồm tên thiết bị, nhà sản xuất, và thông tin trạng thái.

**Nâng cao giao diện người dùng**:

* Cải thiện giao diện người dùng với các yếu tố trực quan hơn và dễ sử dụng, có thể sử dụng thư viện như Tkinter hoặc PyQt.
* Thêm tính năng thông báo và cảnh báo khi phát hiện các thiết bị không mong muốn trên mạng.

**Tối ưu hóa quản lý tài nguyên**:

* Phát triển các phương pháp tối ưu hóa hơn cho việc quản lý RAM và CPU, có thể sử dụng các thuật toán để dự đoán và ngăn chặn tình trạng quá tải tài nguyên.
* Cung cấp khả năng lưu trữ lịch sử sử dụng CPU và RAM để người dùng có thể theo dõi và phân tích hiệu suất theo thời gian.

TÀI LIỆU THAM KHẢO

[1] Kurose, J., & Ross, K. (2017). *Computer Networking: A Top-Down Approach* (8th ed.). Pearson.

[2] Arpaci-Dusseau, R. H., & Arpaci-Dusseau, A. C. (2018). *Operating Systems: Three Easy Pieces*. Arpaci-Dusseau Books.

[3] Bovet, D. P., & Cesati, M. (2005). *Understanding the Linux Kernel* (3rd ed.). O'Reilly Media.

[4] Stevens, W. R., & Rago, S. A. (2013). *Advanced Programming in the UNIX Environment* (3rd ed.). Addison-Wesley.

[5] Lutz, M. (2013). *Learning Python* (5th ed.). O'Reilly Media.

# PHỤ LỤC

|  |
| --- |
| import customtkinter as ctk  import tkinter as tk  from tkinter import ttk, messagebox  from tkinter import font as tkFont  from scapy.all import ARP, Ether, srp  import ipaddress  import threading  from queue import Queue  import time  import socket  import struct  import platform  import subprocess  from openpyxl import Workbook  import psutil  from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  import matplotlib.pyplot as plt  ctk.set\_appearance\_mode("Light")  ctk.set\_default\_color\_theme("green")  stop\_flag = threading.Event() # Global stop flag  def get\_local\_ip\_range():  try:  hostname = socket.gethostname()  local\_ip = socket.gethostbyname(hostname)  subnet\_mask = socket.inet\_ntoa(struct.pack('!L', (1 << 32) - (1 << (32 - 24))))  ip\_parts = local\_ip.split('.')  subnet\_parts = subnet\_mask.split('.')  network\_parts = [str(int(ip\_parts[i]) & int(subnet\_parts[i])) for i in range(4)]  network\_address = '.'.join(network\_parts)  network = ipaddress.ip\_network(f"{network\_address}/24", strict=False)  ip\_range = f"{network.network\_address}-{network.network\_address + (network.num\_addresses - 1)}"  return str(ip\_range)  except Exception as e:  print(f"Failed to get local IP range: {e}")  return ""  def expand\_ip\_range(ip\_range):  ip\_list = []  try:  for ip\_block in map(str.strip, ip\_range.split(',')):  if '-' in ip\_block:  start\_ip, end\_ip = ip\_block.split('-')  if '.' not in end\_ip:  base\_ip = start\_ip.rsplit('.', 1)[0]  end\_ip = f"{base\_ip}.{end\_ip}"  start\_ip\_int = int(ipaddress.IPv4Address(start\_ip))  end\_ip\_int = int(ipaddress.IPv4Address(end\_ip))  ip\_list.extend(str(ipaddress.IPv4Address(ip\_int)) for ip\_int in range(start\_ip\_int, end\_ip\_int + 1))  else:  ip\_list.append(ip\_block)  if len(ip\_list) == 0: # Fix condition  messagebox.showerror(title="Error", message="Please enter a valid address!")  return ip\_list  except:  messagebox.showerror(title="Error", message="Please enter a valid address!")  def get\_device\_name(ip\_address):  try:  name = socket.getfqdn(ip\_address)  return name if name else "Unknown"  except Exception as e:  return f"Error: {e}"  def scan\_network(ip\_list, update\_progress):  devices = []  try:  queue = Queue()  for ip in ip\_list:  queue.put(ip)  def worker():  while not queue.empty() and not stop\_flag.is\_set():  ip = queue.get()  if stop\_flag.is\_set():  break  devices.extend(scan\_single\_ip(ip))  queue.task\_done()  update\_progress() # Update progress after scanning each IP  num\_threads = 20  threads = []  for \_ in range(num\_threads):  thread = threading.Thread(target=worker)  thread.start()  threads.append(thread)  for thread in threads:  thread.join()  except Exception as e:  print(f"Failed to scan network: {e}")  return devices  def scan\_single\_ip(ip):  if stop\_flag.is\_set(): # Early exit if stop flag is set  return []  arp = ARP(pdst=ip)  ether = Ether(dst="ff:ff:ff:ff:ff:ff")  packet = ether/arp  result = srp(packet, timeout=1, verbose=0)[0]  devices = [{'ip': received.psrc, 'mac': received.hwsrc} for sent, received in result]  return devices  def ping(ip, count=2):  param = "-n" if platform.system().lower() == "windows" else "-c"  command = ["ping", param, str(count), ip]  try:  ping\_times = []  for i in range(count):  if stop\_flag.is\_set(): # Early exit if stop flag is set during ping  return "Stopped"  start\_time = time.time()  response = subprocess.run(command, stdout=subprocess.PIPE, stderr=subprocess.PIPE)  end\_time = time.time()  if response.returncode == 0:  ping\_time = (end\_time - start\_time) \* 1000 # Convert to milliseconds  ping\_times.append(ping\_time)  if ping\_times:  avg\_ping = sum(ping\_times) / len(ping\_times)  return f"{avg\_ping:.2f} ms"  else:  return "Unreachable"  except Exception as e:  return f"Error: {str(e)}"  def start\_scan():  if scan\_button.cget('text') == "Scan Network":  stop\_flag.clear()  ip\_range = ip\_range\_entry.get()  ip\_list = expand\_ip\_range(ip\_range)  total\_ips = len(ip\_list)  if total\_ips == 0:  return  progress\_bar.set(0) # Reset progress bar to 0  scan\_button.configure(text="Stop Scan", command=stop\_scan)  def update\_progress():  nonlocal scanned\_ips  scanned\_ips += 1  progress\_value = scanned\_ips / total\_ips  root.after(0, progress\_bar.set, progress\_value) # Update progress bar  def scan\_thread():  devices = scan\_network(ip\_list, update\_progress)  total\_device = len(devices)  root.after(0, clear\_results)  progress\_bar.set(0)  for index, device in enumerate(devices):  if stop\_flag.is\_set():  break  device\_name = get\_device\_name(device['ip'])  ping\_time = ping(device['ip'], count=2)  nonlocal scanned\_device  scanned\_device += 1  progress\_value = scanned\_device / total\_device  root.after(0, progress\_bar.set, progress\_value)  root.after(0, lambda idx=index, dev=device, ping\_time=ping\_time, device\_name=device\_name:  tree.insert('', 'end', values=(idx + 1, device\_name, dev['mac'], dev['ip'], ping\_time)))  root.after(0, reset\_scan\_button)  scanned\_ips = 0 # Track the number of scanned IPs  scanned\_device = 0 # Track the number of scanned devices  t = threading.Thread(target=scan\_thread)  t.daemon = True  t.start()  else:  stop\_scan()  def on\_tree\_select(event):  selected\_item = tree.selection()  if selected\_item:  # Get selected row's values from the first treeview  item\_values = tree.item(selected\_item[0], 'values')    stv.item(0, text=f"Status: Active")  stv.item(1, text=f"Device Name: {item\_values[1]}")  stv.item(2, text=f"IP Address: {item\_values[3]}")  stv.item(3, text=f"MAC Address: {item\_values[2]}")  stv.item(4, text=f"Ping: {item\_values[4]}")  def update\_cpu\_ram\_plot():  for widget in monitor\_tab.winfo\_children():  widget.destroy()  cpu\_data = []  ram\_data = []  fig, ax = plt.subplots(2, 1, figsize=(5, 3))  fig.suptitle('CPU and RAM Usage')  canvas = FigureCanvasTkAgg(fig, master=monitor\_tab)  canvas.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)  def animate():  while True:  cpu\_data.append(psutil.cpu\_percent())  ram\_data.append(psutil.virtual\_memory().percent)  if len(cpu\_data) > 60:  cpu\_data.pop(0)  ram\_data.pop(0)  ax[0].clear()  ax[1].clear()  ax[0].plot(cpu\_data, label=f'CPU %: {cpu\_data[-1]:.1f}%' if cpu\_data else 'CPU %')  ax[0].set\_ylim(0, 100)  ax[0].set\_ylabel("CPU Usage (%)")  ax[0].legend()  ax[1].plot(ram\_data, label=f'RAM %: {ram\_data[-1]:.1f}%', color='green')  ax[1].set\_ylabel("RAM Usage (%)")  ax[1].set\_ylim(0, 100)  ax[1].legend()  canvas.draw()  time.sleep(1)  threading.Thread(target=animate, daemon=True).start()  def stop\_scan():  stop\_flag.set()  reset\_scan\_button()  def reset\_scan\_button():  scan\_button.configure(text="Scan Network", command=start\_scan)  # progress\_bar.configure(value=0)  def clear\_results():  for row in tree.get\_children():  tree.delete(row)  reset\_stv()  def reset\_stv():  stv.item(0, text=f"Status:")  stv.item(1, text=f"Device Name:")  stv.item(2, text=f"IP Address:")  stv.item(3, text=f"MAC Address:")  stv.item(4, text=f"Ping:")  def put\_into\_entry():  selected\_class = ip\_class.get()  ip\_range\_entry.delete(0, tk.END)  if selected\_class == "A":  ip\_range\_entry.insert(0, "1.0.0.0/8")  elif selected\_class == "B":  ip\_range\_entry.insert(0, "128.0.0.0/16")  elif selected\_class == "C":  ip\_range\_entry.insert(0, "192.0.0.0/24")  def export\_results\_excel():  if not tree.get\_children():  messagebox.showerror("Export Error", "No data available to export!")  return  workbook = Workbook()  sheet = workbook.active  sheet.append(["#", "Device Name", "MAC Address", "IP Address", "Ping Status"])  for row in tree.get\_children():  values = tree.item(row)['values']  sheet.append(values)  workbook.save("network\_scan\_results.xlsx")  messagebox.showinfo("Export Results", "Results exported successfully as 'network\_scan\_results.xlsx'.")  # Callback Functions  def on\_exit():  # root.quit() # Exits the application  stop\_flag.set() # Stop all background threads  root.quit()  def show\_about():  messagebox.showinfo("About", "Network Scanner v1.0\nCreated by Xanakone Siphanthong, Đặng Văn Cường")  # Create the main application window  appWidth, appHeight = 800, 700  root = ctk.CTk()  root.title("Network Scanner")  root.geometry(f"{appWidth}x{appHeight}")  root.minsize(appWidth, appHeight)  detected\_ip\_range = get\_local\_ip\_range()  # Create a style object  style = ttk.Style()  # Customize the tabs in the Notebook  style.configure("TNotebook.Tab", font=('Helvetica', 11, 'bold'), padding=[0, 0], background="lightblue", foreground="gray")  style.map("TNotebook.Tab", background=[("selected", "green")], foreground=[("selected", "black")])  # Create the tab control  tab\_control = ttk.Notebook(root, takefocus=False)  # Create tabs  network\_tab = ttk.Frame(tab\_control)  monitor\_tab = ttk.Frame(tab\_control)  # Add tabs to the tab control  tab\_control.add(network\_tab, text="Network Scanner")  tab\_control.add(monitor\_tab, text="System Monitor")  # Pack the tab control to fill the window  tab\_control.pack(expand=True, fill="both")  # Menu bar setup  menu\_bar = tk.Menu(root)  # File menu  file\_menu = tk.Menu(menu\_bar, tearoff=0, font=('Helveti', 11))  file\_menu.add\_command(label="Exit", command=on\_exit)  file\_menu.add\_command(label="Export", command=export\_results\_excel)  menu\_bar.add\_cascade(label="File", menu=file\_menu)  # Help menu  help\_menu = tk.Menu(menu\_bar, tearoff=0, font=('Helvetica', 11))  help\_menu.add\_command(label="About", command=show\_about)  menu\_bar.add\_cascade(label="Help", menu=help\_menu)  # Setting the menu bar  root.config(menu=menu\_bar)  # Create frames for better organization  frame\_controls = ctk.CTkFrame(network\_tab)  frame\_controls.pack(pady=(10, 0), padx=10, fill='x')  frame\_results = ctk.CTkFrame(network\_tab)  frame\_results.pack(pady=(5, 10), padx=10, fill='both', expand=True)  # Control Buttons  scan\_button = ctk.CTkButton(frame\_controls, text="Scan Network", command=start\_scan)  scan\_button.grid(row=0, column=0, padx=5, pady=5, sticky="w")  clear\_button = ctk.CTkButton(frame\_controls, text="Clear Results", command=clear\_results)  clear\_button.grid(row=1, column=0, padx=5, pady=5, sticky="w")  # Select IP Label  Select\_IP\_LB = ctk.CTkLabel(frame\_controls, text="Select IP range class:")  Select\_IP\_LB.grid(row=2, column=0, padx=5, pady=5, sticky="w")  # Create a shared StringVar to hold the selected IP class  ip\_class = tk.StringVar(value="C")  # IP classes Radio Buttons  radio\_a = ctk.CTkRadioButton(frame\_controls, text="Class A", variable=ip\_class, value="A", command=put\_into\_entry)  radio\_a.grid(row=3, column=0, padx=15, pady=5, sticky="w")  radio\_b = ctk.CTkRadioButton(frame\_controls, text="Class B", variable=ip\_class, value="B", command=put\_into\_entry)  radio\_b.grid(row=3, column=0, padx=115, pady=5, sticky="w")  radio\_c = ctk.CTkRadioButton(frame\_controls, text="Class C", variable=ip\_class, value="C", command=put\_into\_entry)  radio\_c.grid(row=3, column=0, padx=210, pady=5, sticky="w")  # Enter IP Label  Enter\_ip\_LB = ctk.CTkLabel(frame\_controls, text="Or Enter IP range:")  Enter\_ip\_LB.grid(row=4, column=0, padx=5, pady=5, sticky="w")    # Entry IP range Field  ip\_range\_entry = ctk.CTkEntry(frame\_controls, placeholder\_text="Example: 192.168.0.0-192.168.255.255")  ip\_range\_entry.grid(row=5, column=0, columnspan=3, padx=5, pady=5, sticky="ew")  ip\_range\_entry.insert(0, detected\_ip\_range)  # Progress bar  progress\_bar = ctk.CTkProgressBar(master=frame\_controls, mode='determinate', height=30, corner\_radius=3)  progress\_bar.grid(row=6, column=0, columnspan=3, padx=5, pady=5, sticky="ew")  progress\_bar.set(0)  # Configure columns in frame\_controls to expand  frame\_controls.columnconfigure(0, weight=1)  frame\_controls.columnconfigure(1, weight=1)  frame\_controls.columnconfigure(2, weight=1)  # Create a font object  custom\_font = tkFont.Font(family="Times New Roman", size=13)  # Treeview style  style = ttk.Style()  style.configure("Treeview", font=custom\_font)  style.configure("Treeview.Heading", font=custom\_font)  style\_stv = ttk.Style()  style\_stv.configure("STV.Treeview", rowheight=30)  # Treeview for results  columns = ("#", "Device Name", "MAC Address", "IP Address", "Ping Status")  tree = ttk.Treeview(frame\_results, columns=columns, show='headings')  tree.heading("#", text="#")  tree.column("#", width=50, anchor='center', stretch=False)  tree.heading("Device Name", text="Device Name")  tree.column("Device Name", width=120, anchor='w', stretch=True)  tree.heading("MAC Address", text="MAC Address")  tree.column("MAC Address", width=120, anchor='w', stretch=True)  tree.heading("IP Address", text="IP Address")  tree.column("IP Address", width=120, anchor='w', stretch=True)  tree.heading("Ping Status", text="Ping Status")  tree.column("Ping Status", width=120, anchor='w', stretch=True)  tree.pack(padx=5, pady=5, fill='both', expand=True)  # Additional information treeview  stv = ttk.Treeview(frame\_results, style="STV.Treeview")  stv.heading('#0', text='Information', anchor=tk.W)  stv.heading('#0', text='Information', anchor=tk.W)  stv.insert('', tk.END, text='Status:', iid=0, open=False)  stv.insert('', tk.END, text='Device Name:', iid=1, open=False)  stv.insert('', tk.END, text='IP Address:', iid=2, open=False)  stv.insert('', tk.END, text='MAC Address:', iid=3, open=False)  stv.insert('', tk.END, text='Ping:', iid=4, open=False)  # Use the custom font for additional information treeview  stv.pack(padx=5, pady=5, fill='both', expand=False)  stv.configure(height=5)  frame\_results.update\_idletasks() # Update the frame before getting the size  stv\_height = 165  stv.pack\_propagate(0) # Prevent Treeview from changing the frame's size  frame\_results.configure(height=stv\_height)  tree.bind("<<TreeviewSelect>>", on\_tree\_select)  def on\_tab\_selected(event):  selected\_tab = event.widget.select()  if selected\_tab == str(monitor\_tab):  update\_cpu\_ram\_plot()  # Bind the event when the tab is changed  tab\_control.bind("<<NotebookTabChanged>>", on\_tab\_selected)  # Bind the close window event to the on\_exit function  root.protocol("WM\_DELETE\_WINDOW", on\_exit)  # Start the application  root.mainloop() |