A Seminar Report on

Title_of_work_carried_out

Submitted in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

by

G DINESH(231AI010)
VINAY(231AI043)
IV Sem B.Tech (AI)

Department of Information Technology

National Institute of Technology Karnataka, Surathkal.

April 2025

CERTIFICATE

This is to certify that the seminar entitled "**Network Traffic Monitoring**" has been presented by **G DINESH(231AI010),VINAY(231AI043)**student of **IV semester B.Tech (IT) /B.Tech (AI)**, Department of Information Technology, National Institute of Technology Karnataka, Surathkal, during the even semester of the academic year **2024 – 2025.** It is submitted to the Department in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology.

Place: _____

Date: (Signature of the Examiner1)

Place: _____

Date: (Signature of the Examiner2)

Place: _____

Date: (Signature of the Coordinator)

DECLARATION BY THE STUDENT

I hereby declare that the Seminar (IT290) entitled "**Network Traffic Monitoring"** was carried out by me during the even semester of the academic year 2024 – 2025 and submitted to the department of IT, in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in the department of Information Technology, is a bonafide report of the work carried out by me. The material contained in this seminar report has not been submitted to any University or Institution for the award of any degree.

Place: _____

Date: _____ _____

(Name and Signature of the Student)

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

In the modern digital world, networks form the backbone of communication between systems. Monitoring network traffic has become a critical task to ensure system performance, security, and reliability. Network traffic monitoring involves capturing, analyzing, and visualizing the flow of data packets across a network. By observing these patterns, administrators can optimize bandwidth usage, detect security threats, and troubleshoot network issues efficiently. As networks grow larger and more complex, the need for intelligent and real-time monitoring solutions has become increasingly vital.

Operating systems (OS) play a crucial role in managing network communication. They handle network interface configuration, packet routing, firewall enforcement, and scheduling of network tasks. Traditional OS-based network monitoring tools, however, often face limitations such as high resource overhead, delayed anomaly detection, and reactive instead of proactive threat responses. These challenges highlight the necessity for more efficient, lightweight, and real-time monitoring mechanisms at the OS level. Technologies like kernel-based packet filtering (e.g., eBPF in Linux) are promising solutions to minimize CPU load while providing deep insights into network traffic.

This project focuses on designing and implementing an OS-level network traffic monitoring system that balances performance, security, and real-time analysis. By integrating packet capture, intelligent traffic filtering, and anomaly detection into a single workflow, the proposed system aims to overcome the limitations of existing solutions. Through the use of efficient system resources and proactive security measures, this work enhances the ability of an operating system to maintain network stability and defend against evolving cybersecurity threats.

OBJECTIVES

The methodology adopted for this project involves the use of Wireshark on MacOS to capture, analyze, and monitor live network traffic in real time. Wireshark, a widely-used network protocol analyzer, enables deep inspection of hundreds of protocols and provides powerful filtering and visualization capabilities for network traffic analysis.

The first step was to research MacOS-based network monitoring techniques and become familiar with Wireshark's packet capture and analysis features. Installation and configuration of Wireshark were carried out on a MacOS environment, ensuring necessary permissions and network interface selections were correctly set to enable packet capturing.

Once Wireshark was set up, live network traffic was captured by monitoring selected network interfaces such as Wi-Fi (en0). Packet captures included important details like source and destination IP addresses, ports, protocol types, and packet sizes. Captured data was then filtered using Wireshark's built-in display filters to focus on relevant traffic patterns and reduce noise.

Anomaly detection was implemented by analyzing the captured traffic manually and identifying patterns that deviated from normal behavior. Specific focus areas included detecting abnormal traffic spikes, frequent SYN packets indicating potential port scans, and unusual protocol usage suggesting malicious activities. Analysis techniques included inspecting packet rates, connection attempts, and payload sizes.

Performance optimization during monitoring was achieved by applying selective capture filters to Wireshark to avoid overwhelming the system with unnecessary packet data. Capture sessions were limited to specific traffic types or IP ranges when required, reducing the load on system resources during extended monitoring periods.

Finally, the captured data and detected anomalies were documented for further analysis. Observations from various testing scenarios, such as high-traffic simulations and intentional port scanning attempts, were used to validate the system's effectiveness in monitoring and identifying network anomalies in real time.

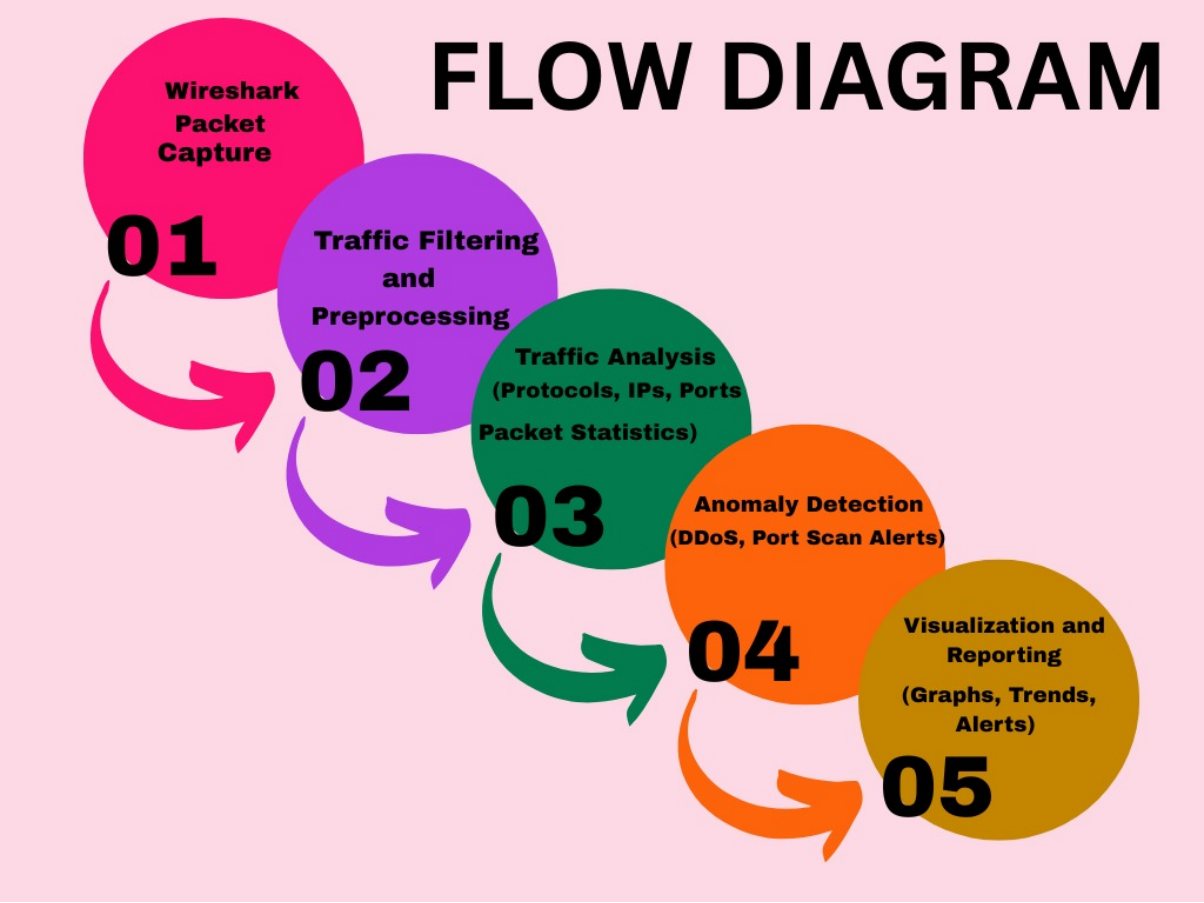
Methodology

Wireshark was used on MacOS to capture live network traffic, including both packet headers and payloads, across various network protocols such as TCP, UDP, ICMP, and others. The raw data collected formed the foundation for detecting anomalies and unusual network behaviors. Wireshark's powerful filtering capabilities allowed the project to isolate specific traffic flows based on parameters like IP addresses, port numbers, or protocol types, ensuring that the analysis focused on relevant and critical data while minimizing noise.

The methodology involved applying a display filter (`display_filter='ip'`) to capture only IP-based packets, thus excluding non-IP traffic such as ARP and Ethernet frames. By focusing specifically on IP traffic, the system streamlined the dataset to include only communications essential to network services like HTTP, TCP, and UDP. This targeted filtering approach made the subsequent traffic analysis more effective and relevant to higher-level protocol behavior.

Captured packets were analyzed to identify key network characteristics, including the types of protocols used, and to track top source and destination IP addresses and ports based on packet count. Suspicious activities such as abnormal traffic volumes, unusually high packet counts, or multiple connections to unique ports were flagged as potential threats. The system provided critical insights into network performance by summarizing statistics like packet count, total bytes transferred, average packets per second, and traffic rate.

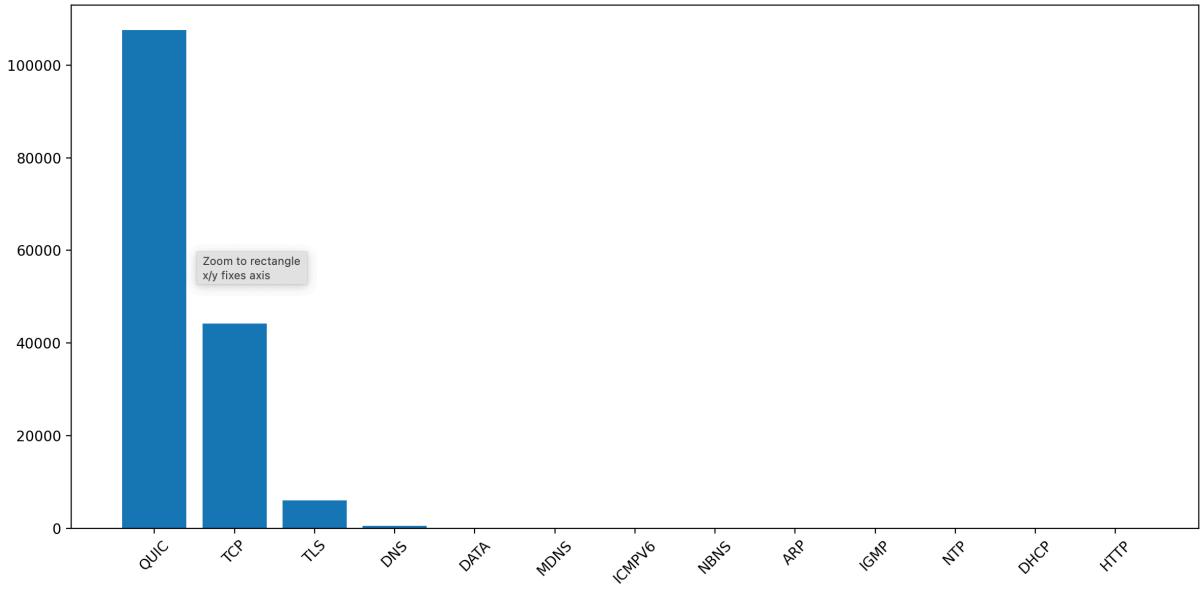
Visualization tools were employed to assist in understanding network behavior over time. Various graphs, including histograms, pie charts, and scatter plots, were generated to represent packet distribution, protocol usage, and traffic fluctuations. These visualizations helped in quickly identifying trends, detecting sudden anomalies, and diagnosing issues such as Distributed Denial-of-Service (DDoS) attacks or port scanning attempts. Overall, the methodology provided an effective framework for anomaly detection and network performance monitoring.



SYSTEM DESIGN

1. **Packet Capture Module**
   The first stage of the system involves setting up Wireshark to capture live network traffic. Wireshark monitors active network interfaces (such as Wi-Fi) and captures packet headers and payloads across different protocols like TCP, UDP, and ICMP. Raw network traffic is collected to serve as the foundation for all further analysis.

-Here the graph shows that collection of packets for each type of protocol.



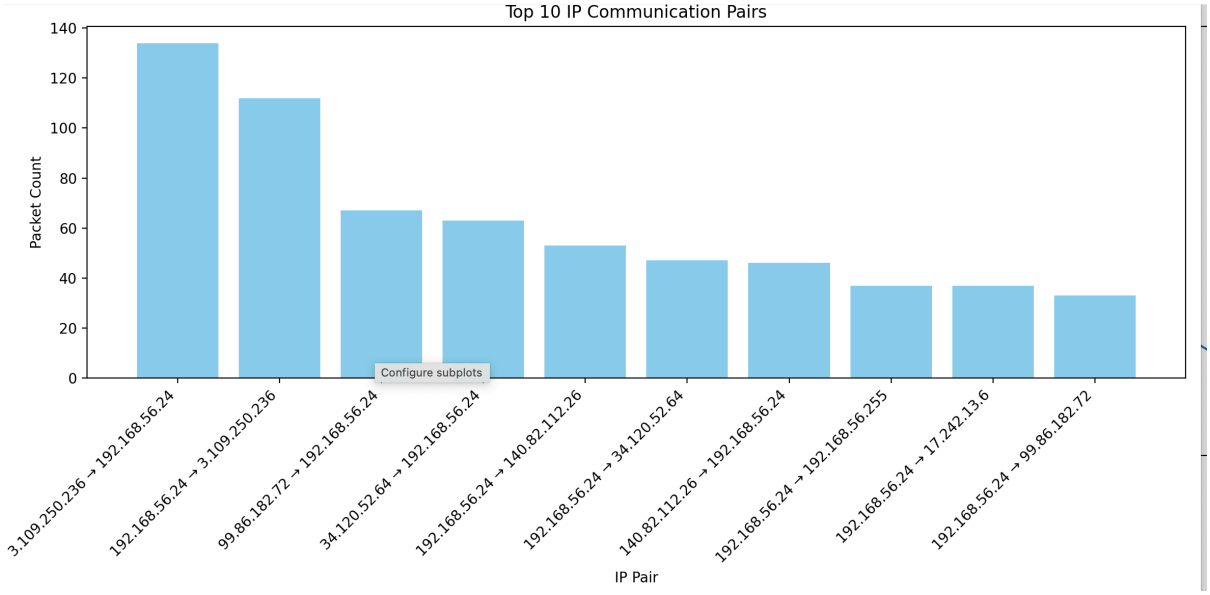
-The Wireshark will do filtertering when the packets are collected .

2. Traffic Filtering and Preprocessing

Captured traffic often contains unnecessary or irrelevant packets (e.g., ARP, Ethernet control frames). To focus the analysis, an IP-based display filter is applied. This ensures that only packets containing IPv4 or IPv6 headers are processed, thus prioritizing the most critical internet communication protocols while reducing noise.

3. Traffic Analysis Module

Filtered packets are analyzed to identify important characteristics such as protocol types, source and destination IP addresses, source and destination ports, and overall packet statistics. Key traffic patterns are extracted, including the identification of top talkers (most active IP addresses) and protocol usage distribution.



-This this the information of captured packets shared by each ip-adresses

and this was collected by using algorithm.

4. Anomaly Detection Engine
Analyzed traffic is scanned for signs of anomalies. Suspicious activities, including sudden spikes in traffic volume, frequent connection attempts on multiple ports (indicating a port scan), or high packet rates from a single IP (suggesting DDoS attacks), are flagged for further inspection. This engine is critical for proactive network security monitoring.

5. Visualization and Reporting Module
To make the network analysis results more understandable, various graphs and visualizations are generated. Histograms, pie charts, and scatter plots are created to depict traffic distribution, protocol usage, and temporal trends. The insights obtained are logged, and suspicious events are highlighted for rapid response and reporting.

IMPLEMENTATION

The network traffic monitoring system was implemented on a MacOS platform using Wireshark for live packet capturing and manual analysis for anomaly detection. Wireshark was installed and configured with the necessary permissions to access the active network interface (such as Wi-Fi, en0). Display filters were applied during the setup phase to ensure that only IP-based traffic was captured, effectively reducing the amount of irrelevant network noise.

Live network traffic was captured over selected durations, recording essential details from each packet, including headers and payloads for protocols like TCP, UDP, and ICMP. The captured data was filtered using Wireshark's display filters to focus specifically on IP traffic, thereby simplifying the dataset and ensuring that the analysis remained centered on the most significant communication activities within the network.

Subsequent to the capture phase, the traffic data was analyzed to track important network attributes such as source and destination IP addresses,

source and destination ports, protocol usage, packet counts, and byte transfers. Key network statistics like total packet count, total bytes transferred, and average packets per second were calculated to establish a baseline understanding of normal network behavior. Analysis also involved identifying top communicating IPs and most frequently used ports, helping in spotting any unusual traffic concentrations.

Anomaly detection was an important aspect of the implementation. Traffic patterns were examined for deviations such as abnormal packet volumes, frequent port accesses from single IP addresses, or sudden traffic spikes that could indicate potential Distributed Denial-of-Service (DDoS) attacks or port scanning attempts. Suspicious IPs were flagged manually based on established thresholds for packet counts and connection diversity.

Visualization techniques were employed to better interpret the network data and anomalies. Using Wireshark's built-in graphing tools and external libraries like Matplotlib, histograms were plotted to represent packet size distributions, pie charts were used to show protocol usage percentages, and scatter plots were created to visualize packet transmission rates over time. These visualizations helped in quickly identifying trends, detecting outliers, and understanding the broader network activity during the monitored periods.

Throughout the implementation, Wireshark served as the primary tool for capturing and dissecting network packets, while statistical summaries and visualizations provided deeper insights into network health and security. The combination of packet capture, selective filtering, statistical analysis, anomaly detection, and data visualization formed a comprehensive approach for effective network traffic monitoring on MacOS.

RESULTS & FUTURE WORK

The implemented network traffic monitoring system successfully captured and analyzed live network traffic on MacOS using Wireshark. Through selective filtering focused on IP-based traffic, the system was able to reduce noise and concentrate on the most significant communication protocols like TCP, UDP, and ICMP. Packet captures included essential attributes such as source and destination IP addresses, ports, protocol types, packet sizes, and timestamps. Statistical summaries were generated, providing insights into total packets captured, total bytes transferred, and average packets per second.

Anomaly detection methods proved effective during testing. The system successfully identified simulated port scanning attempts and traffic spikes, flagging IP addresses with abnormal packet counts and unusual port access patterns. By visualizing the data through histograms, pie charts, and scatter plots, traffic patterns over time were easily observed, making it possible to detect sudden increases or drops in network activity. Overall, the system demonstrated accurate monitoring capabilities with minimal impact on system performance.

While the current system effectively captures and analyzes network traffic, future work can focus on enhancing packet processing efficiency by incorporating scheduling algorithms such as First-Come-First-Serve (FCFS) and Shortest Job First (SJF). Implementing FCFS scheduling would ensure that packets are processed strictly in the order of arrival, maintaining fairness across all captured traffic. On the other hand, adopting SJF scheduling could prioritize packets based on their estimated processing time or size, allowing smaller packets to be processed faster, which would reduce overall system latency during heavy traffic conditions.

Additionally, dynamic and adaptive scheduling strategies could be explored based on real-time traffic behavior to optimize resource utilization and prevent congestion in high-load scenarios. Future extensions could also include automated anomaly detection using machine learning models and the development of a real-time dashboard for centralized monitoring and instant alert generation. These enhancements would make the network traffic monitoring system more scalable, efficient, and suitable for deployment in larger, enterprise-scale networks.

REFERENCES

[1] Wendi Rabiner,Heinzelman, Anantha Chandrakasan, HariBalakrishnan.:Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In:Proceedings ofIEEE 2000.

[2] OssamaYounis, Marwan Krunz,Srinivasan Ramasubramanian, University of Arizona.:Node Clustering in Wireless Sensor Networks: Recent Developments and Deployment Challenges. In: Proceedings of IEEE 2006.

[3] P. Kumarawadu,D. J. Dechene, M. Luccini, A. Sauer.: Algorithms for Node Clustering in Wireless Sensor Networks: A Survey. In: Proceedings of IEEE 2008.

[4] Haowen Chan,Adrian Perrig.:ACE: An Emergent Algorithm for Highly Uniform Cluster Formation.

[5] YaXu, John Heidemann, Deborah Estrin.:Geography informed Energy Conservationfor Ad Hoc Routing.In: Proceedings of IEEE International Conference, 2001.

[6] Benjie Chen, Kyle Jamieson, HariBalakrishnan, Robert Morris.:Span: An EnergyEfficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks.

[7] Abuhelaleh, Mohammed Elleithy, KhaledMismar, Thabet.:Modified LEACH –Energy Efficient Wireless Networks Communication.

[8] V. Loscrì, G. Morabito, S. Marano.:A Two-Levels Hierarchy for Low-Energy Adaptive Clustering Hierarchy (TL-LEACH).In: Proceedings of IEEE 2005.

[9] Mao Ye1, Chengfa Li, Guihai Chen1, Jie Wu.:EECS: An Energy Efficient Clustering Scheme in Wireless SensorNetworks.

[10] OssamaYounis and Sonia Fahmy.:HEED: A Hybrid, Energy-Efficient, DistributedClustering Approach for Ad-hoc Sensor Networks.

[11] Alan D. Amis,Ravi Prakash,Thai H.P.,Vuong Dung,T. Huynh.:Max-Min D-Cluster Formation in Wireless AdHoc Networks.

[12]Maniakchatterjee, Sajal. K.das, DamlaTurgut.:WCA: A Weighted Clustering Algorithm for wireless adhoc networks.

[13] Liyang Yu, Neng Wang, Wei Zhang, Chunlei Zheng.:GROUP: a Grid-clustering Routing Protocol for WirelessSensor Networks.

[14] Siva D. Murugunathan, Daniel C. F. MA, Rolly I. Bhasin, abraham O. Fapojuwo.:A Centralized Energy-Efficient Routing Protocol for Wireless Sensor Networks. In:Proceedings of IEEE 2005.

[15] Guangyan Huang, Xiaowei L,and Jing He.:Dynamic Minimal Spanning Tree Routing Protocol for Large Wireless Sensor Networks.In: Proceedings of IEEE 2006.

[16] Bhaskar P. Deosarkarl, Narendra Singh Yadav,R.P. Yadav.:Cluster head Selection in Clustering Algorithmsfor Wireless Sensor Networks: A Survey.In:Proceedings of 2008 IEEE.

[17] W. Heinzelman, A. Chandrakasan and H. Balakrishnan.:"An Application-Specific Protocol Architecture for Wireless Microsensor Networks".IEEE Trans. Wireless Communications, Vol. 1, No.4,October 2002, pp.660-670

[18] W. Heinzelman.:Application-Specific Protocol Architectures for Wireless Networks.Ph.D Thesis, Massachusetts Institute of Technology, June2000.