DEPARTMENT OF CYBER SECURITY

WAQAR UL HASSAN 231270
ALYAN GULZAR 231302
MUHAMMAD TALHA 231350

PROJECT Documentation

Computer Networks

22-December-2024

Project Overview:

The Wi-Fi Speed Tester is a Python-based desktop application designed to measure and display crucial network performance metrics. These metrics include download speed, upload speed, and latency (commonly referred to as ping). The application leverages the **Tkinter** library for creating a graphical user interface (GUI) that is intuitive and visually appealing. This tool is particularly useful for users looking to diagnose internet speed issues or analyze their network's performance.
The primary objective of this application is to provide users with a user-friendly and efficient way to test their internet speed while also displaying additional details about their network, such as their ISP (Internet Service Provider) and public IP address.

Features

1. **Accurate Speed Measurements**
    - **Download Speed**: Measured in Mbps (Megabits per second), indicating the rate at which data is received.
    - **Upload Speed**: Measured in Mbps, indicating the rate at which data is sent.
    - **Latency (Ping)**: Displayed in milliseconds, showing the time taken for data packets to travel to a remote server and return.
2. **Detailed Network Information**
    - Displays the user's ISP name, helping identify the service provider.
    - Shows the public IP address, which is the unique identifier of the user's device on the internet.
3. **User-Friendly GUI**
    - A sleek, dark-themed interface designed for easy navigation.
    - Incorporates images, buttons, and labels to ensure the user experience is smooth.
4. **Reset Functionality**
    - Allows users to clear all displayed test results with a single click.
5. **Responsiveness**
    - Uses threading to ensure the GUI remains interactive and responsive during speed tests.

How the Speed Tester Works:

Internet speed tests work by sending and receiving data packets to and from a server and calculating the time taken. The following metrics are measured:

1. **Latency (Ping)**:
    - A small data packet is sent to a server, and the time it takes for the packet to return is recorded.
    - Lower latency indicates a faster connection, which is essential for real-time applications like gaming or video conferencing.
2. **Download Speed**:
    - Measures the speed at which the user's device receives data from the internet.
    - Chunks of data are downloaded, and the time taken is used to calculate the rate.
3. **Upload Speed**:
    - Measures the speed at which the user's device sends data to the internet.
    - Chunks of data are uploaded, and the time taken is calculated similarly to download speed.

The Process in This Application

- The speedtest-cli library is used to handle the core functionality of performing these tests.
- Once the "Test" button is pressed, the application initiates a connection to the nearest speed test server and calculates the above metrics.
- The results are then displayed on the GUI, offering users a clear understanding of their network performance.

Graphical User Interface (GUI)

The application uses **Tkinter**, a built-in Python library, to create an intuitive and visually appealing GUI. Key elements of the interface include:

1. **Main Window**:
    - Configured with a dark theme (#141527) for better visibility and aesthetics.
    - Dynamically centered on the user's screen using precise geometry calculations.
2. **Buttons**:
    - **Test Button**: Triggers the internet speed test.
    - **Reset Button**: Resets all displayed metrics, allowing users to perform another test.
3. **Labels**:
    - Display real-time results, including latency, download speed, upload speed, ISP name, and IP address.
4. **Images**:
    - Custom icons and background images enhance the application's appearance and provide a professional look.

Code Breakdown

Importing Libraries

- **Tkinter**: Used for GUI creation and management.
- **speedtest-cli**: Provides methods for measuring download, upload speeds, and latency.
- **threading**: Ensures the GUI remains responsive during speed tests by running the test in a separate thread.

GUI Configuration

The application window (root) is configured with a dark theme, set dimensions, and is centered on the screen using:

root.geometry("%dx%d+%d+%d" % (w, h, x, y))

Custom icons and background images enhance the application's visual appeal.

Core Functions

1. **Test Function**
    - Initiates the speed test when the "Test" button is pressed.
    - Utilizes a separate thread to run the test, ensuring the GUI remains responsive.
    - Updates labels dynamically with the results (ping, upload, download speeds, ISP name, and IP address).

```
def Test():
    def run_speed_test():
        Download.config(text="Loading")
        test = speedtest.Speedtest()
        test.get_servers([])
        data = test.get_config()

        uploading = round(test.upload() / (1024 * 1024), 2)
        downloading = round(test.download() / (1024 * 1024), 2)

        ping.config(text=test.results.ping)
        upload.config(text=uploading)
        download.config(text=downloading)
        Download.config(text=downloading)
        service.config(text=data["client"]["isp"])
        ip.config(text=data["client"]["ip"])

    threading.Thread(target=run_speed_test).start()
```

How does this logic works:

- **def Test():**
  This defines the main function Test(), which is likely triggered when the user clicks a "Test" button on the GUI. This function initializes and runs the speed test.
- **def run_speed_test():**
  This defines an inner function run_speed_test() that contains the core logic for performing the speed test. It is nested within the Test() function and will be run in a separate thread to keep the GUI responsive.
- **Download.config(text="Loading")**
  This updates the Download label in the GUI to display the text "Loading", signaling to the user that the speed test is in progress. This gives a visual cue that the application is performing the test.
- **test = speedtest.Speedtest()**
  This initializes the Speedtest() class from the speedtest-cli library, which provides methods to perform the actual internet speed test (measuring download speed, upload speed, and latency).
- **test.get_servers([])**
  This method fetches the available servers for the speed test. The empty list [] means it will use the default server (the closest one by latency), but no specific server is manually selected here.
- **data = test.get_config()**
  This method retrieves configuration data, including the user's ISP (Internet Service Provider) and public IP address. The data variable stores this information as a dictionary.
- **uploading = round(test.upload() / (1024 * 1024), 2)**
  The test.upload() method measures the upload speed in bits per second (bps). This result is divided by 1024 * 1024 to convert it into megabits per second (Mbps), then rounded to two decimal places for better readability. The result is stored in the uploading variable.

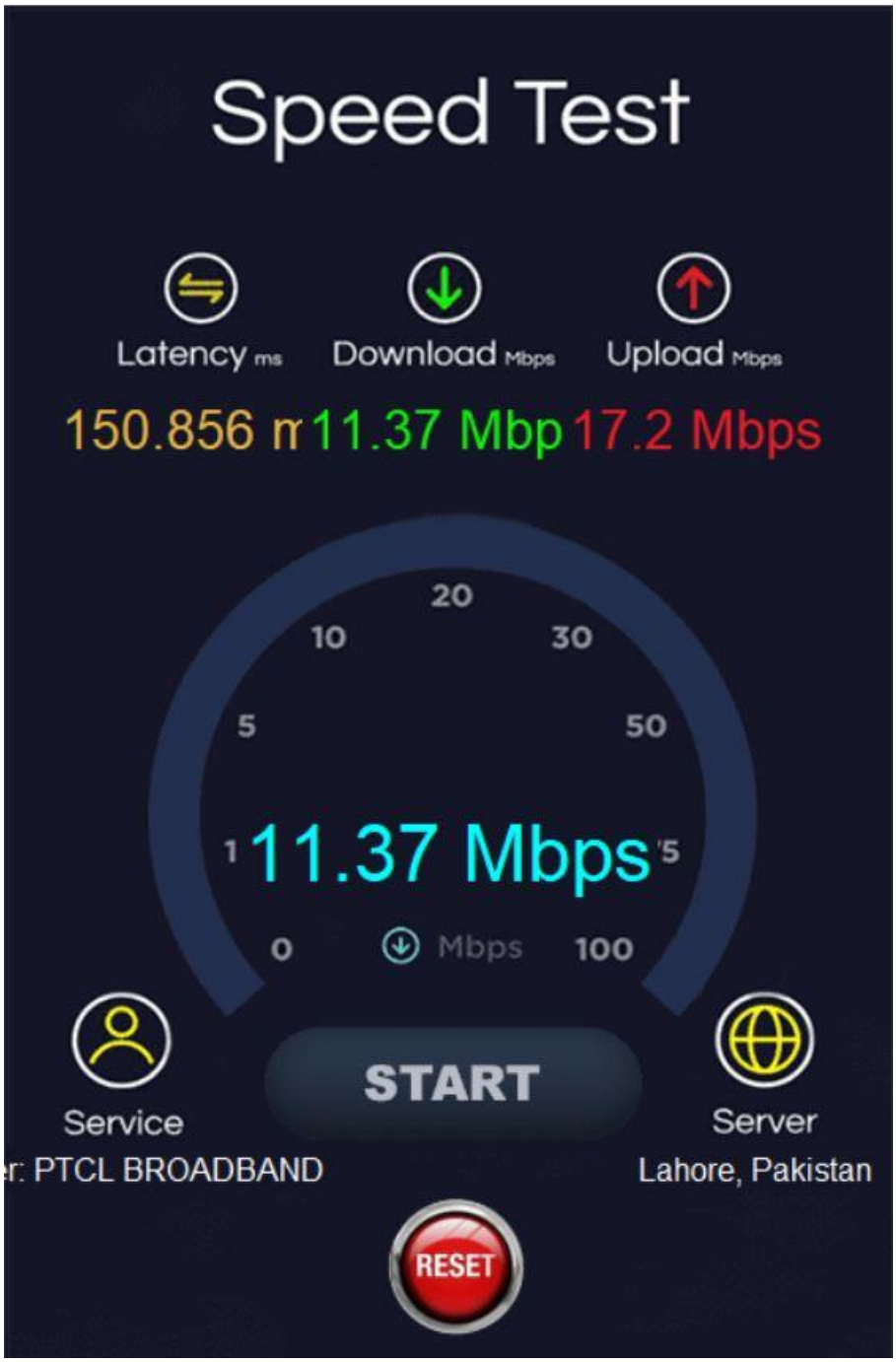
- **downloading = round(test.download() / (1024 * 1024), 2)**
  Similarly, the test.download() method measures the download speed in bps. The result is divided by 1024 * 1024 to convert it to Mbps and rounded to two decimal places. The result is stored in the downloading variable.
- **ping.config(text=test.results.ping)**
  This updates the ping label in the GUI to display the ping value (latency) measured by the test.results.ping attribute. This is the round-trip time for data to travel to the server and back, displayed in milliseconds.
- **upload.config(text=uploading)**
  This updates the upload label with the calculated upload speed (uploading), displaying the value in Mbps.
- **download.config(text=downloading)**
  This updates the download label with the calculated download speed (downloading), displaying the value in Mbps.
- **Download.config(text=downloading)**
  This updates the Download label again, this time displaying the download speed (downloading). This is likely to be part of the GUI where users can easily see the current download speed.
- **service.config(text=data["client"]["isp"])**
  This updates the service label to show the name of the user's ISP (Internet Service Provider), which is retrieved from data["client"]["isp"] in the configuration data.
- **ip.config(text=data["client"]["ip"])**
  This updates the ip label to display the user's public IP address, which is retrieved from data["client"]["ip"] in the configuration data.
- **threading.Thread(target=run_speed_test).start()**
  This creates a new thread to run the run_speed_test() function. By using threading, the speed test can be performed in the background without freezing the GUI. This allows the user to continue interacting with the application while the test runs.

2. **Button Actions**
   - "Test" Button: Starts the internet speed test.
   - "Reset" Button: Clears all displayed metrics to allow a new test.

GUI Elements

- **Labels**: Display test results (ping, upload, download speeds, ISP name, and IP address).
- **Buttons**: Trigger actions (start test or reset results).
- **Images**: Provide a polished look to the interface.

Graphical User Interface:



Instructions for Use

1. Launch the application by running the Python script.
2. Click the **Test** button to initiate the speed test. The application will:
    - Display "Loading" while the test is in progress.
    - Update the displayed metrics (ping, upload speed, download speed, ISP name, and IP address).
3. Click the **Reset** button to clear all displayed metrics and prepare for another test

Future Enhancements

1. **Graphical Representations**
    - Integrate charts and graphs to visualize speed test results over time.
2. **Cross-Platform Compatibility**
    - Extend the application's compatibility to macOS and Linux environments.
3. **Optimization Suggestions**
    - Provide recommendations to improve internet speed based on test results.
4. **Advanced Metrics**
    - Include additional metrics like jitter and packet loss for more comprehensive performance analysis.

Full Code:

```
from tkinter import *
import speedtest
import threading

root = Tk()
root.title("Internet Speed Test")
root.resizable(False, False)
root.configure(bg="#141527")

# Dimensions for the Window
w = 450
h = 700
ws = root.winfo_screenwidth()
hs = root.winfo_screenheight()
x = (ws / 2) - (w / 2)
y = (hs / 2) - (h / 2)
root.geometry("%dx%d+%d+%d" % (w, h, x, y))

def test_speed():
    # Set initial text while loading
    ping.config(text="Testing...")
    upload.config(text="Testing...")
    download.config(text="Testing...")
    Download.config(text="Testing...")
    service.config(text="Testing...")
    ip.config(text="Testing...")

    # Initialize SpeedTest
    test = speedtest.Speedtest()
    best = test.get_best_server()
    download_speed = round(test.download() / (1024 * 1024), 2)  # Download in
Mbps
    uploading = round(test.upload() / (1024 * 1024), 2)  # Upload in Mbps
    ping_value = test.results.ping

    ping.config(text=f"{ping_value} ms")
    upload.config(text=f"{uploading} Mbps")
    download.config(text=f"{download_speed} Mbps")
    Download.config(text=f"{download_speed} Mbps")
```

```
    service.config(text=f"{best['sponsor']}")
    ip.config(text=f"{best['host']}")

def start_test():
    def run_speed_test():
        test_speed()

    threading.Thread(target=run_speed_test, daemon=True).start()

def Check():
    start_test()

def Reset():
    ping.config(text="--")
    upload.config(text="--")
    download.config(text="--")
    Download.config(text="--")
    service.config(text="- - -")
    ip.config(text="- - - - - - - -")

image_icon = PhotoImage(file="D:/Cyber Study/Speed-Test-
main/images/logo.png")
root.iconphoto(False, image_icon)

Top = PhotoImage(file="D:/Cyber Study/Speed-Test-main/images/background.png")
Label(root, image=Top, bg="#141527").place(x=0, y=0)

Test_Image = PhotoImage(file="D:/Cyber Study/Speed-Test-
main/images/button.png")
Test_Button = Button(
    root,
    image=Test_Image,
    bg="#141527",
    bd=0,
    activebackground="#141527",
    cursor="hand2",
    command=Check,
)


Test_Button.place(x=125, y=510)
```

```
Reset_Image = PhotoImage(file="D:/Cyber Study/Speed-Test-
main/images/reset.png")
Reset_Button = Button(
    root,
    image=Reset_Image,
    bg="#141527",
    bd=0,
    activebackground="#141527",
    cursor="hand2",
    command=Reset,
)
Reset_Button.place(x=190, y=600)

# Labels to show values
ping = Label(root, text="--", font="arial 20", bg="#141527", fg="#e9b342")
ping.place(x=100, y=215, anchor="center")

download = Label(root, text="--", font="arial 20", bg="#141527", fg="#0cf107")
download.place(x=225, y=215, anchor="center")

upload = Label(root, text="--", font="arial 20", bg="#141527", fg="#e61c25")
upload.place(x=350, y=215, anchor="center")

Download = Label(root, text="--", font="arial 30", bg="#141527", fg="#00FFFF")
Download.place(x=225, y=430, anchor="center")

service = Label(root, text="- - -", font="arial 12", bg="#141527", fg="white")
service.place(x=55, y=590, anchor="center")

ip = Label(root, text="- - - - - - - -",
          font="arial 12", bg="#141527", fg="white")
ip.place(x=380, y=590, anchor="center")

root.mainloop()
```