NETWORK PERFORMANCE MONITOR

21CSCT302J /COMPUTER NETWORKS

A MINI PROJECT REPORT

Submitted by

S HARINI	**RA2311003020399**
S M VIVEK RAO	**RA2311003020409**
S KRITHIKA	**RA2311003020416**

Under the guidance of

Dr. R. Nagalakshmi, M.E., Ph.D.,

(Assistant Professor, Department of Computer Science and Engineering)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

RAMAPURAM, CHENNAI

October 2025

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
(Deemed to be University U/S 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled **"NETWORK PERFORMANCE MONITOR" is** the bonafide work of **S HARINI (RA2311003020399), S M VIVEK RAO (RA2311003020409), S KRITHIKA (RA2311003020416)** who carried out the project work under my supervision. This project work confirms to 21CSC302J /Computer Networks, V Semester, III year, 2025.

SIGNATURE
Dr. R. Nagalakshmi, M.E, Ph.D.,

Assistant Professor
Computer Science and Engineering,
SRM Institute of Science and Technology,
Ramapuram, Chennai.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

RAMAPURAM, CHENNAI

DECLARATION

We hereby declare that the entire work contained in this project report titled "**NETWORK PERFORMANCE MONITOR**" has been carried out by **S HARINI (RA2311003020399), S M VIVEK RAO (RA2311003020409), S KRITHIKA (RA2311003020416)** at SRM Institute of Science and Technology, Ramapuram, Chennai, under the guidance of **Dr. R. Nagalakshmi, M.E., Ph.D.,** Assistant professor, Department of Computer Science and Engineering.

Place: Chennai	**S HARINI**
Date:	**S M VIVEK RAO**
	S KRITHIKA

ABSTRACT

In modern computer networks, continuous monitoring of network performance is essential to ensure reliable connectivity, efficient bandwidth usage, and quick identification of issues such as high latency, packet loss, or abnormal bandwidth consumption. This project presents a **Live Network Monitoring Tool** implemented in Python, which provides real-time visualization and analysis of network metrics including latency, packet loss, upload speed, and download speed. By combining system-level network statistics with periodic host reachability checks, the tool offers both detailed and intuitive insights into network behavior.

The system leverages the **psutil** library to track network I/O statistics and compute bandwidth usage over time, while the **ping3** library is used to perform ICMP ping requests to external hosts, thereby measuring latency and detecting packet loss. These metrics are collected at regular intervals and stored in a structured format for live plotting. To ensure continuous operation and timely updates, the project employs the **schedule** library for task scheduling and **multi-threading** to allow monitoring and visualization to run concurrently without blocking the main thread.

A key feature of the tool is its **interactive live plotting** implemented using **Matplotlib**, where three synchronized subplots display latency, packet loss, and network bandwidth (both upload and download) in real-time. The tool automatically adjusts the axes to accommodate the latest data points and maintains a sliding window of recent measurements to focus on current network conditions. Threshold-based alerts are incorporated to notify the user when metrics exceed predefined limits, such as high latency, excessive packet loss, or unusually high bandwidth usage, thereby enabling quick troubleshooting and proactive network management.

This project not only serves as a practical utility for monitoring network performance in real-time but also provides a hands-on demonstration of Python programming, data visualization, multithreading, and automated scheduling. It is suitable for educational purposes, network diagnostics, and personal monitoring of home or office networks. By integrating live data collection with dynamic visualization, the Live Network Monitoring Tool enables users to gain actionable insights into network health, identify potential bottlenecks, and optimize performance effectively.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

The beauty of the modern digital landscape lies in the fact that an efficient running network is simply unalienable. Whether for business firms, service providers, or personal use, the performance of a network determines productivity, customer satisfaction, and resultant operational efficiency. Increasing usage of cloud services, video conferencing, and data-intensive applications has transformed real-time monitoring of network performance into one of the most critical elements of infrastructure integrity. This is a network performance monitor, which continuously assesses the health of a network by monitoring key performance indicators like latency, packet loss, and bandwidth utilization.

The goal of this project is to design a simple but effective tool that can give immediate feedback regarding network behavior. The Network Performance Monitor utilizes the Python programming language, along with additional libraries such as `psutil` and `ping3`, to gather data about the network and then review it. It does pings on a provided host-in this case, Google's public DNS server at IP address `8.8.8.8`, and measures round-trip time to evaluate latency. It also uses network input/output counters on the system to monitor bandwidth usage, including data sent and received. All these metrics, including the packet loss rate, are required to identify where the bottleneck lies, slowdowns occurred unexpectedly, or where services simply went down in the network.

Network latency is a very important metric that measures the responsiveness of a connection. When latency is high, communication becomes delayed, which could be frustrating with time-sensitive applications such as online gaming, video conferencing, or real-time data streaming. This tool directly measures latency by pinging a host and measuring the time taken for data packets to travel to their destination and back. To choose threshold values for acceptable latency, the project sets a threshold value for acceptable latency at 100 milliseconds and this system raises an alert if it goes beyond that threshold. That feature is vital because they require quick response times in such environments to have smooth user experience.

The other very critical factor includes packet loss-data packets failed to reach their destination. Even minor loss of packets resulted in communication breaks and artifacts in the data due to retransmission, reduced throughput, and latency. The Network Performance Monitor logs unreachability from a host or packet loss. Administrators get real-time information about detecting problems in networks.

- **Purpose:** The project is designed to monitor network performance in real-time, tracking key metrics such as latency, packet loss, upload speed, and download speed to ensure reliable connectivity and efficient network usage.
- **Data Collection:** It uses Python libraries like psutil to fetch network I/O statistics and ping3 to measure latency and detect packet loss, providing a comprehensive overview of network health.
- **Visualization:** Real-time graphs are generated using Matplotlib, showing live updates of latency, packet loss, and bandwidth, which helps users quickly identify network issues.
- **Automation & Alerts:** The tool employs scheduling and multithreading to continuously monitor the network without interruption and provides alerts when metrics exceed predefined thresholds, enabling proactive troubleshooting.

Another important feature is bandwidth monitoring, which monitors the amount of data being sent and received over a network. Users can be sure their network is not overburdened by the transfer of excessive amounts of data from any system that could slow down the critical processes by monitoring network bandwidth usage. This tool includes a threshold on bandwidth usage, at 1 megabyte per second, and will raise alerts when the system surpasses those levels; this will help identify misuse or overload of network resources by the administrators.

The project runs in a continuous loop; it is scheduled to check every 10 seconds, thus enabling real-time monitoring of networks and giving up-to-date information regarding the performance of the network to administrators. The historical data collected during monitoring is stored and can be used to compute in-depth analysis of network trends over time. That feature can help point out the places where the same recurring issues are appearing, which may indicate how degrading over time the network's performance is.

In a summary, this Network Performance Monitor is a very useful tool that guarantees smooth and stable networking. Real-time detection of network latency, packet loss, and bandwidth usage allows users to foresee network problems before their actual main disruption occurs. It, therefore, makes it a suitable solution for use in personal and professional environments, which provides an essential tool in maintenance of robust network performance.

CHAPTER 2
PROBLEM STATEMENT

Companies operating cloud-based applications to a plain citizen streaming high-definition media on their personal computers, the performance of networks is basically fundamental to smooth and flawless operations. However, the issue remains when maintaining stable and efficient networks is not an easy task. These can introduce latency, packet loss, and bandwidth hunger; all factors that degrade network performance and can cause disruptions that easily affect user experience and operational efficiency.

One of the main issues in network management is real-time problem detection and responses to such performance issues. Among the most common causes of reduced network quality is latency, that is, delay in the transmission of data from one point to another. It can be critical when the scenario needs a quick response, which includes video conferencing or online gaming. High latency may cause disappointment with being frustrated by this delay in communication. Packet loss may also critically degrade network reliability because slowdowns or retran- missions will consume important throughput while packets do not reach their destination. Similarly, too much bandwidth utilization can also lead to network congestion that naturally throttles the potential for further data transfers and can lead to a bottleneck in their performances.

Traditional network performance monitoring tools have traditionally required complex configurations, costly hardware or technical expertise that is infeasible for smaller organizations or individual users. Furthermore, unless constant monitoring is performed, network issues may go undetected until the problems are at a point where service quality is severely impacted. It can result in downtime, lost productivity, and a decrease in customer satisfaction especially in those environments where high performance and uptime are key.

It needs to be accessed in real time while being able to monitor the key network performance indicators, latency, packet loss and bandwidth, where timely warnings are provided in case any of these issues occur. This tool will enable the users, and even the administrators of the networks, to be proactive in keeping the health and performance of the network going rather than waiting to see significant problems come out before responding. Perhaps continuous monitoring of network conditions and giving users clear insight into possible problem areas allows them to make informed decisions about optimizing their networks and avoiding costly downtimes or service interruptions.

CHAPTER 3
PROPOSED METHODOLOGY

3.1 OBJECTIVE

The primary objective of this project is to design and implement a real-time network monitoring system that ensures the efficient and reliable performance of computer networks. In modern networked environments, uninterrupted connectivity and optimal data transfer are critical for both personal and organizational operations. Network performance can be affected by multiple factors, including high latency, packet loss, and bandwidth congestion. Therefore, continuously monitoring these parameters is essential for proactive network management and troubleshooting.

This project focuses on monitoring three key aspects of network performance: **latency**, **packet loss**, and **bandwidth utilization**. Latency is measured by pinging a target host and calculating the response time, which helps identify delays in data transmission. Packet loss is recorded when a ping request fails or times out, indicating potential network instability or congestion. Bandwidth utilization is tracked by measuring the bytes sent and received over the network, providing insight into the network's load and helping detect abnormal spikes in usage.

The system not only collects these metrics in real time but also maintains a historical log for trend analysis. By comparing current data against predefined thresholds, the system generates alerts when network performance deviates from acceptable limits. This proactive approach enables network administrators to identify and resolve issues before they escalate, ensuring uninterrupted service and improved overall network reliability.

Additionally, the project demonstrates the practical application of **network monitoring tools** such as psutil for bandwidth measurement and ping3 for latency checks, integrated into a scheduled monitoring framework for automated execution. Ultimately, this project contributes to understanding how real-time monitoring can enhance network performance management, prevent potential downtime, and support informed decision-making in network administration.

3.2 SCOPE

The scope of this project focuses on providing a cost-effective and efficient solution for network performance monitoring, which can be utilized by individuals, small organizations, or enterprises. It offers key functionalities such as latency tracking, bandwidth usage monitoring, and packet loss detection. While initially designed to monitor a single host, the system is scalable and can be expanded to handle multiple hosts or additional network metrics, such as jitter and packet delay variance. Future enhancements may include integration with data visualization tools, remote alert systems like SMS or email notifications, and advanced network diagnostic features. The project aims to fill a gap in affordable, easy-to-use network monitoring tools by creating a system that runs automatically, logs performance data, and provides real-time feedback to users, ensuring optimal network performance.

3.3 EXISTING SYSTEM

Traditional network performance monitoring solutions often rely on complex hardware, specialized software, or expensive third-party services, making them inaccessible to smaller organizations or individuals. These systems typically require extensive configuration and technical expertise, creating barriers for non-technical users. Moreover, they are designed primarily for large-scale networks, which means they may be overkill for personal or small business use. In addition, existing tools often focus on isolated metrics without offering a comprehensive view of network health, or they may lack real-time alerting, which can delay responses to network issues. While some existing systems provide high levels of customization and integration with other enterprise tools, they can be resource-intensive, requiring significant computing power, storage, and bandwidth to operate effectively.

Furthermore, most traditional systems do not include sufficient error handling mechanisms, which can lead to crashes or missed data during network disruptions. This can leave networks vulnerable during periods of instability. As a result, users often struggle to maintain stable, high-performing networks without a costly or cumbersome solution in place. By addressing these gaps, the proposed Network Performance Monitor (NPM) offers a lightweight, cost-effective alternative that is easy to use, efficient, and reliable for monitoring key network.

3.4 PROPOSED SYSTEM

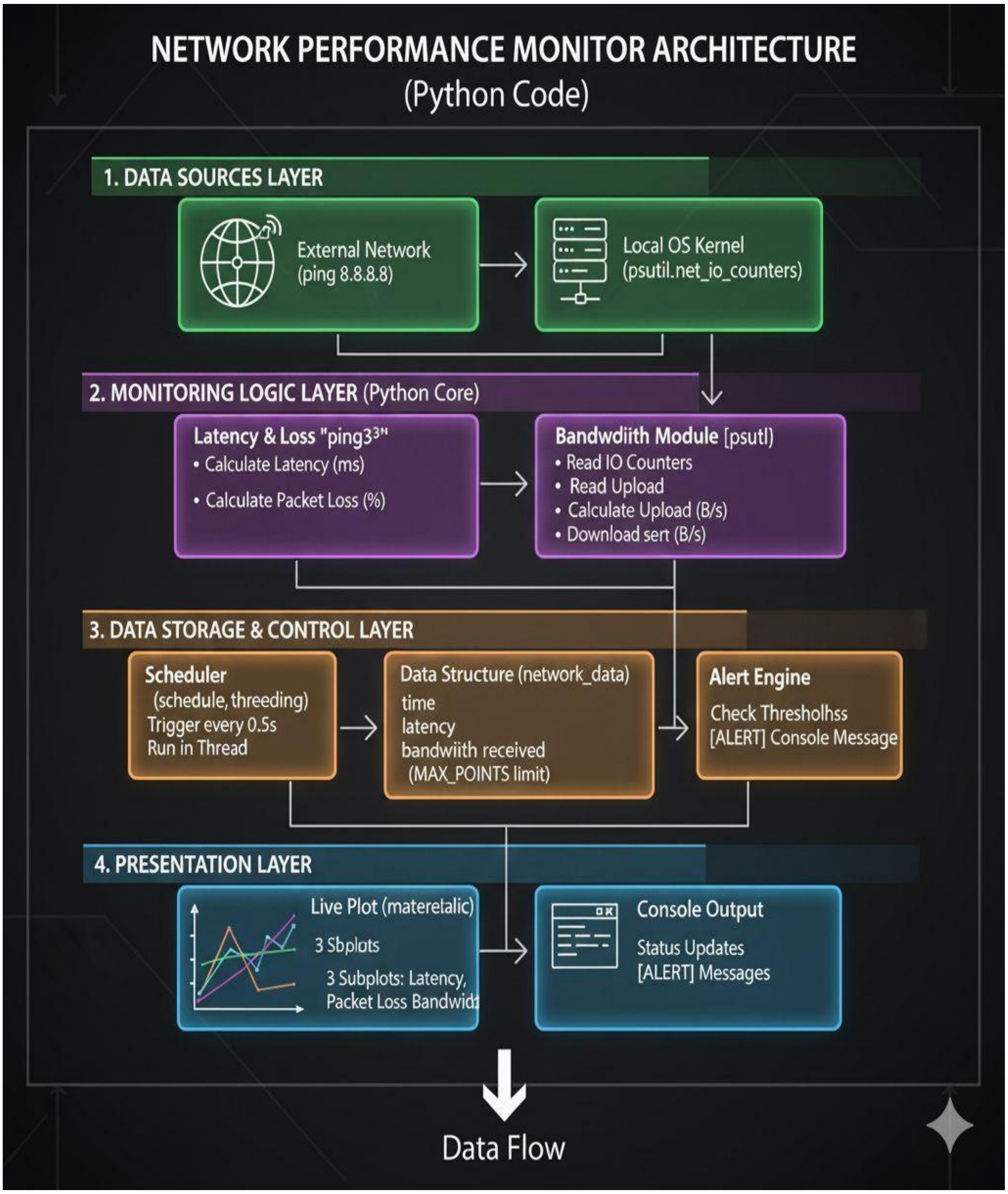
The proposed system is a Python-based Network Performance Monitor (NPM) designed to track and analyse key network metrics in real time. This system will monitor parameters such as **latency**, **bandwidth usage**, and **packet loss** to provide a comprehensive view of the network's performance. The system utilizes libraries like psutil to gather bandwidth data and ping3 to assess network latency by pinging a specific IP address (e.g., Google DNS 8.8.8.8). This approach enables the detection of high-latency events, abnormal bandwidth usage, and network unreachability, ensuring early identification of potential network issues.

The system is designed to execute monitoring tasks at regular intervals using the schedule library, providing periodic checks of network performance. The data gathered is continuously compared against user-defined thresholds. If metrics such as latency exceed a predefined threshold (e.g., 100 ms) or bandwidth usage surpasses a certain limit (e.g., 1 MBps), the system generates **alerts** to notify users of potential network problems. These real-time alerts allow users to take immediate action to mitigate the issues before they cause significant network disruptions.

Moreover, the system will include robust **error-handling mechanisms** to maintain continuous monitoring even when network connectivity issues arise. For instance, if the network becomes unreachable, the system will log the error and continue to function without crashing. This ensures that the NPM is reliable and resilient under various network conditions. Future enhancements to the proposed system could include the addition of data visualization tools, advanced metrics tracking, and integration with more sophisticated network diagnostic tools.

The proposed system's architecture is designed to be both lightweight and efficient, ensuring that it can operate in various network environments without consuming excessive system resources. By utilizing Python's psutil for real-time monitoring of network bandwidth (sent and received data) and ping3 for measuring latency, the NPM can provide an accurate snapshot of the network's performance at any given time. Additionally, the system is modular, allowing for easy customization and future expansion, such as monitoring multiple hosts simultaneously or integrating additional metrics like jitter and packet delay variance. This modularity makes the system adaptable to different network scales, from home setups to more complex enterprise networks.

3.5 ARCHITECTURE DIAGRAM


NETWORK PERFORMANCE MONITOR ARCHITECTURE
(Python Code)
1. DATA SOURCES LAYER
External Network (ping 8.8.8.8)
Local OS Kernel (psutil.net_io_counters)
2. MONITORING LOGIC LAYER (Python Core)
Latency & Loss "ping3³"
• Calculate Latency (ms)
• Calculate Packet Loss (%)
Bandwdiith Module [psutl)
• Read IO Counters
• Read Upload
• Calculate Upload (B/s)
• Download sert (B/s)
3. DATA STORAGE & CONTROL LAYER
Scheduler (schedule, threeding) Trigger every 0.5s Run in Thread
Data Structure (network_data) time latency bandwiith received (MAX_POINTS limit)
Alert Engine Check Thresholhss [ALERT] Console Message
4. PRESENTATION LAYER
Live Plot (materetalic) 3 Sbplots 3 Subplots: Latency, Packet Loss Bandwid
Console Output Status Updates [ALERT] Messages
Data Flow

CHAPTER 4

MODULES DESCRIPTION

The Network Performance Monitor (NPM) is composed of several key modules, each responsible for specific aspects of network monitoring and alerting. Below is a detailed breakdown of each module and its functionality:

1. Latency Monitoring Module

This module is responsible for measuring the **latency** (network delay) between the local machine and a specified remote host, typically using an ICMP ping. It leverages the ping3 library to send periodic ping requests to the chosen IP address (e.g., Google's DNS server 8.8.8.8) and measures the response time in milliseconds. If the latency exceeds a user-defined threshold, such as 100 ms, the module generates an alert. The latency values are also logged for performance trend analysis.

2. Bandwidth Monitoring Module

The **bandwidth** module monitors the data sent and received by the system over the network. Using the psutil library, this module tracks the number of bytes transmitted (sent) and received by the network interface. The module periodically checks this data, calculates the bandwidth usage in MB, and compares it to a defined threshold (e.g., 1 MBps). If the bandwidth usage exceeds this limit, the module issues an alert, ensuring users are notified of potential excessive data consumption.

3. Alerting System Module

This module handles the **real-time alerting** functionality of the NPM. It monitors the outputs from the latency and bandwidth modules and compares the values against predefined thresholds. If any metric breaches its threshold, the alerting system sends an immediate warning to the console, notifying the user. In the future, this module could be expanded to send alerts via email or SMS, enabling remote monitoring.

4. Error Handling and Recovery Module

To ensure the NPM runs smoothly even in unstable network conditions, this module manages **error handling**. If a host becomes unreachable or there is a socket error, the module captures the exception and prevents the system from crashing. This ensures continuous network monitoring, even if transient network issues occur. Errors such as unreachable hosts are logged for further review and troubleshooting.

4.1 SOURCE CODE

```
import psutil
from ping3 import ping
import time
import schedule
import matplotlib.pyplot as plt
from datetime import datetime
from threading import Thread

# ------------------ CONFIG ------------------
LATENCY_THRESHOLD = 100      # ms
PACKET_LOSS_THRESHOLD = 0.1  # 10%
BANDWIDTH_THRESHOLD = 1e6    # bytes
MAX_POINTS = 50              # max points shown on graph

# Data storage
network_data = {
    "time": [],
    "latency": [],
    "packet_loss": [],
    "bandwidth_sent": [],
    "bandwidth_received": []
}
prev_io = psutil.net_io_counters()
prev_time = time.time()
running = True

# ------------------ MONITORING ------------------
def monitor_latency(host="8.8.8.8"):
    """Ping the host and measure latency."""
    try:
        latency = ping(host, timeout=2)
        if latency is None:
```

```
print(f"[ALERT] Host {host} unreachable!")
            packet_loss = 100
            latency_ms = 0
        else:
            latency_ms = latency * 1000
            packet_loss = 0
            if latency_ms > LATENCY_THRESHOLD:
                print(f"[ALERT] High latency: {latency_ms:.2f} ms")

        network_data["latency"].append(latency_ms)
        network_data["packet_loss"].append(packet_loss)
    except Exception as e:
        print(f"[ERROR] Ping failed: {e}")
        network_data["packet_loss"].append(100)
        network_data["latency"].append(0)

def monitor_bandwidth():
    """Calculate bandwidth since last check."""
    global prev_io, prev_time
    current_io = psutil.net_io_counters()
    current_time = time.time()
    interval = current_time - prev_time

    sent_bps = (current_io.bytes_sent - prev_io.bytes_sent) / interval
    recv_bps = (current_io.bytes_recv - prev_io.bytes_recv) / interval

    if sent_bps > BANDWIDTH_THRESHOLD:
        print(f"[ALERT] High upload speed: {sent_bps/1e6:.2f} MB/s")
    if recv_bps > BANDWIDTH_THRESHOLD:
        print(f"[ALERT] High download speed: {recv_bps/1e6:.2f} MB/s")

    prev_io = current_io
    prev_time = current_time
    network_data["bandwidth_sent"].append(sent_bps)
    network_data["bandwidth_received"].append(recv_bps)

    print(f"Upload: {sent_bps/1e6:.2f} MB/s | Download: {recv_bps/1e6:.2f} MB/s")

def monitor_network():
    """Run latency and bandwidth checks."""
    timestamp = datetime.now().strftime("%H:%M:%S")
    network_data["time"].append(timestamp)

    print(f"\n--- Network Check @ {timestamp} ---")
    monitor_latency()
    monitor_bandwidth()

    # Keep only recent data
    for key in network_data:
        network_data[key] = network_data[key][-MAX_POINTS:]

# ------------------ PLOTTING ------------------
plt.ion()  # Interactive mode ON
fig, axs = plt.subplots(3, 1, figsize=(10, 8))
fig.suptitle("Live Network Monitor")

latency_line, = axs[0].plot([], [], label="Latency (ms)", color="orange")
packet_line, = axs[1].plot([], [], label="Packet Loss (%)", color="red")
upload_line, = axs[2].plot([], [], label="Upload (MB/s)", color="blue")
download_line, = axs[2].plot([], [], label="Download (MB/s)", color="green")
for ax in axs:
    ax.legend()
    ax.grid(True)
    ax.tick_params(axis="x", rotation=45)

axs[0].set_ylabel("Latency (ms)")
```

```
axs[1].set_ylabel("Packet Loss (%)")

axs[2].set_ylabel("Speed (MB/s)")

def update_plot():
    """Force refresh of the live graph."""
    if not network_data["time"]:
        return
    x = range(len(network_data["time"]))  # use numeric X for better performance

    latency_line.set_data(x, network_data["latency"])
    packet_line.set_data(x, network_data["packet_loss"])
    upload_line.set_data(x, [s/1e6 for s in network_data["bandwidth_sent"]])
    download_line.set_data(x, [r/1e6 for r in network_data["bandwidth_received"]])
    for ax in axs:
        ax.relim()
        ax.autoscale_view()

    # Redraw immediately
    fig.canvas.draw()
    fig.canvas.flush_events()
# ---------------- SCHEDULER ----------------
def run_scheduler():
    schedule.every(0.5).seconds.do(lambda: (monitor_network(), update_plot()))
    while running:
        schedule.run_pending()
        time.sleep(0.1)
# Start scheduler in background
thread = Thread(target=run_scheduler, daemon=True)
thread.start()
print("✅ Live network monitor started — close the graph window to stop.")
plt.show(block=True)
running = False
```

4.2 SCREENSHOTS

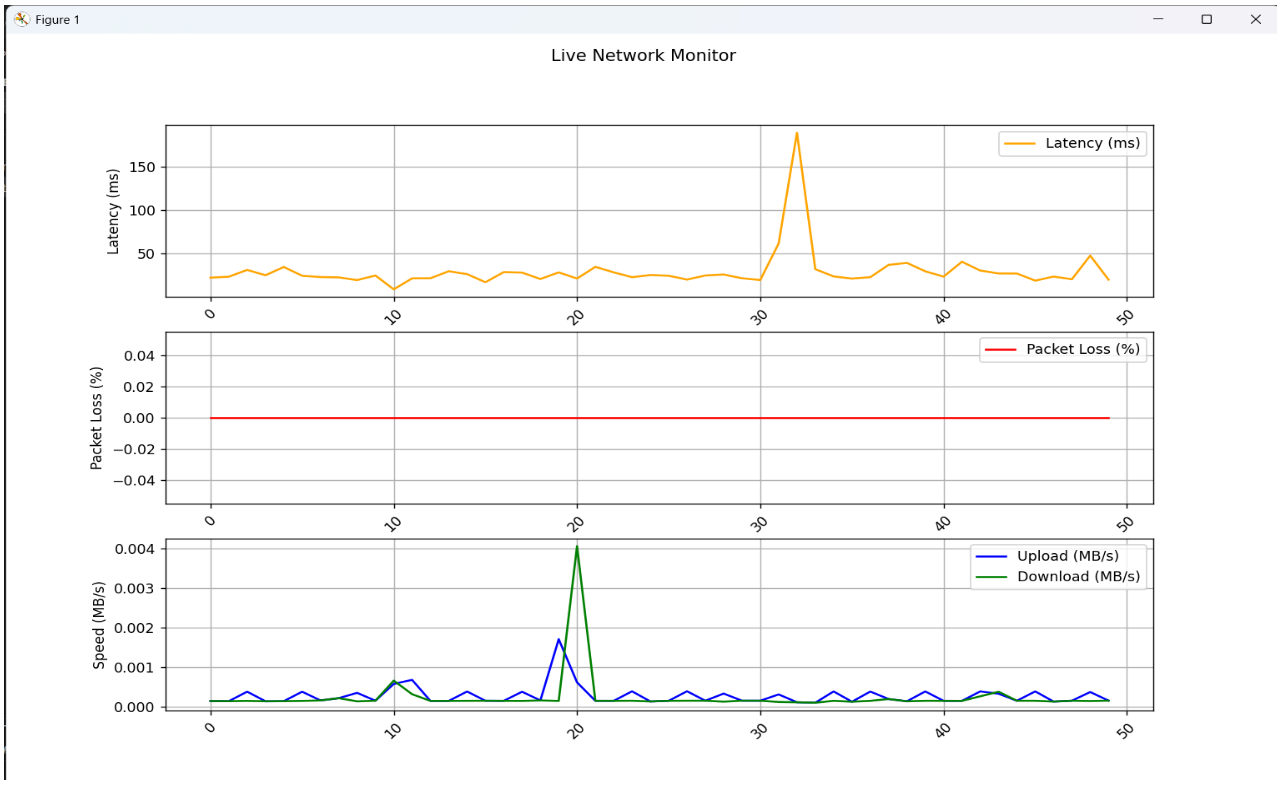


FIGURE 4.1

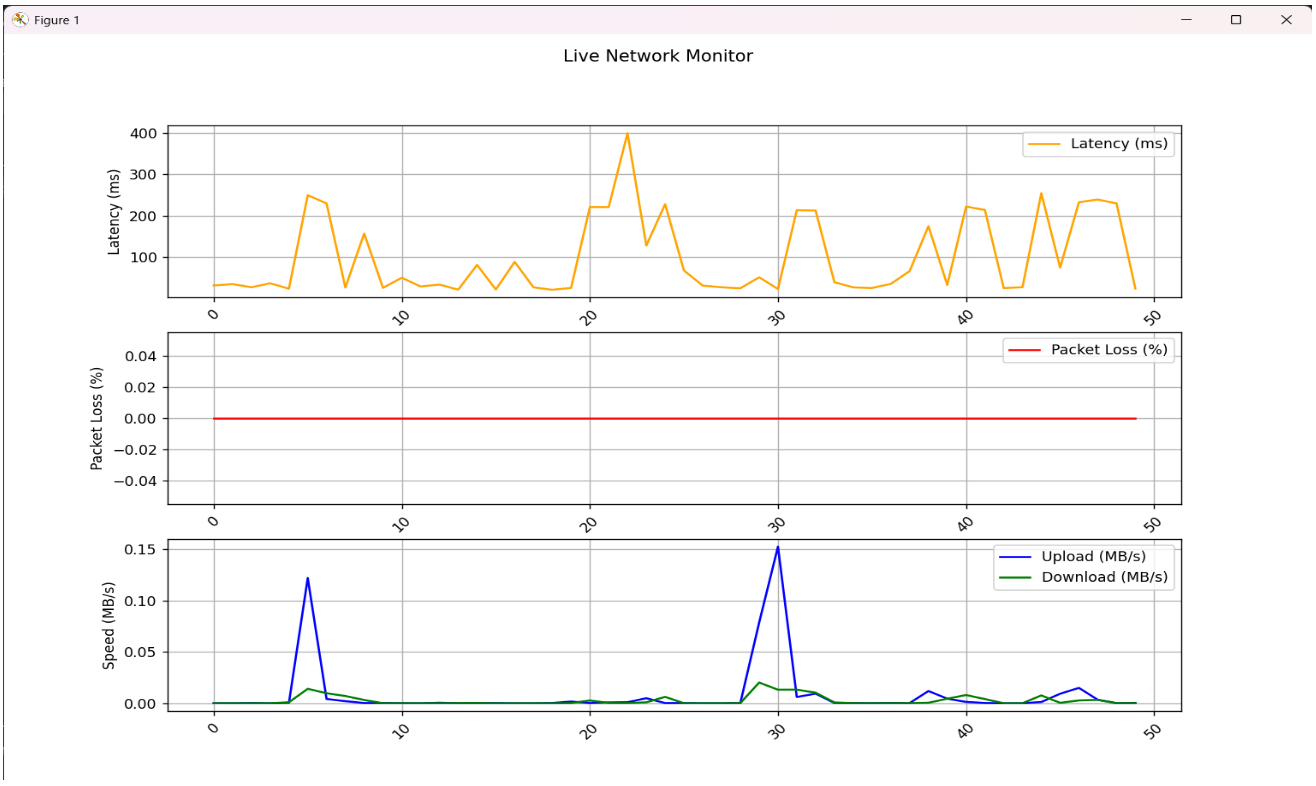
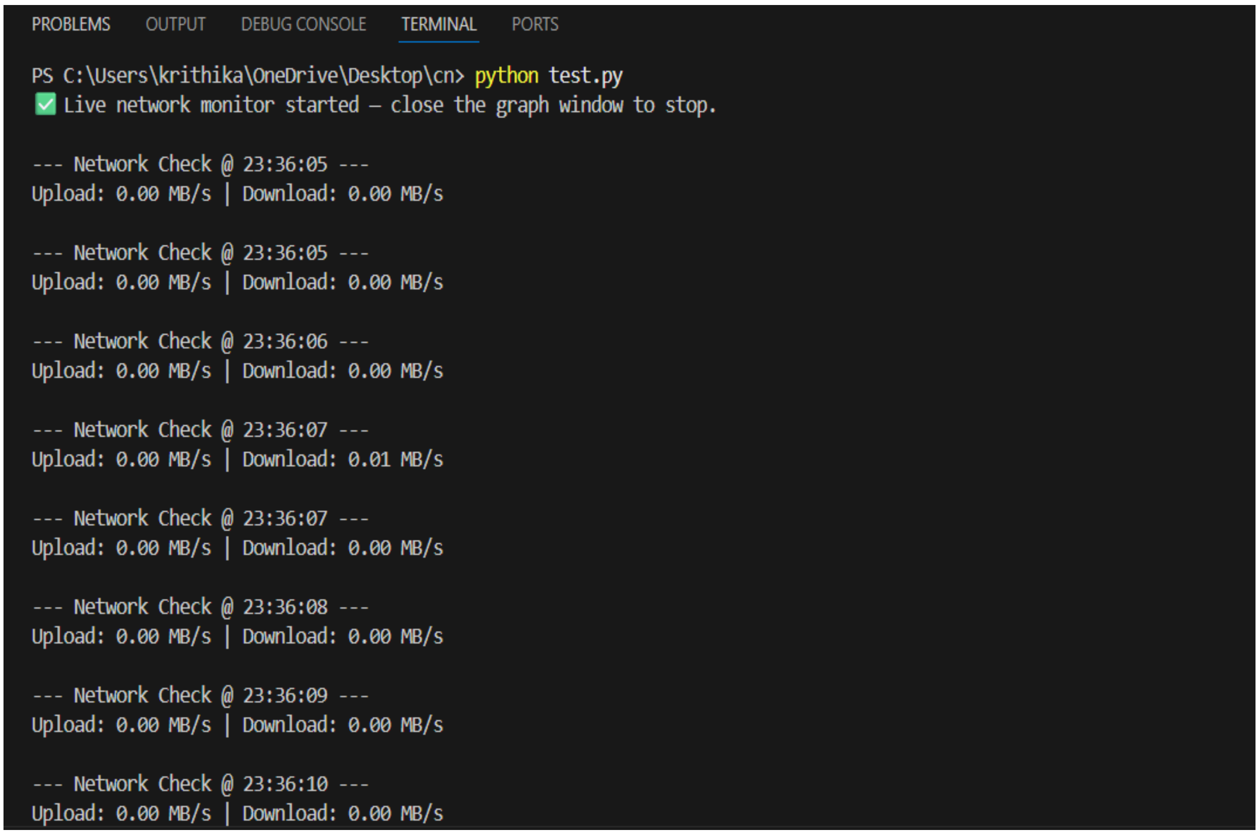
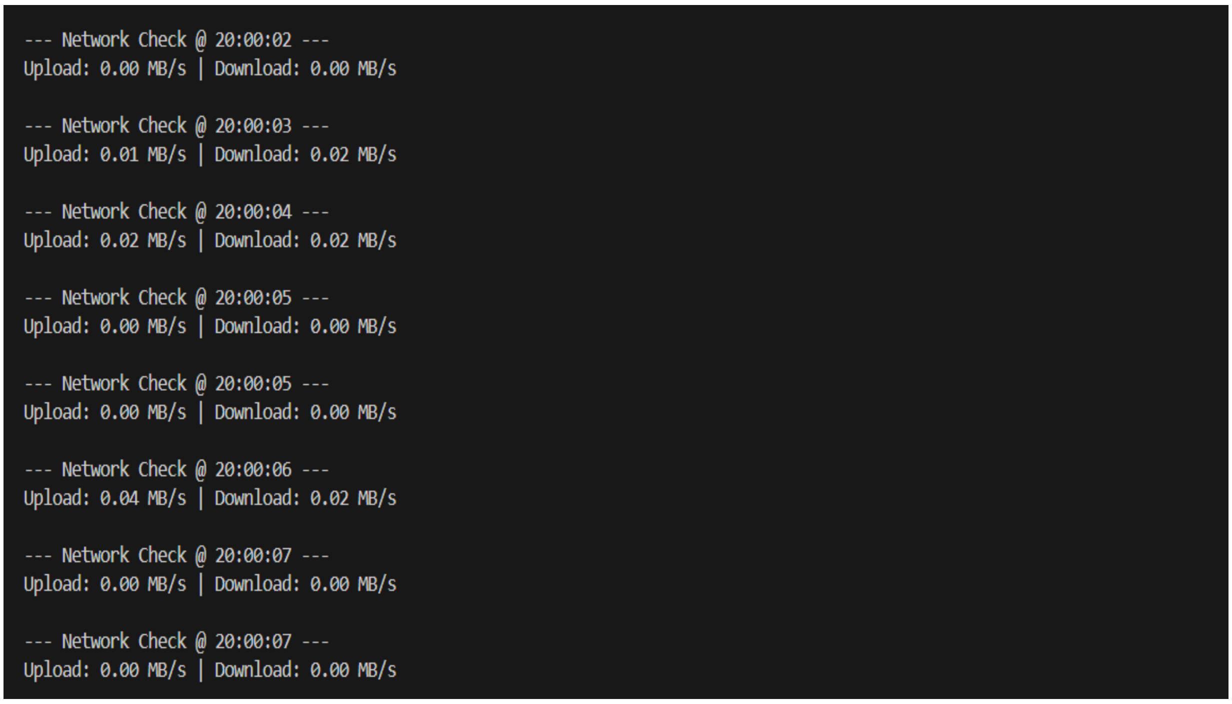


FIGURE 4.2



```
PS C:\Users\krithika\OneDrive\Desktop\cn> python test.py
✅ Live network monitor started – close the graph window to stop.

--- Network Check @ 23:36:05 ---
Upload: 0.00 MB/s | Download: 0.00 MB/s

--- Network Check @ 23:36:05 ---
Upload: 0.00 MB/s | Download: 0.00 MB/s

--- Network Check @ 23:36:06 ---
Upload: 0.00 MB/s | Download: 0.00 MB/s

--- Network Check @ 23:36:07 ---
Upload: 0.00 MB/s | Download: 0.01 MB/s

--- Network Check @ 23:36:07 ---
Upload: 0.00 MB/s | Download: 0.00 MB/s

--- Network Check @ 23:36:08 ---
Upload: 0.00 MB/s | Download: 0.00 MB/s

--- Network Check @ 23:36:09 ---
Upload: 0.00 MB/s | Download: 0.00 MB/s

--- Network Check @ 23:36:10 ---
Upload: 0.00 MB/s | Download: 0.00 MB/s
```

FIGURE 4.3



```
--- Network Check @ 20:00:02 ---
Upload: 0.00 MB/s | Download: 0.00 MB/s

--- Network Check @ 20:00:03 ---
Upload: 0.01 MB/s | Download: 0.02 MB/s

--- Network Check @ 20:00:04 ---
Upload: 0.02 MB/s | Download: 0.02 MB/s

--- Network Check @ 20:00:05 ---
Upload: 0.00 MB/s | Download: 0.00 MB/s

--- Network Check @ 20:00:05 ---
Upload: 0.00 MB/s | Download: 0.00 MB/s

--- Network Check @ 20:00:06 ---
Upload: 0.04 MB/s | Download: 0.02 MB/s

--- Network Check @ 20:00:07 ---
Upload: 0.00 MB/s | Download: 0.00 MB/s

--- Network Check @ 20:00:07 ---
Upload: 0.00 MB/s | Download: 0.00 MB/s
```

FIGURE 4.4

CHAPTER 5
CONCLUSION

This project demonstrates the development of a Python-based Network Performance Monitor (NPM) that effectively tracks key network metrics like **latency**, **bandwidth usage**, and **network availability**. The system operates in real-time, continuously measuring these metrics and alerting users when performance thresholds are breached. By using libraries such as ping3 for latency monitoring and psutil for bandwidth tracking, the NPM provides valuable insights into network performance, helping users detect and resolve issues before they cause significant disruptions. Its lightweight design and modularity make it an adaptable solution for a wide range of network environments.

One of the key achievements of this system is its ability to **generate alerts** when network performance drops below acceptable standards. Alerts based on latency spikes, bandwidth overuse, or network unavailability ensure that users are notified in real-time, enabling proactive management of network resources. This reduces the likelihood of severe network failures and improves the overall quality of service for users. Additionally, by maintaining detailed logs of network performance, the system provides historical data for diagnosing recurring issues or optimizing network configurations.

Moreover, the system is designed with future scalability in mind. The modular nature of the NPM allows for **easy integration of additional features**, such as real-time data visualization through tools like Grafana or Matplotlib, remote alerting via email or SMS, or the ability to monitor more advanced metrics like jitter, packet loss, and connection stability. These enhancements would provide a more comprehensive network monitoring solution, making it suitable for both small-scale personal use and larger enterprise applications.

In conclusion, this project successfully achieves its goal of creating a functional and adaptable Network Performance Monitor. The current system is a strong foundation for further improvements, providing essential network insights while maintaining flexibility and ease of use. By implementing additional features in the future, the NPM can become a powerful tool for network administrators, offering enhanced monitoring, alerting, and diagnostic capabilities to ensure optimal network performance and reliability.

REFERENCES

1. Psutil Documentation. (2024). psutil: Cross-platform process and system utilities for Python. Retrieved from https://psutil.readthedocs.io

2. Ping3 Documentation. (2024). ping3: Pure Python3 version of ICMP ping implementation. Retrieved from https://github.com/kyan001/ping3

3. Cisco Systems. (2023). Understanding Network Performance Metrics: Latency, Bandwidth, and Packet Loss. Cisco Technical White Paper. Retrieved from https://www.cisco.com

4. Jain, R., & Paul, S. (2013). Network virtualization and software defined networking for cloud computing: A survey. IEEE Communications Magazine, 51(11), 24–31. https://doi.org/10.1109/MCOM.2013.6658648

5. Xu, M., & Li, Z. (2019). Real-time network performance monitoring and analysis based on data visualization. International Journal of Computer Applications, 182(14), 1–5. https://doi.org/10.5120/ijca2019918609

6. Gupta, A., & Singh, D. (2021). A Review on Network Performance Monitoring Tools and Techniques. International Journal of Engineering Research & Technology (IJERT), 10(4), 234–238.

7. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communications Surveys & Tutorials, 17(4), 2347–2376. https://doi.org/10.1109/COMST.2015.2444095

8. Tanenbaum, A. S., & Wetherall, D. J. (2011). Computer Networks (5th ed.). Pearson Education.

9. Nagios Enterprises. (2022). Nagios Network Monitoring Software Documentation. Retrieved from https://www.nagios.org

10. Zabbix LLC. (2023). Zabbix: Open Source Network Monitoring Tool. Retrieved from https://www.zabbix.com