Ministry of Education
University of Jeddah
Collage of Computer Science and Engineering

جامعة جدة
University of Jeddah

Networks Project

Yara Sulaiman Almutairi 2110553
Aleen Hamad Alsulaiman 2111910

1-Introduction :

Real-time traffic analyzer and monitor project. The tool captures the packets from the network using the Scapy library and will analyze the different protocols used on the network (Ethernet, IP, TCP, UDP) and also calculate some metrics like throughput and latency. As such, the end result being explained would be to visualize/re-elaborate the network traffic data in order to understand how our devices are performing on the network.

2. Project Outline

The project can be used to monitor real time network traffic gathering important metrics like throughput (data rate), latency (packet delay) and protocol activity. To realize this goal, the system consists of several aspects that work synchronously:

Using scapy: it can capture network packets from your network and analyze some protocols by inspecting the packets to get the required information.

Data Analysis: For every packet, the system retrieves details about the source and destination addresses, packet size, and the containing protocol. The system keeps tabs on how much data has been transferred through each protocol and records throughput and latency on active TCP/UDP connections in real-time.

On Real-Time Stats: The tool periodically shows the number of unique IPs and MAC addresses observed, data throughput per protocol, and the average latency.

Visualization: System produces graphs to visualize the network performance. These consist of time-based throughput, latency distributions for TCP/UDP packets, and captured packet counts for each protocol (Ethernet, IP, TCP, UDP).

3- Functions Used

scapy. sniff() : as the packet capturing from the network The script reads each packet in real-time using a callback function: analyze_packet(), and runs until the user interrupts it.

analyze_packet(pkt) : This is invoked for each packet captured by pcap. It parses data like Ethernet, IP, TCP, and UDP headers and updates ACC traffic statistics.

update_traffic(protocol, src, dest, pkt_size, timestamp): Updates the traffic data such as data volume for each protocol, timestamps of connections for latency calculations and unique IP/MAC addresses observed.

display_stats_periodically(): Function Similarity: Display network statistics (every 30 seconds) Real-time updates on unique IP counts, unique MAC counts, throughput, and average latency.

calculate_throughput(): Given an interval, the function computes and outputs the throughput (data rate) as the output in bits per second for each protocol

calculate_latency: This function takes a series of TCP/UDP packets from a single connection and calculates the average latency in seconds on that connection by calculating the time difference between the packets in the connection.

visualize_results(): Generate matplotlib visualizations. It produces graphs of throughput over time, latency histograms, and protocol activity.

terminate() : Responsible for stopping the monitoring process when the user triggers it to stop (e.g., keyboard interrupt).

c**leanup():** Function that saves logs and creates the plot after monitoring, and exits the program.

4-Attributes:

data_volume: Monitors the total data volume (in bytes) for every network protocol

connection_timestamps: The start times of the connections for TCP/UDP traffic to compute latency.

observed_ips and observed_macs: Contains unique IP and MAC Addresses seen while sniffing traffic.

packet_details : Stores the size of packets for each protocol.

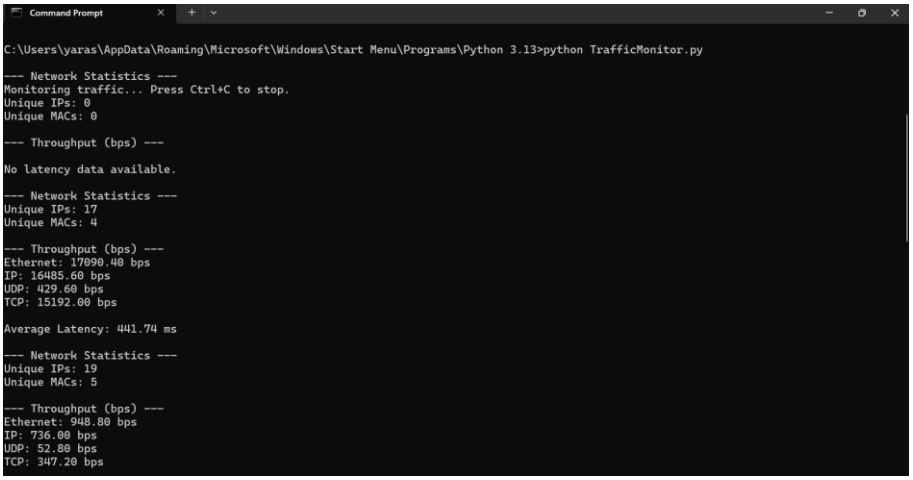
throughput_tracker: Tracks throughput over time per protocol

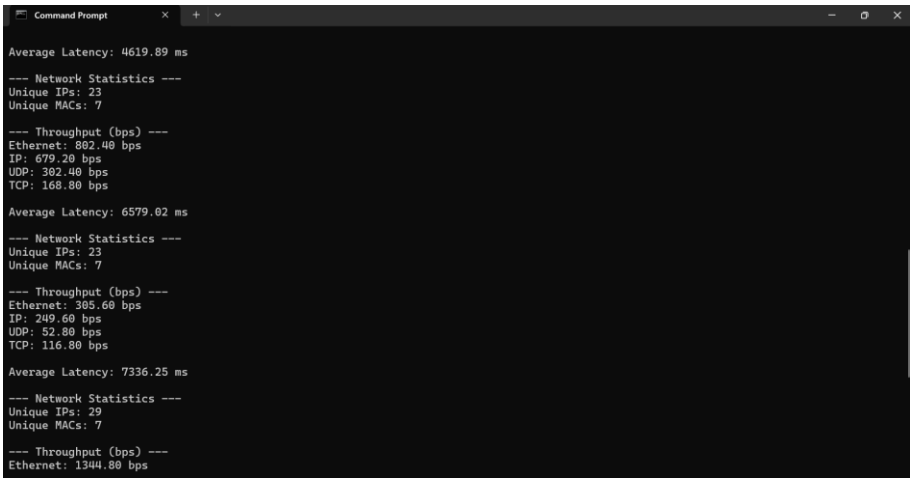
delay_records: This keeps track of latencies in TCP/UDP connection.

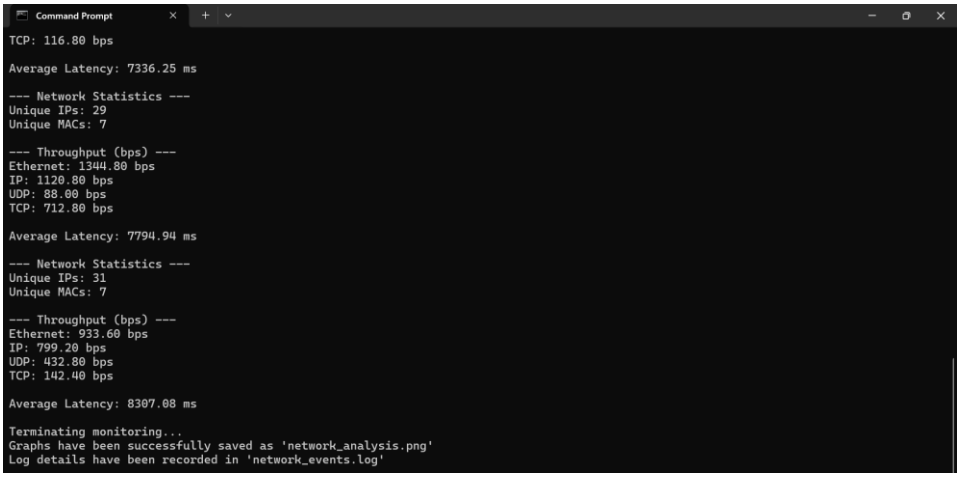
protocol_activity: Counts the amount of packets captured for each protocol

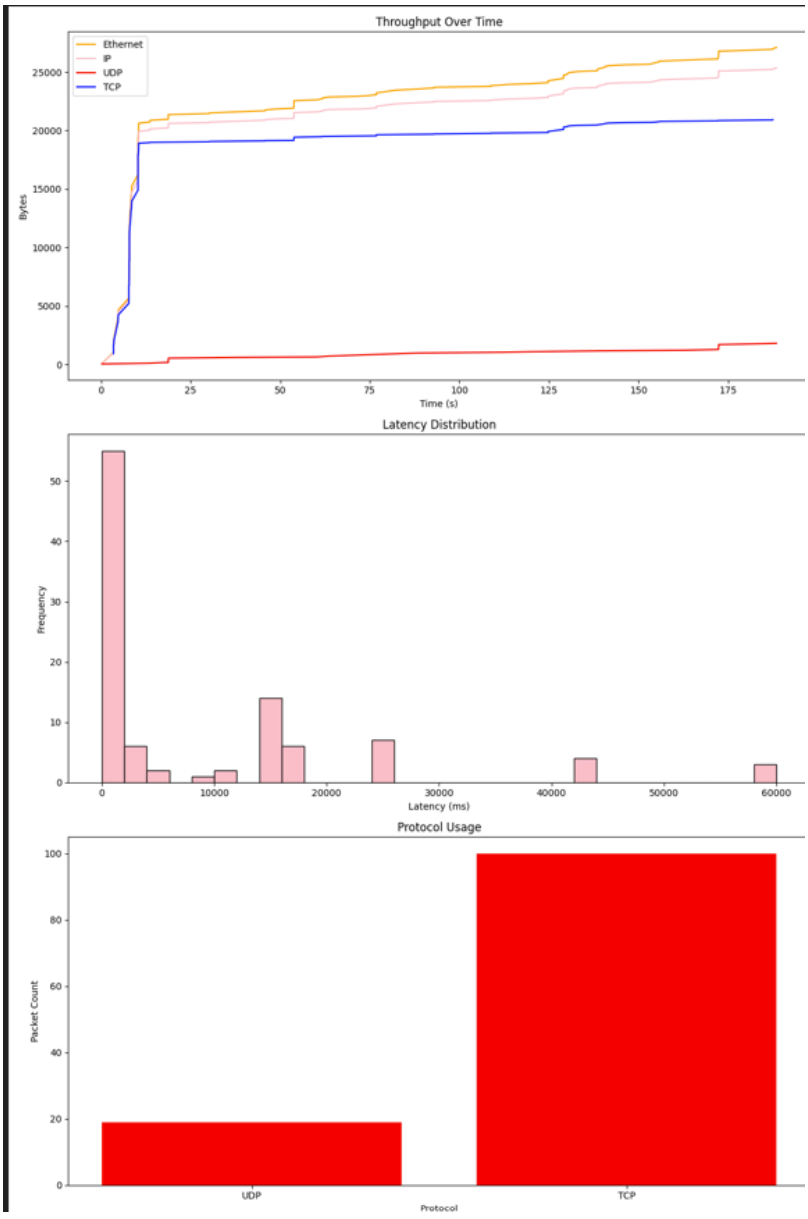
5-Results :

Running the code on the first device :

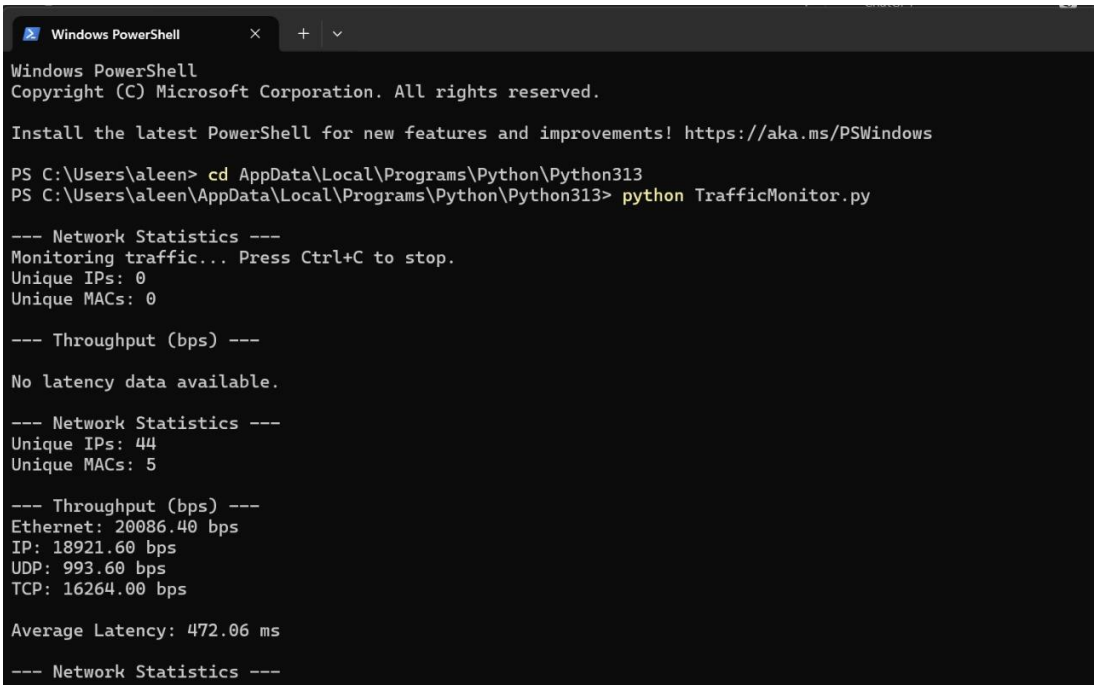


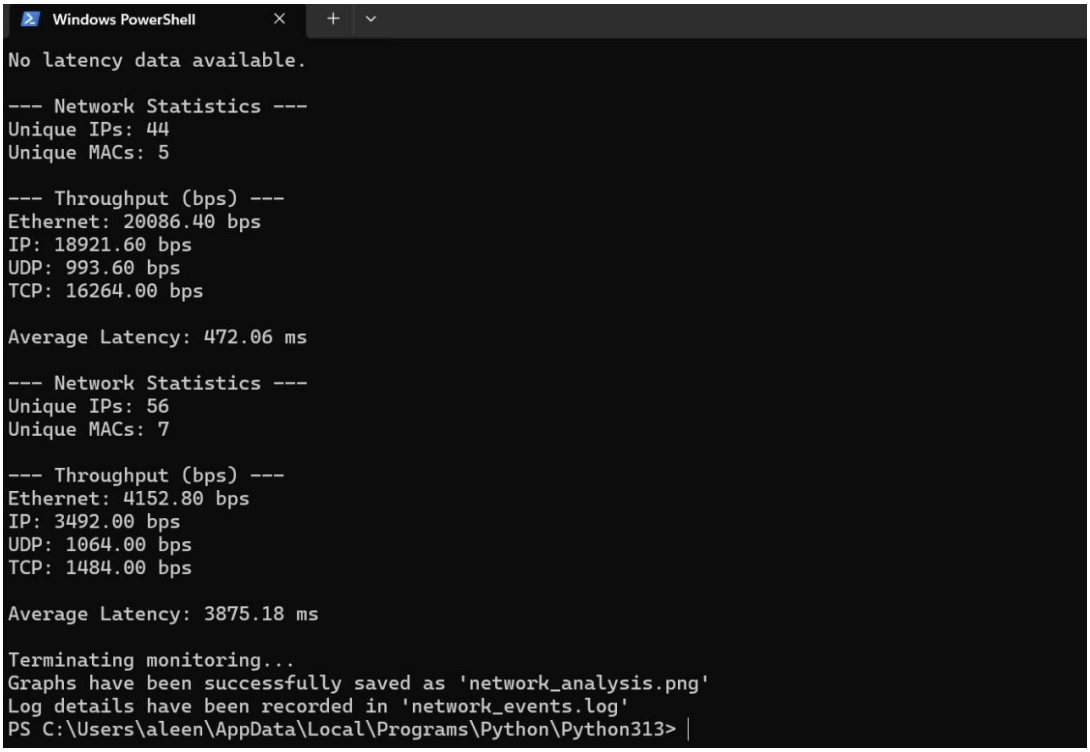


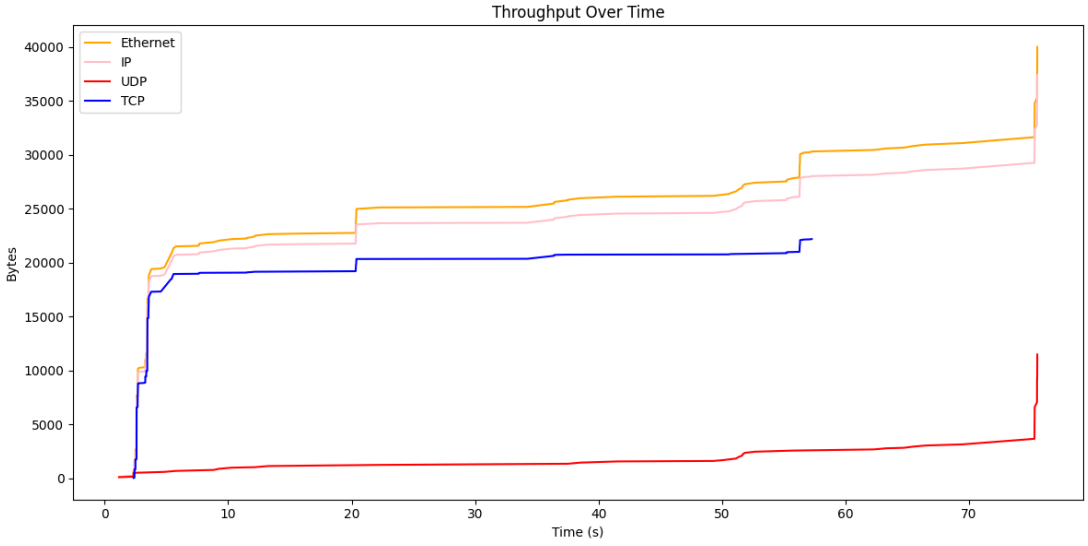


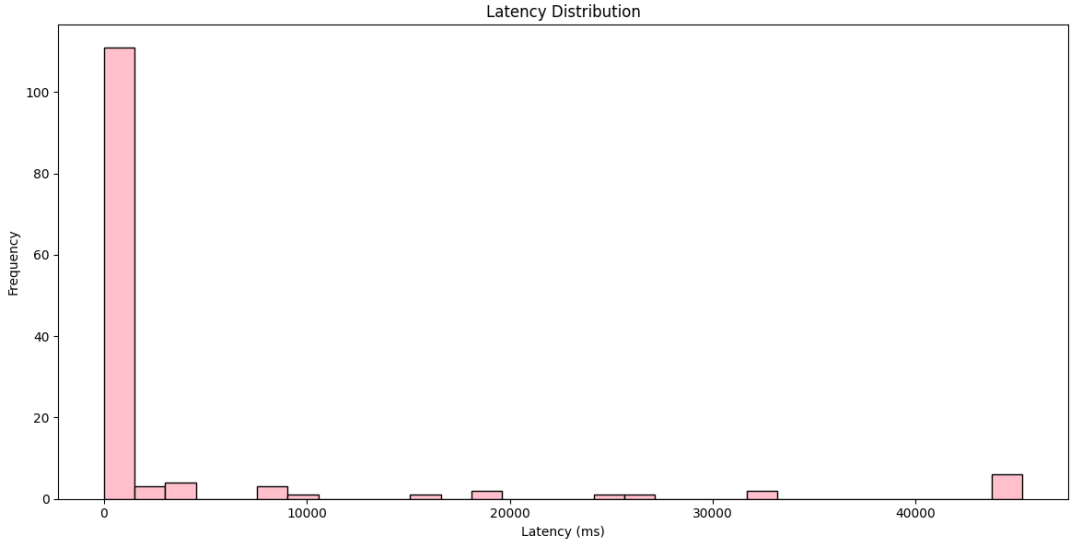

Throughput Over Time
Ethernet
IP
UDP
TCP
Bytes
Time (s)
Latency Distribution
Frequency
Latency (ms)
Protocol Usage
Packet Count
UDP
TCP
Protocol

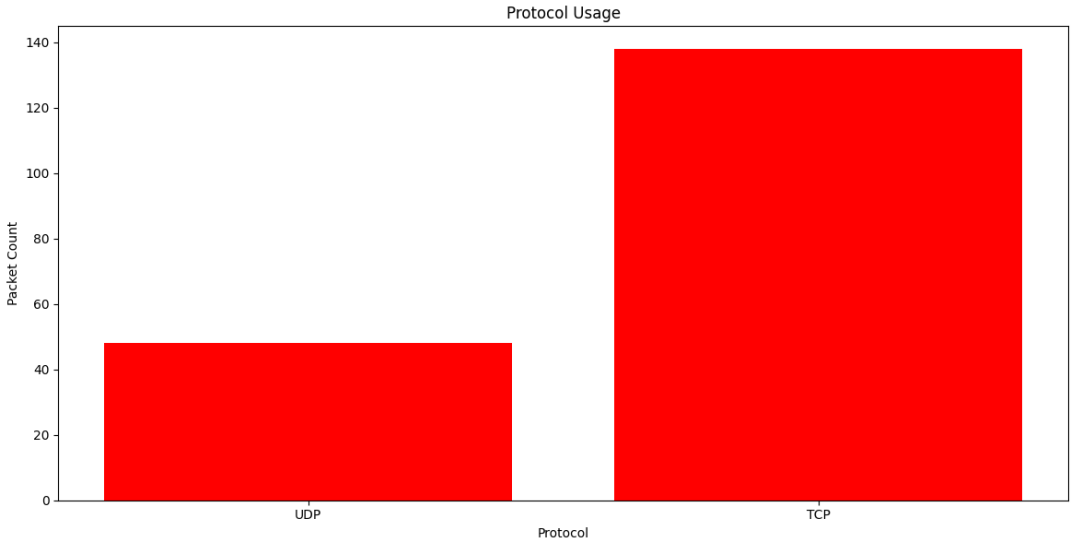
2-Running the code on the second device :






Throughput Over Time
Ethernet
IP
UDP
TCP
Bytes
Time (s)


Latency Distribution
Frequency
Latency (ms)


Protocol Usage
Packet Count
UDP
TCP
Protocol

Visualizations:

Throughput Graph: This graph displayed the cumulative amount of data transferred based on different protocols used (Ethernet/IP/TCP/UDP) over time. This visualization served the purpose of pinpointing data-heavy intervals.

Latency Histogram: This showed the latencies of TCP connection packets between its packets. The most of the connections went through lowest latencies as evident from the histogram, but few of them spiked during peak traffic times.

Protocol Activity: The bar chart displayed the number of packets captured for each protocol. It was observed that the TCP protocol accounted for a significant portion of the captured traffic, followed by UDP and Ethernet.

6. Discussion on the Network Environment

The network monitoring tool was tested on a local network environment consisting of two laptops connected to different Wi-Fi and wireless router . No wired Ethernet connections were used during the testing.

When the testing on both laptops many applications were run to stimulate network traffic include web browsing and video streaming. All of which generated a mix of TCP and UDP traffic. The network environment was stable, but due to the nature of Wi-Fi connections, there were fluctuations in both throughput and latency, particularly when the network was under heavier usage.

While testing this the following highlights were noted :

- **Protocol Activity:** showed that the TCP protocol was in very high use and significantly increase because of the applications including web browsing and file transfers. UDP traffic also increased but less than the tcp .
- **Latency outliers:** Latency was stable in low traffic and saw occasional spikes in high traffic. This happened by networks or with large file transfers.
- **Throughput Monitoring:** The throughput of all protocols was calculated correctly. The tool also identified periods of network congestion, in which throughput for specific protocols dropped as data saturation occurred.

In conclusion, the network monitoring tool worked well on our Wi-Fi network, providing useful data on throughput and latency. The Tool still gave a clear picture of how the network was performing. These results reflect the challenges of using a wireless network, and the performance would likely be more consistent with a wired connection or in a more stable network setup.