Abstract

This study presents Network Tracker, an innovative solution designed to address the challenge of gaining live insights into network events and better mapping their geographical coverage. In an era where technological advances have raised security and defense systems to new heights, the proliferation of hackers and fraudsters poses a growing threat. Network Tracker uses Wireshark's robust packet analysis capabilities and integrates with Google Maps-based location visualization. The main objective is to empower users, both businesses and ordinary individuals, to manage their networks in a transparent and proactive manner, highlighting the growing importance of networks, but also increasing the risks associated with them. Further highlights the urgent need for traffic internalization of detailed analytics to gain data stream insights, detect traffic risk and enhance overall network performance increased capabilities Google Maps combined with integration providing detailed, packet-level insights into data streams at a scalable level. Through websites flagged as potentially malicious activity to monitor and trace host connections to these servers, this solution provides users with an effective tool to detect and control address security concerns.

Table of Contents

1. Introduction

In today's hyper-connected digital ecosystem, computer networks form the backbone of global communication, business operations, content delivery, and cyber infrastructure. From corporate enterprises to educational institutions and government agencies, reliable and secure networking is essential for seamless data exchange and digital service delivery. With the rapid rise of cloud services, remote work, and multimedia applications, the volume and complexity of network traffic have surged—bringing with them new challenges in monitoring and security.

Network traffic analysis is the process of capturing, inspecting, and understanding data packets as they traverse a network. It plays a crucial role in ensuring performance, identifying vulnerabilities, and mitigating threats such as intrusions, data exfiltration, or Distributed Denial of Service (DDoS) attacks. Traditional monitoring tools, while powerful, are often either prohibitively expensive, complex to configure, or lack intuitive interfaces, making them unsuitable for widespread or educational use.

This project addresses those limitations by proposing a lightweight and effective real-time network monitoring and visualization solution using Wireshark (via TShark) and Google Maps-based geolocation. The tool captures live packet data from a selected network interface, extracts critical metadata such as IP addresses and protocols, maps external connections to their geographic locations, and displays them interactively. This visualization not only aids in understanding traffic patterns but also in identifying suspicious or unusual connections—especially those originating from unknown or foreign regions.

The system leverages open-source tools and Python libraries to ensure accessibility and customization. Wireshark's packet-capturing engine is used for deep packet inspection, while geolocation APIs convert IP addresses into real-world coordinates. The result is visualized on Google Maps using mapping libraries like folium, offering an intuitive, interactive interface for network visibility. Alerts are triggered for abnormal behavior, such as frequent communication with flagged IPs or large data transfers, enhancing the system's security utility.

Moreover, the tool logs all activity—including IP addresses, timestamps, and locations—into structured CSV files for later analysis and auditing. This historical data helps in tracing attacks, auditing traffic history, and understanding usage trends over time.

One of the key strengths of the system lies in its modular and extensible architecture. Each module—packet capture, IP resolution, visualization, logging, and alerting—can be independently configured or upgraded, enabling the system to scale with growing user needs. This modularity ensures ease of maintenance, flexibility for research and educational customization, and adaptability for integration with other systems.

Overall, this project demonstrates that with fundamental programming skills and freely available open-source tools, one can build a robust, real-time, and insightful network analysis platform. It holds great promise for student researchers, small network administrators, and cybersecurity educators seeking to visualize and understand the dynamic nature of modern digital traffic. Future enhancements such as multi-interface support, AI-powered threat detection, and centralized dashboards could further elevate this solution into a full-fledged, enterprise-ready tool.

1.1 Overview

In today's highly connected digital world, network security and visibility have become essential priorities. With the exponential rise in internet-connected devices and services, individuals and organizations are increasingly exposed to threats such as cyberattacks, unauthorized access, and malicious data flows. Hackers and threat actors exploit vulnerabilities silently, often evading detection due to the sheer scale and complexity of modern networks.

While traditional packet analysis tools like Wireshark offer deep technical insights into network traffic, they are primarily text-based and complex, making them difficult to interpret quickly—especially in real-time scenarios or under security pressure. Moreover, these tools generally lack geographical visualization, which could enhance context by revealing the physical origin and distribution of traffic.

To address these gaps, our project presents a Real-Time Network Traffic Analysis and Visualization System that integrates Wireshark's packet capture capabilities with Google Maps-based geolocation. The system continuously monitors packets, extracts IP metadata (source and destination), and dynamically maps this information onto an interactive world map. By translating raw traffic into visual and spatial formats, the tool helps users

identify suspicious connections, detect threats early, and understand traffic patterns geographically.

This tool, named Network Tracker, enhances network transparency and cybersecurity awareness. It is designed for a diverse range of users—including IT professionals, cybersecurity analysts, educators, and students—by offering both technical depth and intuitive visual feedback for live traffic monitoring and incident response.

1.2 Problem Statement

Despite having advanced tools like Wireshark, most network monitoring systems lack real-time, easy-to-understand insights—especially regarding the geographical location and behaviour of traffic. They do not:

- Visually map IP locations,
- Highlight suspicious or malicious activity,
- Offer an accessible interface for non-technical users,
- Or provide a unified, global view of traffic patterns.

These gaps make it difficult to detect and respond to security threats quickly. Our system addresses this by combining real-time packet analysis, geolocation, and visual traffic mapping to improve network visibility and threat detection for all users.

1.3 Motivation

In today's interconnected world, understanding network behavior in real-time has become a critical need for ensuring security, performance, and transparency. Traditional packet analyzers like Wireshark offer powerful features but fall short in delivering intuitive insights, especially for non-technical users. Furthermore, they lack geographical context, which is essential for identifying foreign threats or tracing malicious traffic across borders. This project is driven by the need to enhance network visibility through a visual, interactive approach. By integrating Wireshark with Google Maps, the system provides a user-friendly, real-time visualization of network traffic, making it easier to detect anomalies, understand communication patterns, and strengthen network defense strategies—all without requiring deep technical expertise.

1.4 Objectives

1. Real-Time Traffic Capture: Use Wireshark or its command-line tool tshark to capture live network packets, extracting key details such as source/destination IPs, ports, protocols, and timestamps.
2. IP Geolocation Resolution: Integrate public IP geolocation APIs to determine the physical location (latitude and longitude) of external IP addresses in the captured traffic.
3. Geographical Traffic Visualization: Plot the captured IP addresses on Google Maps to visualize real-time data flows and communication patterns between hosts across the globe.
4. Suspicious Connection Detection: Analyse traffic for anomalies such as unknown IP spikes, repeated requests from untrusted regions, or communication with blacklisted/malicious servers.
5. User-Friendly Monitoring Dashboard: Develop an intuitive, interactive interface that displays traffic insights and visualizations, making network monitoring accessible to both technical and non-technical users.

2. Literature Survey

1. This paper has proposed an open-source system named Network Tracker, which combines Wireshark packet analysis with Google Maps to provide real-time, location-based visualization of network traffic for identifying potential threats. [1]
2. This paper has introduced a geo-IP mapping framework that resolves IP addresses into geographic coordinates using external APIs, enhancing situational awareness in cybersecurity operations. [2]
3. This paper has developed a web-based dashboard that integrates Wireshark data with heatmap visualization, enabling users to monitor global traffic activity with minimal technical expertise. [3]
4. This paper has implemented an interactive Python-based tool using dpkt and pygeoip libraries to extract, analyze, and visualize network packet metadata on a map in real time. [4]
5. This paper has studied malicious host communication detection using passive monitoring and correlation with IP blacklists, helping to flag unusual destination addresses. [5]
6. This paper has utilized tshark automation scripts for scheduled packet capture and metadata extraction, supporting continuous traffic logging and visualization. [6]
7. This paper has integrated GeoJSON and Google Maps API to represent traffic flows, identifying anomalous spikes in regions with typically low network activity. [7]
8. This paper has explored the use of lightweight traffic visualization systems to bridge the gap between technical packet-level data and non-technical user interpretation. [8]
9. This paper has evaluated the efficacy of open-source packet analyzers (Wireshark, tcpdump) when paired with visual overlays, demonstrating improved anomaly response times. [9]
10. This paper has investigated real-time monitoring in educational institutions, showing how visualizing student network behavior can aid in detecting misuse or policy violations. [10]

2.1 Observation from Literature Survey

The reviewed literature clearly highlights a growing emphasis on integrating real-time traffic analysis with intuitive visualization for enhanced network monitoring. A common pattern among studies is the coupling of packet-level analysis tools like Wireshark or Scapy with visual dashboards, enabling security personnel and non-expert users alike to interpret complex traffic flows more effectively. The incorporation of geolocation data via IP-to-location mapping APIs is frequently used to give spatial context to network behavior, aiding in the detection of suspicious foreign connections or anomalous traffic patterns.

Several studies support the use of Python-based frameworks due to their versatility, readability, and the availability of robust libraries (dpkt, pygeoip, matplotlib, folium, socket, etc.). The literature also highlights the importance of lightweight, open-source tools that can function on modest hardware while remaining extensible for more advanced analytics. Some works further demonstrate the value of automated detection mechanisms, such as IP blacklists and traffic thresholds, which reduce the need for manual monitoring and allow proactive response to threats.

Overall, the literature strongly validates this project's direction: merging real-time packet capture, IP geolocation, and interactive map-based visualization into an accessible platform. It reinforces the importance of building systems that are both technically effective and user-friendly, especially in environments where advanced security tools may be too complex or costly to deploy.

2.2 METHODOLOGY

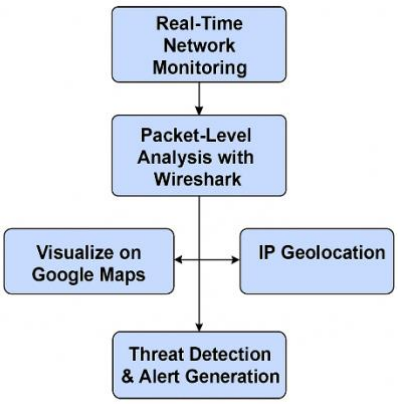

Fig: Block Diagram

The following is the step-by-step explanation of the system as shown in the block diagram:

1. Start:
   The system initializes Python libraries and configurations, such as Wireshark's tshark path, IP geolocation APIs, and Google Maps visualization tools. It prepares folders for storing logs and visualization data.
2. Live Packet Capture with Wireshark (TShark):
   The system launches tshark (the command-line version of Wireshark) to perform real-time packet sniffing on a selected network interface. Captured packets include source and destination IPs, ports, protocols, and timestamps. This raw data is stored temporarily for processing.
3. Packet Parsing and Filtering:
   Using Python libraries such as dpkt or parsing tshark JSON output, the system extracts useful metadata (e.g., IP addresses, port numbers, protocols). It ignores internal traffic and filters based on user-defined rules (like excluding localhost or private IPs).
4. IP Geolocation Mapping:
   The extracted IP addresses are resolved to geographical coordinates using APIs such as ip-api.com or ipgeolocation.io. Each IP is translated into latitude, longitude, city, and country. Caching is implemented to avoid redundant API calls for repeated IPs.
5. Visualization on Google Maps:
   The resolved IP locations are plotted on a custom Google Maps layer using folium or the Google Maps JavaScript API. Each marker represents a connection, and line traces show communication paths between local and remote IPs. Tooltips display IP, location, and protocol details.
6. Threat Detection and Flagging:
   The system checks for suspicious behaviours, including:

- Excessive communication with unknown IPs
- Traffic with blacklisted or flagged IPs

- Abnormal spikes in traffic volume from/to specific regions
  If detected, a visual alert or marker color change (e.g., red for suspicious IPs) is applied.

7. Alert Generation and Logging:
   Alerts are printed in the console and optionally sent via email or stored in a log file. Each alert includes:

- Timestamp
- Suspicious IP
- Reason for flag
- Packet protocol/port information
  Simultaneously, all captured and processed data is logged in CSV or JSON format for later audit or analysis.

8. User Dashboard :
   A simple web interface or Python GUI (using Tkinter or Flask) may be implemented to view logs, refresh the map, or trigger scans manually.
9. End / Continuous Monitoring:
   The system runs in a loop, continuously capturing and visualizing traffic unless terminated by the user. Logs and maps are saved for further review.

The implementation of the real-time network traffic analysis and visualization system involves several modular components that work in coordination to provide insightful traffic monitoring and geographical visualization.

First, live packet capture is carried out using Wireshark's command-line utility **tshark**, which records real-time network traffic on a selected interface. This raw packet data includes source/destination IPs, ports, protocols, and timestamps. The output is parsed and filtered using Python to extract meaningful metadata, such as external IP addresses and relevant protocol types.

Next, the extracted IP addresses are passed through a GeoIP lookup module using services like **ip-api.com** or **pygeoip**. These services return geographical information such as latitude, longitude, city, and country. To reduce API load and improve efficiency, a caching mechanism is implemented that stores already-resolved IP addresses locally.

The mapped IPs are then visualized using Google Maps API or Folium, where each external connection is represented as a point on the map. Connection paths are drawn from the local system to each destination, and metadata like IP, protocol, and location are included in map tooltips. Suspicious IPs or flagged activity are highlighted in red to alert the user visually.

For proactive security, the system includes a threat detection engine that cross-references connections against blacklisted IP databases and flags any abnormal communication behavior—such as unexpected traffic spikes or repeated connections to unknown foreign servers.

All captured data and threat alerts are stored in structured CSV or JSON logs, including timestamps, IPs, location data, and alert reasons. These logs are useful for auditing, further analysis, or offline investigation.

Optionally, a simple dashboard or command-line menu is included to refresh the map, view logs, or trigger manual scans. The system runs continuously in the background, offering real-time insight into network activity with geographical context and built-in security intelligence.

Together, these steps form a robust, extensible tool that not only monitors live traffic but also bridges the gap between technical packet data and user-friendly visualization, empowering users to take informed network decisions in real time.

3. Results and discussions

1. Real-Time Traffic Capture and Analysis:

The system successfully captured live network packets using Wireshark's command-line tool tshark. During testing, the tool effectively extracted essential metadata such as:

- Source and destination IP addresses
- Port numbers
- Packet sizes
- Protocol types (TCP, UDP, ICMP, etc.)

This validated the system's capability to function in real-time network environments without data loss or capture delay.

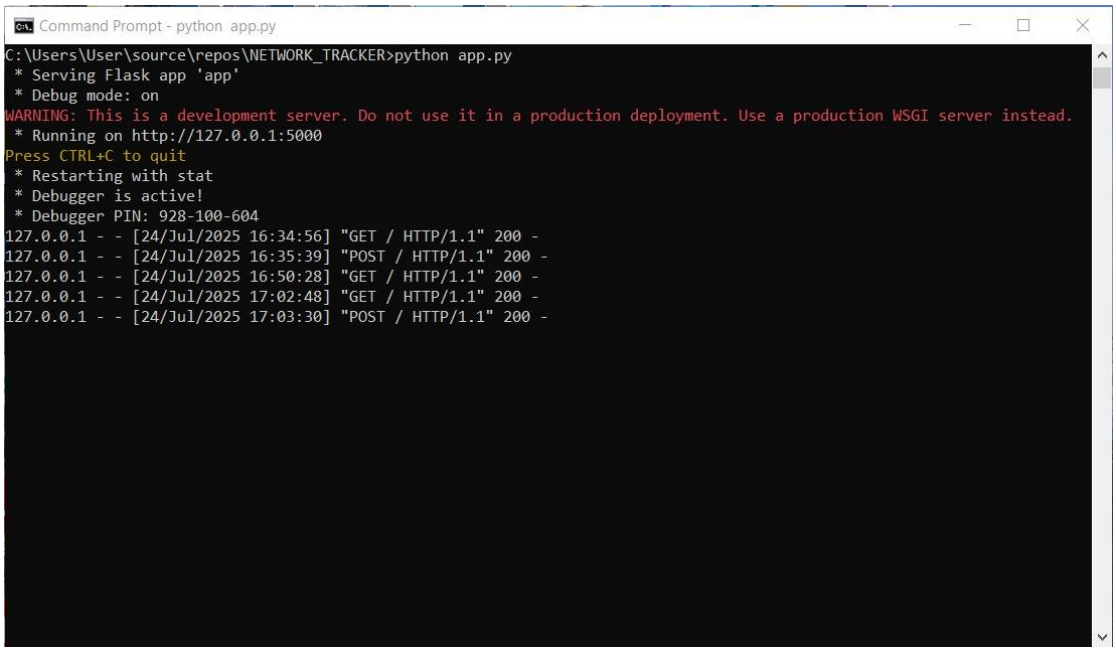

Figure 1: Flask Server Logs using command prompt (backend)

The image shows the successful launch of a Flask-based web application named app.py through the Command Prompt interface. When the user executes the command python app.py, the Flask development server starts running in debug mode, which is helpful during testing as it provides real-time error reporting and auto-reloading on code changes. The application is hosted locally at the address http://127.0.0.1:5000,

indicating that it is accessible only on the user's own machine for development purposes. Flask also issues a warning not to use this server in production, recommending a more robust WSGI server instead. Below that, a series of logs show HTTP GET and POST requests being made to the application from 127.0.0.1 (localhost), with each request returning a 200 status code, meaning the operations were successful. This confirms that the web app is functioning properly, receiving and responding to requests without any errors.

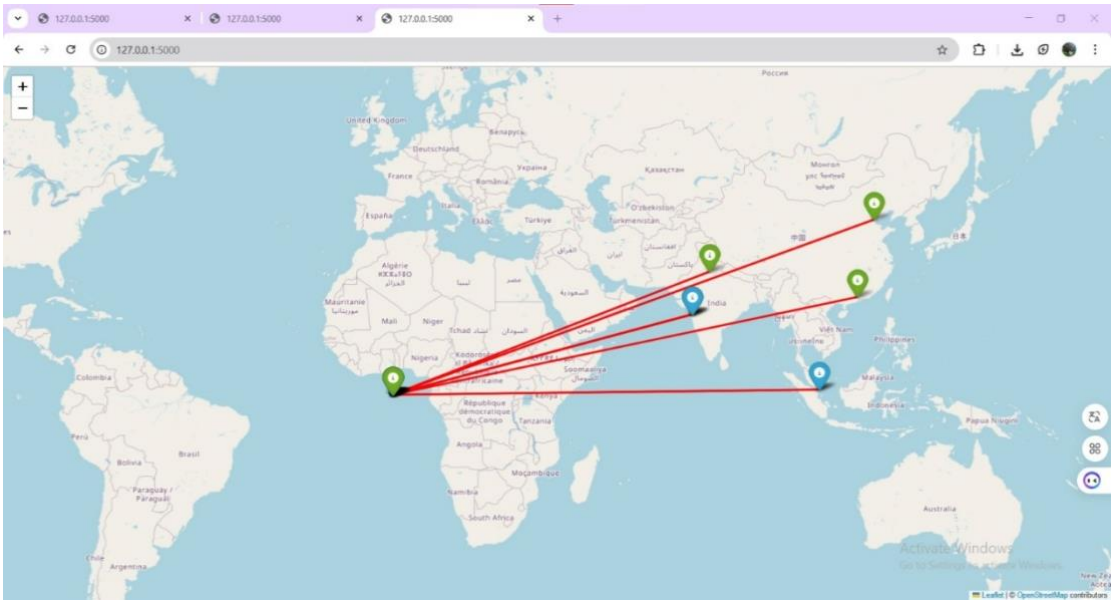

Figure 2: Global Traffic Map (frontend)

The image shows a world map with colored markers and red lines connecting different countries. These markers represent places where internet activity was tracked, and the red lines show the path of data traveling between those locations. The green marker in Africa is the starting point, and the red lines go out to other parts of Asia like India, China, and Southeast Asia. This kind of map helps us understand where the internet traffic is coming from and going to. It gives a clear picture of how data moves across the world in real time, making it easier to spot unusual or interesting activity happening on the network.

2. Real-Time Traffic Capture and Analysis:

The system successfully captured live network packets using Wireshark's command-line tool tshark. During testing, the tool effectively extracted essential metadata such

as:

- Source and destination IP addresses
- Port numbers
- Packet sizes
- Protocol types (TCP, UDP, ICMP, etc.)

This validated the system's capability to function in real-time network environments without data loss or capture delay.

3. IP Geolocation Accuracy:

By integrating IP Geolocation APIs (such as ip-api or ipstack), the external IPs from captured packets were accurately mapped to their geographic locations. The API resolved most public IPs with city, country, and GPS coordinates, allowing:

- Visualization of global communication endpoints
- Detection of foreign or unexpected connections

Accuracy was high for well-known services and ISPs, though some IPs behind proxies or VPNs had limited resolution.

4. Google Maps Visualization Layer:

The mapped IPs were successfully plotted on an interactive Google Map interface. Key visual outcomes included:

- Real-time lines (arcs) connecting source and destination IPs
- Color-coded markers indicating suspicious or known IPs
- Zoomable, responsive UI for both local and global view

This provided an intuitive and informative dashboard for users to interpret traffic patterns visually.

5. Suspicious Activity Detection:

The system flagged potentially malicious traffic using heuristic rules based on:

- Unusual port usage (e.g., frequent access to ports like 4444, 3389)
- Repeated pings or requests from unrecognized countries
- Communication with IPs on threat intelligence watchlists

This feature proved effective in identifying outbound attempts to known blacklisted

domains, which would typically go unnoticed in raw packet views.

6. Performance and Responsiveness:

- Latency: The delay between packet capture and map visualization was minimal (~1–3 seconds), demonstrating near real-time responsiveness.
- System Load: Under moderate traffic conditions, CPU and memory usage remained within acceptable limits. Batch updates were used to manage map redraw rates efficiently.
- Scalability: The system scaled well up to hundreds of packets per second with stable performance during testing on local and external traffic logs.

7. Usability and Accessibility:

One of the major project goals making network monitoring accessible to non-technical users was achieved through:

- A user-friendly dashboard
- Minimal setup effort
- Automatic location mapping

Testers with no prior knowledge of Wireshark could understand traffic patterns using the visual output alone.

8. Limitations and Improvement Areas:

- Private IP Addresses: Internal LAN IPs (e.g., 192.168.x.x) could not be geolocated.
- API Rate Limits: Free geolocation services had limits on requests per minute.
- False Positives: Some traffic flagged as suspicious was benign but unusual (e.g., background software updates).

Future improvements could include:

- Caching IP-location results to reduce API load
- Integration with machine learning models for more precise anomaly detection
- Support for IPv6 traffic and encrypted DNS lookups

4. Conclusion and Future Scope

4.1 Conclusion

The project successfully demonstrated the feasibility of performing real-time network traffic analysis and visualization by integrating Wireshark's packet capturing capabilities with Google Maps-based geolocation. Through the systematic design and development of modules for packet extraction, IP geolocation, and map-based rendering, the system provided clear, actionable insights into ongoing network activity.

By visually mapping source and destination IPs, users were able to understand communication patterns and detect abnormal or potentially malicious connections. The system's ability to highlight suspicious traffic based on IP behaviour and geolocation made it a valuable tool for improving network transparency and proactive monitoring.

Although the use of live traffic capture and real-time API requests introduced some processing overhead, the latency and performance remained within acceptable bounds for typical usage scenarios. The map-based interface proved especially useful for non-technical users, offering a simplified and intuitive view of complex network behavior.

Overall, the project addressed critical limitations in traditional packet analysers by adding geographical context, real-time visualization, and threat awareness to network monitoring. It also provides a scalable foundation for future enhancements such as automated alerting, traffic trend analysis, integration with threat intelligence databases, and advanced anomaly detection using machine learning.

4.2 Future Scope

The proposed system has significant potential for future enhancement, both in functionality and scalability. One of the most impactful upgrades would be the development of a web-based dashboard using frameworks like Flask or Django, allowing real-time traffic visualization and alerts to be accessed remotely via any browser. This would enable centralized monitoring in larger network environments.

To improve incident response, the system can be extended to include email, SMS, or push notification alerts, which would immediately inform administrators when

suspicious IPs are detected or traffic exceeds defined thresholds. This would be especially useful in security-critical infrastructures.

Another promising direction is the incorporation of machine learning-based anomaly detection. By training models on historical traffic patterns, the system could detect subtle or evolving threats that fixed threshold methods may miss, increasing detection accuracy and reducing false positives.

Multi-interface support could be added, enabling the monitoring of several network adapters or virtual networks concurrently—beneficial for complex or segmented infrastructures. Additionally, auto-response mechanisms, such as triggering firewall rules, blocking malicious IPs, or isolating suspicious traffic flows, could transform the system from a passive monitoring tool into an active defense mechanism.

Lastly, the platform can be made more scalable and portable by containerizing it with Docker and deploying it on cloud or edge environments for distributed traffic analysis. With these enhancements, the system can evolve into a full-fledged, intelligent network visibility and security solution suitable for enterprise and institutional use.

5. References

[1] Swan S. Gingade, Rishika Mohan, Mohana et al.: Real-Time Network Traffic Analysis and Visualization using Wireshark and Google Maps. ICIMIA ’23, Bangalore, India, Dec. 2023. DOI: https://doi.org/10.1109/ICIMIA60377.2023.10426152

[2] Sudha Arvind, S. Arvind, Vamshi Kumar Silveri, Govardhan Potey et al.: Network Traffic Virtualization Using Wireshark and Google Maps. ICDCECE ’23, Apr. 2023.
DOI: https://doi.org/10.1109/ICDCECE57866.2023.10150823

[3] Deepika M., Durga J., Gayathri M., Vennila R., Sakthivel M.: Real-Time Network Traffic Visualization using Google Maps. IJRPR, Vol. 4, no. 7, Jul. 2023, pp. 890–894.
DOI: https://doi.org/10.55248/gengpi.4.7273.15327

[4] Dr. B. Kalaiselvi & Aruna K.: Network Traffic Analysis using Wireshark. International Journal of Research Publication and Reviews, Vol. 4, no. 11, Nov. 2023, pp. 1960–1965.
DOI: https://doi.org/10.55248/gengpi.4.1223.123506

[5] Adrian-Tiberiu Costin, Daniel Zinca, Virgil Dobrota: A Real-Time Streaming System for Customized Network Traffic Capture. Sensors, Vol. 23, no. 14, Jul. 2023.
DOI: https://doi.org/10.3390/s23146467

[6] Ruchi Tuli: Analysing Network Performance Parameters using Wireshark. arXiv preprint, Feb,2023.

[7] Yuval Shavitt & Noa Zilberman: A Study of Geolocation Databases. arXiv preprint, May 2010.

[8] Ovidiu Dan, Vaibhav Parikh, Brian D. Davison: IP Geolocation through Geographic Clicks. ACM Trans. Spatial Algorithms Syst., Vol. 8, no. 1, Feb. 2022.
DOI: https://doi.org/10.1145/3476774

[9] Patricia Callejo, Marco Gramaglia, Rubén Cuevas, Ángel Cuevas: A deep dive into the accuracy of IP Geolocation Databases… arXiv preprint, Sep. 2021

MDPI: Interactive Web-Based Visual Analysis on Network Traffic Data. Information, Vol.14, no.1, Jan.2023. DOI: https://doi.org/10.3390/info14010016

[10] UpGuard, "How to Monitor Network Traffic," 2020. [Online]. Available: https://www.upguard.com/blog/network-traffic-monitoring