



# KLE Technological University

Creating Value,  
Leveraging Knowledge

Dr. M. S. Sheshgiri Campus, Belagavi

A course Project Report on

## Traffic Categorization

Mini Project (22ECSW301)

Submitted by

Team Number: A-13		
Name	Roll no	SRN
Abhishek Angadi	03	02FE22BCS005
Prem V	23	02FE22BCS072
Soukhya Nayak	47	02FE22BCS148
Sharanamma Katti	36	02FE22BCS119

Under the guidance of:

Prof. Anita Kenchannavar

Faculty In-charge:

Dr. Shankar Biradar

Department of Computer Science and Engineering

KLE Technological University's  
Dr. M. S. Sheshgiri College of Engineering and Technology,  
Belagavi – 590 008



Dr.M.S.Sheshgiri College of Engineering &amp; Technology

Department of Computer Science &amp; Engineering

## DECLARATION

We hereby declare that the matter embodied in this report entitled “**Traffic Categorization**” submitted to **KLE Technological University** for the course completion of **Mini Project in the 5th Semester of Computer Science and Engineering** is the result of the work done by us in the Department of Computer Science and Engineering, **KLE Dr. M. S. Sheshgiri College of Engineering, Belagavi**, under the guidance of **Prof. Anita Kenchannavar**, Department of Computer Science and Engineering .We further declare that, to the best of our knowledge and belief, the work reported herein doesn’t form part of any other project on the basis of which a course or award was conferred on an earlier occasion by any other student(s). Also, the results of the work are not submitted for the award of any course, degree, or diploma within this or in any other University or Institute. We hereby also confirm that all of the experimental work in this report has been done by us.

Place: Belagavi

Date: J a n 9 , 2 0 2 4

Abhishek Angadi  
(02FE22BCS005)Prem V  
(02FE22BCS072)Soukhya Nayak  
(02FE22BC148)Sharanamma Katti  
(02FE22BCS119)



Belagavi Campus

Dr.M.S.Sheshgiri College of Engineering &amp; Technology

Department of Computer Science &amp; Engineering

## CERTIFICATE

This is to certify that the project entitled “**Traffic Categorization**”, submitted to **KLE Technological University’s Dr. M. S. Sheshgiri College of Engineering and Technology (Dr. MSSCET), Belagavi**, for the partial fulfilment of the requirements for the course **Mini Project**, is a Bonafide record of the work carried out by **Abhishek A, Prem V, Soukhya N and Sharanamma K**, students in the Department of Computer Science and Engineering, **KLE Technological University’s Dr. MSSCET, Belagavi**, under my supervision. I further certify that the contents of this report, in full or in part, have not been submitted to any other institute or university for the award of any other course or degree.

Place: Belagavi

Date: Jan 9, 2024

Prof. Anita Kenchannavar  
(Project Guide)

Dr. Shankar Biradar  
(Course Coordinator)

Dr. Rajashri Khanai  
(Head of Department)

## ABSTRACT

Efficient network management requires real-time identification and classification of network protocols. Existing tools often focus on general traffic analysis, leaving a gap in protocol-specific insights crucial for monitoring and optimizing performance. A real-time protocol classification system addresses this need by categorizing and analyzing network protocols, enhancing administrators' ability to manage bandwidth, troubleshoot issues, and improve overall network efficiency. The system employs advanced packet inspection techniques and visualization tools to provide actionable insights, streamlining the network management process.

## CONTENTS

LIST OF TABLES	6
LIST OF FIGURES	6
1.INTRODUCTION	8
1.1 BACKGROUND	8
1.2 PROBLEM STATEMEN	8
1.2.1OBJECTIVES	8
2.LITERATURE SURVEY	10
3.DESIGN SPACE	11
3.1 UNDERSTANDING THE PROBLEM	11
3.1.1 PROBLEM SPACE	11
3.1.2 STATE OF ART	11
3.1.3 INTERNAL EXTERNAL FACTORS	12
3.2 PROBLEM FLOW	12
3.2.1 SOLUTION SPACE	13
3.2.2 FEATURES	13
3.2.3 MODULES	14
3.2.4 RELATIONSHIP BETWEEN MODULES	16
3.3 INTERFACE DESIGNING	17
3.4 INCLUSIVE DESIGNING	
3.4.1 SOCIETY	17
3.4.2 END USERS	17
3.4.3 PRIVACY AND SECURITY	17
4.REQUIREMENTS ENGINEERING	19
4.1ACTORS	19
4.2GENERAL REQUIREMTS	20
4.2.1INFRASTRUCTURE	21
4.2.2COMPETITOR	22
4.3FUNCTIONAL AND NON- FUNCTIONAL REQUIREMENTS	23
4.3.1FUNCTIONAL REQUIREMENTS	23
4.3.2NON-FUNCTIONAL REQUIREMENTS	24
4.4USE CASE DIAGRAM	

5.SYSTEM MODELING	
5.1 UML DIAGRAMS	26
5.1.1 CONTEXT DIAGRAMS	27
5.1.2 ACTIVITY DIAGRAMS	28
5.2 MODEL ARCHITECTURE	28
6.IMPLEMENTATION	31
6.1 ALGORITHM DEVELOPMENT	32
6.2 USER INTERFACE DESIGN	32
7.TESTING	35
7.1 RESULTS	35
7.2 TESTING REPORT	36
7.3 TESTING TOOL	37
RESULTS AND OUTCOMES	39
CONCLUSION	42
REFERENCES	42

### LIST OF TABLE

VII.	Table 7.1 Test Cases	36
------	----------------------	----

### LIST OF FIGURES

IV.	Fig. 4.1 Use case Diagram	24
V.	Fig. 5.1 UML Diagram	26
V.	Fig. 5.2 Context Diagram	27
V.	Fig. 5.4 Activity Diagram	28
V.	Fig. 5.5 Sequence Diagram	29
VI.	Fig. 6.1 User Interface	34
VII.	Fig. 7.1 Wireshark Testing Interface	35

---

VII.	Fig. 7.2	Running Python Script	37
VII.	Fig. 7.3	Checking the Iptables Rules	37
VII.	Fig. 7.4	Adding the Iptables rules	38
VII.	Fig. 7.5	Pie chart	38

# CHAPTER 1

## INTRODUCTION

### 1.1 BACKGROUND

Network traffic refers to the flow of data, in the form of packets, being transmitted across a network. It encompasses all the communications that occur between devices (such as computers, servers, smartphones, etc.) within a network, including local area networks (LANs), wide area networks (WANs), and the internet. Traffic can be categorized based on protocols (HTTP, FTP, etc.) or based on the actual content carried in the packets (e.g., video, audio, image). Our project focuses on this protocol-based categorization, where the goal is to classify traffic by analyzing the protocol type inside the packets.

### 1.2 PROBLEM STATEMENT

Understanding and classifying network protocols in real-time is important for efficient network management. Existing tools often provide general traffic analysis but lack detailed insights into specific protocols. This makes it difficult for administrators to monitor protocol usage and optimize network performance. A real-time protocol classification system solves this by identifying and categorizing protocols, making network management more effective and streamlined.

#### 1.2.1 OBJECTIVES

- **Packet Analysis:**
  - To analyze and monitor network traffic at the packet level.
- **Protocol Identification:**
  - To categorize network packets based on their protocol types (e.g., TCP, UDP, HTTP, HTTPS, DNS, etc.).
- **Real-Time Classification:**



- To classify packets in real-time to provide immediate insights into network activity.
- **Visualization:**
  - To represent classified packets visually using tools such as pie charts or graphs for better understanding and interpretation.
- **Monitor Traffic:**
  - Provide a basic overview of network traffic distribution.
- **Simplify Network Understanding:**
  - Help users understand what types of data are flowing through the network.
- **Automate Classification:**
  - Automatically identify and group packets by protocol

## CHAPTER 2

### LITERATURE SURVEY

Velan et al. (2015) reviewed and categorized methods for analyzing encrypted traffic using feature-based and behavior-based classification methods [1]. Their study highlighted challenges in encrypted traffic classification, which are crucial for network security and monitoring. However, the results may not generalize due to differing datasets and challenges associated with encrypted data.

Khalife et al. (2014) developed a systematic taxonomy for traffic classification, considering encryption and obfuscation challenges [2]. They used a three-level hierarchical taxonomy based on input, technique, and output levels. This approach provided a consistent framework for evaluating traffic classification methods and guiding future research. Nevertheless, limited benchmarks and difficulties with encrypted and obfuscated traffic posed challenges.

Piskač (2012) focused on improving protocol detection in high-speed networks by analyzing the time characteristics of network flows, specifically inter-arrival times [3]. By leveraging flow behavior and machine learning, the study aimed to enhance high-speed network security by improving traffic classification accuracy. However, inaccuracies in behavioral analysis due to changing network conditions and limited hardware capabilities for precise measurements were notable drawbacks.

Silvio Valenti et al. (2013) conducted a comparative analysis of traffic classification approaches using supervised classifiers [4]. Behavioral classifiers employing supervised machine learning algorithms were essential for improving traffic management and security operations for ISPs and network administrators. However, computational challenges with large datasets and limited accuracy for encrypted traffic were significant disadvantages.

## CHAPTER 3

### DESIGN SPACE

#### 3.1 UNDESTANDING THE PROBLEM

Software-Defined Networking (SDN) makes managing networks easier with its centralized control and flexibility, but this also makes it a target for Distributed Denial of Service (DDoS) attacks. These attacks flood the network with fake traffic, causing it to slow down or stop working entirely. Traditional methods struggle to detect these attacks quickly because the patterns of attacks keep changing. This creates a need for smart systems that can analyze network traffic in real time, spot unusual activity, and respond effectively to protect the network. Solving this problem is essential to keeping SDN networks secure and reliable.

##### 3.1.1 PROBLEM SPACE

The problem space of traffic categorization focuses on identifying and managing network packets based on their protocols (e.g., HTTP, DNS, TCP) rather than their data types. Modern networks, driven by applications like IoT, streaming services, and cloud computing, generate large volumes of diverse traffic, making efficient categorization crucial. This process enables enhanced security, performance optimization, and effective resource allocation. Challenges arise due to the increasing use of encrypted protocols like HTTPS, which limit visibility, the evolving nature of protocol standards, and the real-time constraints in high-speed networks. Addressing these challenges requires balancing accuracy, computational efficiency, and scalability to ensure robust categorization solutions.

##### 3.1.2 STATE OF ART

Current methods for traffic categorization include traditional rule-based approaches and advanced AI-driven techniques. Rule-based systems like Wireshark use static port and protocol identification, which are simple and fast but struggle with encrypted or non-standard traffic. Machine learning (ML) methods, such as Random Forest and Support Vector Machines (SVM), analyze packet patterns and metadata to classify protocols dynamically. Deep learning approaches, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), excel at handling complex traffic but require significant computational power. Emerging trends focus on hybrid systems that combine static and dynamic methods, lightweight algorithms

for resource-constrained environments, and AI-driven models that adapt to evolving protocols, ensuring real-time efficiency and accuracy

### 3.1.3 INTERNAL EXTERNAL FACTORS

#### **Internal Factors:**

Internal factors are those directly controlled by the development team and significantly influence the design and functionality of the traffic categorization system. These include the availability and quality of labeled datasets, which are essential for training machine learning models. Computational resources, such as GPUs and memory, determine the system's ability to process packets in real time and handle large traffic volumes. The scalability of the system is critical for adapting to high-throughput environments, ensuring smooth operation as traffic grows. Additionally, the system's adaptability to new protocols and traffic patterns through model retraining and updates is vital for maintaining accuracy over time. Lastly, simplicity and transparency in model design help ensure that the system is maintainable and interpretable.

#### **External Factors:**

External factors are those beyond the direct control of the developers but heavily influence the system's deployment and effectiveness. The network environment, including topology, traffic patterns, and volume, dictates the diversity of protocols encountered. Encrypted protocols, such as HTTPS and TLS, limit visibility into packet payloads, making protocol identification more challenging. Regulatory compliance requirements, such as GDPR and CCPA, mandate strict adherence to data privacy and security laws, affecting how data is processed and stored. Cybersecurity threats like DDoS attacks and intrusion attempts necessitate robust mechanisms for real-time anomaly detection. Advances in networking technologies, such as 5G and edge computing, further require the system to be adaptable to emerging trends and scalable to meet increasing demands.

## 3.2 PROBLEM FLOW

The problem flow in traffic categorization involves several sequential steps: capturing network packets, extracting relevant features, classifying the packets, and deciding actions based on their classification. Network traffic is captured using tools like iptables and nfqueue, redirecting packets to user-space applications for analysis. Features like protocol type, packet size, and source/destination addresses are extracted. These features

are then used in classification systems, which can be rule-based or employ machine learning and deep learning models. After classification, actions such as allowing, dropping, or modifying packets are performed to ensure network security and performance.

### 3.2.1 SOLUTION SPACE

- I. Static Rule-Based Methods:** Static rule-based methods rely on predefined rules, such as port numbers or protocol identifiers, to classify traffic. For instance, port 80 indicates HTTP traffic, while port 443 identifies HTTPS traffic. Tools like Wireshark and Snort are widely used in this approach. These methods are straightforward, fast, and easy to interpret, making them ideal for simple traffic scenarios. However, they struggle with encrypted protocols, port obfuscation, and non-standard traffic, limiting their effectiveness in modern networks.
- II. Machine Learning-Based Methods :** Machine learning (ML) approaches, such as Random Forest, Support Vector Machines (SVM), and k-Nearest Neighbors (k-NN), use statistical models to identify patterns in network traffic data. By analyzing features like packet size, inter-arrival times, and flow characteristics, ML models can classify packets dynamically without relying on static rules. These methods are adaptive and can handle complex traffic patterns. However, they require extensive labeled datasets for training and significant computational resources, which may limit their applicability in resource-constrained environments.
- III. Deep Learning-Based Methods :** Deep learning models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) excel at identifying complex traffic patterns and handling large-scale data. CNNs analyze packet data as structured sequences, while RNNs leverage temporal dependencies for flow-based classification. These models are particularly effective in handling encrypted traffic and evolving protocols, offering high accuracy. However, they are computationally intensive, requiring powerful hardware such as GPUs, and often lack interpretability, which may be a drawback in environments demanding transparency.
- IV. Hybrid Approaches:** Hybrid methods combine rule-based techniques with machine learning or deep learning models. For example, rule-based systems can perform initial filtering, while ML models classify more ambiguous traffic. This approach balances speed and adaptability, making it suitable for diverse traffic environments. Hybrid systems can efficiently handle known protocols while leveraging AI for dynamic and unknown traffic patterns, providing a robust and flexible solution.

**V. Flow-Based Methods:** Flow-based approaches analyze aggregated packet flows rather than individual packets, focusing on flow metadata such as source/destination IPs, flow duration, and packet counts. These methods are lightweight and efficient, making them suitable for encrypted traffic. However, they lose granularity and may struggle with irregular or low-volume traffic patterns. Flow-based methods are ideal for large-scale networks like ISPs but may miss finer details critical for detecting specific anomalies.

### 3.2.2 FEATURES

#### 1. Real-Time Processing

The system processes network packets as they are captured, ensuring immediate classification and action. This minimizes delays, making it suitable for real-time applications like traffic monitoring and intrusion detection, where quick decisions are critical.

#### 2. Protocol-Based Categorization

Packets are categorized based on standard protocols such as HTTP, DNS, TCP, and UDP by analyzing their headers and metadata. This allows administrators to monitor traffic types and prioritize latency-sensitive protocols while ensuring accurate network management.

#### 3. Scalability

The system is designed to manage traffic loads effectively, whether in small-scale setups like home networks or large-scale enterprise environments. It ensures seamless performance even as the volume of network traffic increases.

#### 4. Adaptability

The system can evolve with network trends by learning new protocols or patterns through model updates and retraining. This adaptability ensures it remains effective even as technologies and protocols change over time.

#### 5. Visualization

Categorized traffic is represented through intuitive visual tools like pie charts and graphs. These real-time visualizations provide users with a clear understanding of traffic distribution and allow for easy monitoring of network activity.

### 3.2.3 MODULES

#### 1. Packet Capture Module

- **Purpose:** This module captures network traffic in real time for further processing.
- **How It Works:** Tools like iptables and nfqueue are used to redirect packets from the network stack to a user-space application.
- **Significance:** It ensures all incoming and outgoing packets are captured for analysis without missing any data.

#### 2. Feature Extraction Module

- **Purpose:** This module extracts essential features from each packet for classification.
- **Key Features:** Protocol type, packet size, source and destination IPs, and flow characteristics like inter-arrival times.
- **How It Works:** The module parses packet headers to collect relevant metadata while avoiding sensitive payload inspection.
- **Significance:** Provides the data necessary for the classification process while maintaining efficiency and privacy.

#### 3. Classification Module

- **Purpose:** This module classifies packets into predefined categories based on extracted features.
- **Techniques:**
  - **Rule-Based:** Uses predefined rules like port numbers for standard traffic types.
  - **Machine Learning Models:** Applies algorithms like Random Forest or SVM for dynamic classification of complex traffic patterns.
- **Significance:** Differentiates between legitimate, suspicious, or unknown traffic efficiently.

#### 4. Action Module

- **Purpose:** Decides the appropriate action for each packet based on its classification.
- **Actions Taken:**
  - **Allow:** Legitimate packets are forwarded to their destination.

- **Drop:** Malicious or suspicious packets are blocked.
- **Modify:** Packets may be altered, such as anonymizing data before forwarding.
- **Significance:** Ensures network security and traffic management by handling packets appropriately.

## 5. Feedback Module

- **Purpose:** Continuously improves the system by adapting to new traffic patterns.
- **How It Works:**
  - Updates classification rules or retrains machine learning models based on new data.
  - Implements feedback loops to refine accuracy over time.
- **Significance:** Keeps the system robust and relevant by addressing evolving network trends and newly introduced protocols.

## 3.2.4 RELATIONSHIP BETWEENMODULES

### 1. Packet Capture and Feature Extraction

The Packet Capture Module is the entry point of the system, where all incoming and outgoing packets are captured using tools like iptables and nfqueue. The captured packets are forwarded to the Feature Extraction Module, which parses packet headers to collect critical metadata such as protocol type, source/destination IPs, ports, and packet size. This dependency ensures that raw packet data is transformed into meaningful features required for analysis. Without accurate and complete data capture, feature extraction would fail to provide reliable inputs for classification.

### 2. Feature Extraction and Classification

The features extracted from the packets are fed into the Classification Module, which uses predefined rules or machine learning models to categorize the packets. The accuracy of the classification heavily depends on the quality and relevance of the features extracted. For instance, identifying specific protocols like HTTP or DNS requires analyzing both packet metadata and patterns within the flow. This tight coupling ensures that the classification process is efficient and robust, addressing both standard and dynamic traffic patterns.

### 3. Classification and Action Execution

The Action Module operates based on the decisions made by the Classification Module. Once a packet is categorized, the Action Module determines whether to allow, drop, or modify it. For example, legitimate traffic is allowed, while suspicious packets are blocked to maintain security. Modified packets, such as anonymized



data, ensure compliance with privacy policies. The seamless handover of classified data ensures that appropriate actions are taken promptly, minimizing network delays and improving security.

#### **4. Feedback and Continuous Adaptation**

The Feedback Module plays a critical role in refining the system over time. It analyzes the outcomes of the classification and actions taken to identify inaccuracies or evolving patterns in network traffic. Insights from this module are looped back into the Feature Extraction and Classification Modules, enabling updates to rules or retraining of machine learning models. This relationship ensures that the system adapts to new protocols, changing traffic behaviors, and emerging threats.

#### **5. Overall Interconnection and Synergy**

The modules form an interdependent ecosystem where each component relies on the others for optimal performance. The Packet Capture Module feeds raw data into the pipeline, Feature Extraction provides meaningful inputs for classification, the Classification Module categorizes traffic, and the Action Module enforces decisions. The Feedback Module enhances adaptability, creating a dynamic and self-improving system. This interconnected design ensures the system is efficient, accurate, and responsive to both current and future network challenges.

### **3.4 INCLUSIVE DESIGNING**

#### **3.4.1 SOCIETY**

The system contributes to a safer and more inclusive digital society by addressing critical network security and performance issues. It enhances network security by identifying and mitigating cyber threats such as malicious traffic and unauthorized access. This improves trust in digital systems, fostering safe online interactions for individuals and organizations alike. Accessibility is a key focus, with multi-language support enabling diverse users to benefit from the system irrespective of their linguistic backgrounds. Intuitive visual tools, such as pie charts and real-time dashboards, simplify complex traffic data, making it comprehensible to both technical and non-technical users. By promoting secure and efficient network operations, the system supports the global digital infrastructure, creating an environment that is more inclusive, accessible, and secure for everyone.

#### **3.4.2 END USERS**

The system caters to a wide range of end users, including network administrators, security analysts, and casual users with minimal technical knowledge. For technical users, such as administrators and analysts, it offers advanced insights into network traffic, customizable classification rules, and real-time alerts for quick decision-making. These features empower them to monitor and manage networks effectively. Non-technical users

benefit from user-friendly interfaces, simplified dashboards, and clear visualizations, allowing them to understand network status without needing technical expertise. Accessibility features, such as high-contrast modes for visually impaired users, screen reader compatibility, and keyboard navigation, ensure that people with disabilities can also interact with the system seamlessly. This inclusive approach makes the system practical and accessible for a diverse user base.

### **3.4.3 PRIVACY AND SECURITY**

The system is designed with privacy and security as core principles. It focuses on analyzing protocol headers rather than packet payloads, ensuring that sensitive information like message content and personal data remains unexposed. Data is encrypted during storage and transmission to prevent unauthorized access or tampering. Role-based access control ensures that only authorized users can interact with sensitive system functions, while detailed logging and regular security audits maintain transparency and accountability. Additionally, compliance with data protection regulations such as GDPR and CCPA guarantees that the system respects user privacy and upholds legal standards. This robust approach to privacy and security builds user trust while protecting networks from potential breaches and ensuring that operations remain transparent and ethical.

## CHAPTER 4

# REQUIREMENTS ENGINEERING

### 4.1 ACTORS

The actors in the system represent the primary users and components interacting with the solution. Their roles and responsibilities ensure the seamless operation of the system. These include:

#### 1. Network Administrators:

- **Role:** These are the primary users responsible for managing the system and analyzing network traffic.
- **Responsibilities:**
  - Monitoring network traffic in real-time to identify potential anomalies or bottlenecks.
  - Reviewing classified traffic data to make informed decisions on network optimization or threat mitigation.
  - Configuring and maintaining the system components, such as updating classification rules or retraining models as needed.

#### 2. End Users:

- **Role:** Individuals or organizations that indirectly benefit from the system's outcomes, such as enhanced security and network reliability.
- **Responsibilities:**
  - Leveraging secure and stable internet connections for day-to-day operations.
  - Trusting the system to protect sensitive data and ensure network performance.

### 3. System Components:

- **Role:** These include the hardware and software elements that collaborate to perform packet capturing, processing, and analysis.
- **Responsibilities:**
  - Ensuring reliable and accurate data capture through tools like iptables and nfqueue.
  - Performing classification tasks using the implemented algorithms.
  - Providing processed data to the visualization and feedback modules for actionable insights.

## 4.2 GENERAL REQUIREMENTS

### 4.2.1 INFRASTRUCTURE

The infrastructure requirements are critical for the effective deployment and operation of the system. These include:

#### 1. Hardware:

- **Components:**
  - High-performance servers capable of running the classification algorithms and handling data storage.
  - Routers and networking devices for seamless traffic monitoring and interception.
- **Specifications:**
  - Multi-core processors and high memory capacity to process traffic in real-time.
  - High-speed network interfaces to capture data without loss.

#### 2. Software:

- Tools and frameworks are essential for enabling system operations:
  - **iptables:** To redirect packets to the nfqueue for processing.

- **nfqueue:** For temporarily holding and analyzing captured packets.
- **Python Libraries:** Including Scapy for packet manipulation, Matplotlib/Plotly for visualization, and machine learning frameworks like TensorFlow or Scikit-learn for classification tasks.

### 3. Network Bandwidth:

- The system requires adequate bandwidth to process high volumes of traffic in real-time without affecting network performance.
- **Requirement:** At least 1 Gbps connectivity for large-scale networks, ensuring smooth packet interception and analysis.

### 4. Storage:

- Storage must support both short-term and long-term data handling:
  - Real-time storage for immediate packet processing and visualization.
  - Archival storage for logs, historical traffic data, and audit trails.
- **Specifications:**
  - SSDs for high-speed real-time operations.
  - Large-capacity drives or cloud storage for data retention.

## 4.2.2 COMPETITOR

Competitor analysis highlights the gaps in existing systems and positions the proposed solution effectively.

### 1. Existing Systems:

- Tools like **Wireshark** and commercial platforms offer packet analysis capabilities but lack real-time classification and user-friendly interfaces.

### 2. Gaps in Existing Systems:

- **Real-Time Classification:** Most competitors focus on capturing and analyzing packets but don't classify them in real time.
- **User-Friendly Dashboards:** Limited graphical tools for end-users, especially those unfamiliar with technical aspects.

- **Adaptability:** Competitor systems are often rigid, making them less effective in dynamic network environments where traffic patterns frequently change.

### 3. Advantages of the Proposed System:

- **Visualization Module:** Provides an intuitive and real-time graphical representation of traffic, enabling easy understanding for technical and non-technical users.
- **Feedback Loop:** Ensures adaptability to new protocols or changing traffic patterns.
- **Accessibility:** Simplified interfaces make the system usable for a diverse audience, including non-expert users.

## 4.3 FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

### 4.3.1 FUNCTIONAL REQUIREMENTS

These requirements outline the essential features and functionalities the system must provide:

#### 1. Packet Capture:

- **Requirement:** The system must intercept packets traveling across the network.
- **Implementation:**
  - Utilize iptables to redirect packets to nfqueue for capturing.
  - Support for various protocols, including TCP, UDP, and ICMP.

#### 2. Classification:

- **Requirement:** Identify the protocol types of captured packets.
- **Implementation:**
  - Use machine learning models or rule-based approaches to classify traffic as HTTP, FTP, DNS, or other protocols.
  - Provide options for administrators to refine classification rules.

#### 3. Visualization:

- **Requirement:** Display real-time traffic distribution through graphical tools.

- **Implementation:**

- Use pie charts, bar graphs, and line graphs to represent traffic proportions, packet counts, or protocol types.
- Provide a dashboard accessible via a web interface for administrators and end-users.

#### 4. Feedback Loop:

- **Requirement:** The system must adapt classification rules or models based on changing traffic patterns.
- **Implementation:**
  - Enable administrators to upload new training data for machine learning models.
  - Automatically identify anomalies and suggest updates to classification rules.

### 4.3.2 NON-FUNCTIONAL REQUIREMENTS

These requirements address the performance, security, and usability aspects of the system:

#### 1. Scalability:

- **Requirement:** Handle increasing traffic volumes without performance degradation.
- **Implementation:**
  - Design the system to operate efficiently on both small-scale (local networks) and large-scale (enterprise-level) setups.
  - Support distributed processing for high-traffic environments.

#### 2. Security:

- **Requirement:** Protect data integrity and confidentiality.
- **Implementation:**
  - Encrypt data during storage and transmission to prevent unauthorized access.
  - Focus on analyzing packet headers instead of payloads to avoid handling sensitive data directly.

### 3. Reliability:

- **Requirement:** Ensure consistent uptime and accurate results.
- **Implementation:**
  - Implement robust error-handling mechanisms to manage unexpected scenarios.
  - Use redundant storage and failover systems to maintain high availability.

### 4. Usability:

- **Requirement:** Design the system to be intuitive and accessible for diverse users.
- **Implementation:**
  - Provide a clean and simple user interface with easy navigation.
  - Include accessibility features like high contrast, screen reader compatibility, and multi-language support.

## 4.4 USE CASE DIAGRAM

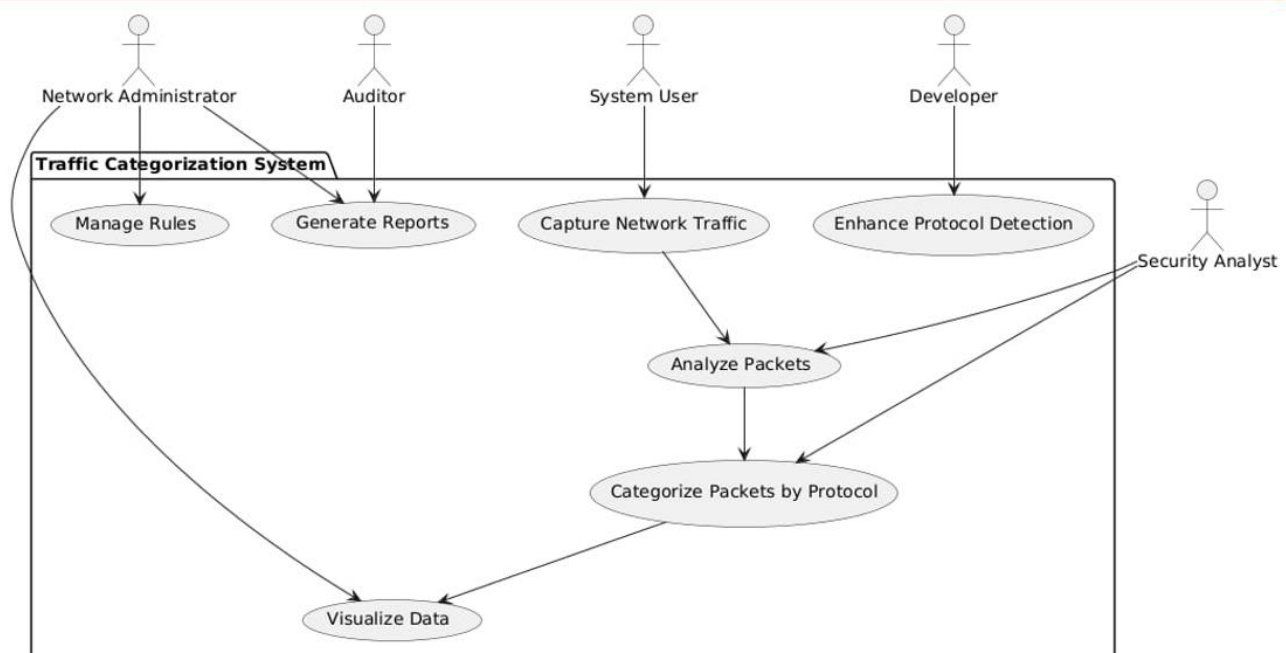


Fig 4.1 use case diagram



In Fig. 4.1 The Use Case Diagram depicts the Traffic Categorization System and its interactions:

Actors:

- Network Administrator: Manages traffic rules.
- Auditor: Generates reports.
- System User: Captures network traffic.
- Developer: Improves protocol detection.
- Security Analyst: Analyzes traffic for security.

Use Cases:

- Manage Rules: Set or modify rules.
- Generate Reports: Create traffic summaries.
- Capture Network Traffic: Gather data for analysis.
- Analyze Packets: Inspect packets for details.
- Categorize Packets by Protocol: Group by protocol type.
- Enhance Protocol Detection: Refine classification algorithms.
- Visualize Data: Present insights visually.

## CHAPTER 5

### SYSTEM MODELLING

System modeling refers to the process of creating abstract representations of a system to understand, analyze, and improve its performance or functionality. These models help visualize the system's structure, behavior, and interactions to solve problems or make decisions.

#### 5.1 UML DIAGRAMS

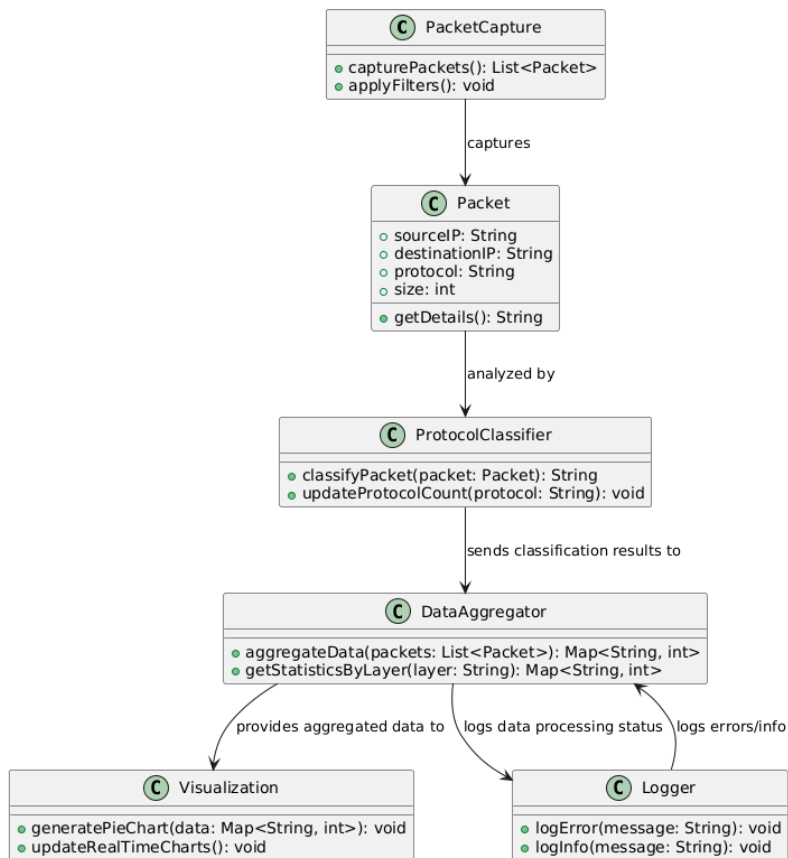


Fig 5.1 UML diagram

In Fig. 5.1 The diagram represents a class diagram for a Traffic Classification System with the following components:

Packet Capture: Captures network packets and applies filters.

Packet: Represents a network packet with attributes like source IP, destination IP, protocol, and size.

Protocol Classifier: Classifies packets and updates protocol counts.

DataAggregator: Aggregates packet data and provides statistics, interacting with the visualization and logging components.

Visualization: Displays traffic data through pie charts and real-time charts.

Logger: Logs errors and information about data processing.

Relationships:

Packet Capture captures Packet.

Packet is analyzed by the protocol classifier.

protocol Classifier sends results to data aggregator.

DataAggregator provides data to Visualization and logs processing status via Logger.

### 5.1.1 CONTEXT DIAGRAMS

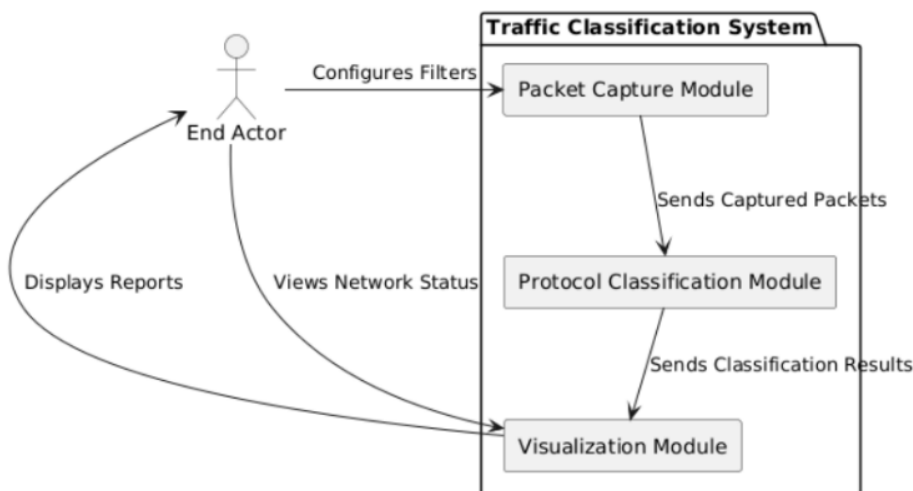


Fig 5.2 Context Diagram

In Fig. 5.2 The context diagram shows how the Traffic Classification System interacts with two main actors:

**Network Administrator:** Configures filters for the Packet Capture Module and views reports from the Visualization Module.

**End User:** Views network status via the Visualization Module.

The system consists of:

**Packet Capture Module:** Captures network packets.

**Protocol Classification Module:** Classifies the packets.

**Visualization Module:** Displays reports and real-time traffic data.

The flow connects modules for packet capture, classification, and visualization while facilitating interactions with the administrator and end users.

## 5.1.2 ACTIVITY DIAGRAM

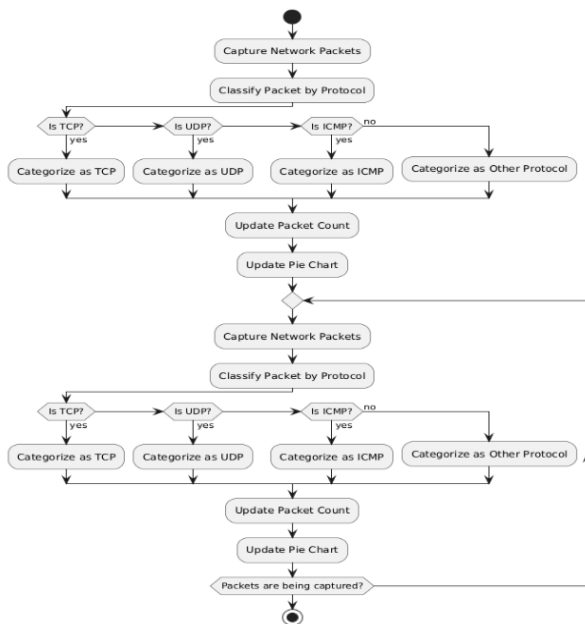


Fig 5.3 Activity Diagram

In Fig 5.3 This flowchart represents the process of capturing and classifying network packets in real time:

**Start:** The system begins capturing network packets.

**Classification:** Each packet is checked to determine its protocol (TCP, UDP, ICMP, or other).

Categorization: The packet is categorized based on its protocol type.

Data Update: Counts are updated, and the pie chart reflects the changes dynamically.

Loop: The process repeats continuously while packets are being captured.

Stop: The system stops when no more packets are detected.

It visualizes a real-time packet analysis and visualization workflow.

### 5.1.3 SEQUENCE DIAGRAM

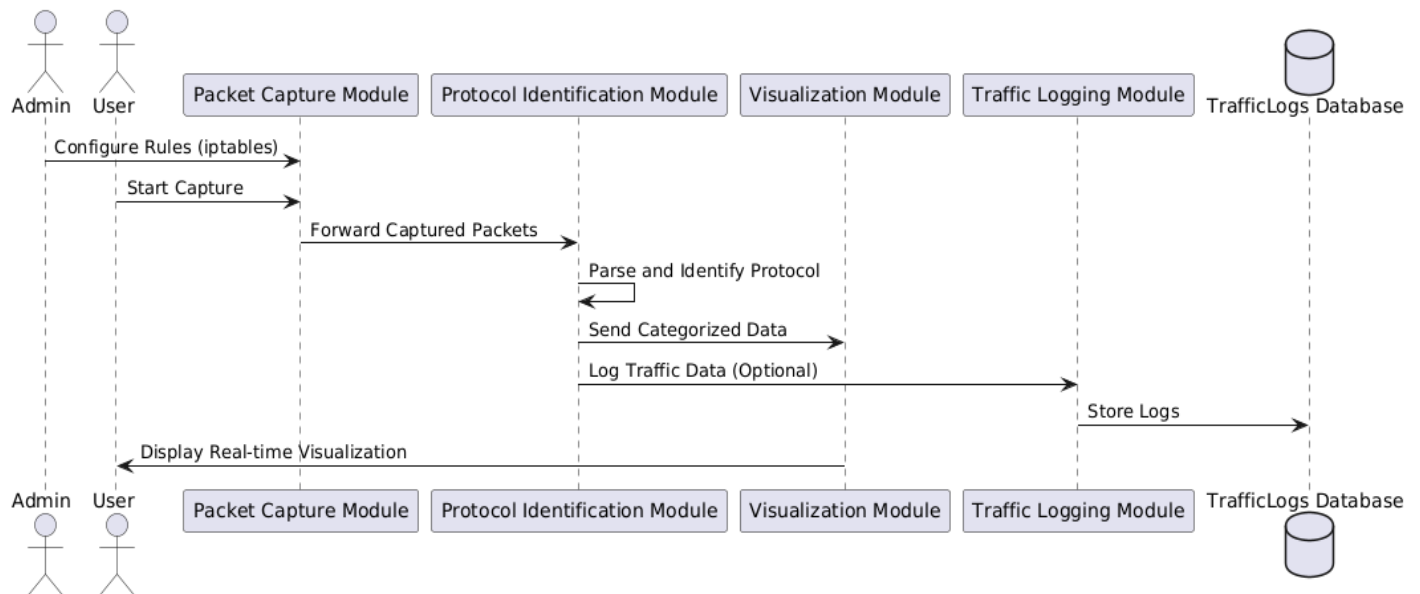


Fig 5.4 Sequence Diagram

In Fig 5.4 The Sequence Diagram outlines the workflow of a Traffic Categorization System:

Admin/User Interaction:

Admin configures packet capture rules using iptables.

User starts the packet capture process.

Packet Capture Module: Captures network packets and forwards them to the Protocol Identification Module.

Protocol Identification Module: Parses the packets to identify their protocols.

Sends the categorized data to the Visualization Module.

Visualization Module: Displays real-time traffic data visualization to the Admin and User.

Traffic Logging Module (Optional): Logs the categorized traffic data.

Stores the logs in the Traffic Logs Database for future reference.

This system efficiently captures, categorizes, visualizes, and optionally logs network traffic data

## CHAPTER 6

### IMPLEMENTATION

#### 6.1 ALGORITHM DEVELOPMENT

The algorithm for the Traffic Categorization Project is designed to classify and visualize network traffic in real-time. It leverages Python scripts, iptables, and nfqueue for packet redirection, analysis, and categorization. The steps are as follows:

##### Step 1: Packet Capture and Redirection

1. Use iptables rules to redirect network packets to the nfqueue.
  - Example: Sudo iptables -I INPUT -j NFQUEUE --queue-num 1
  - Sudo iptables -I OUTPUT -j NFQUEUE --queue-num 1

2. Packets reaching the queue are passed to the Python script for processing.

##### Step 2: Packet Parsing and Feature Extraction

1. In the Python script, packets from nfqueue are parsed using the scapy library.
2. Key features are extracted:
  - Network Layer: Identify IPv4 or IPv6.
  - Transport Layer: Classify protocols such as TCP, UDP, or ICMP.
  - Application Layer: Determine the protocol type based on port numbers (e.g., HTTP, DNS, FTP).

##### Step 3: Packet Classification

1. Use the extracted features to increment counters for specific protocols at each layer.
2. Maintain relational statistics combining layers, such as HTTP-TCP-IPv4.

#### Step 4: Real-Time Visualization

1. Update pie charts dynamically using matplotlib to reflect the following:
  - Application Layer Protocols
  - Transport Layer Protocols
  - Network Layer Protocols
  - Relationships between these layers.
2. Ensure smooth rendering with the FuncAnimation module.

#### Step 5: Error Handling and Logging

1. Ensure the script gracefully handles malformed packets.
2. Log errors for debugging without interrupting real-time processing.

## 6.2 USER INTERFACE DESIGN

The user interface (UI) design focuses on providing real-time insights into network traffic through intuitive visualizations. Below are the components of the UI:

### Layout

1. Dashboard View:
  - Four dynamic pie charts displayed in a grid layout:
    - Application Layer
    - Transport Layer
    - Network Layer
    - Layer Relationships



- A status bar at the bottom for:
  - Total packets processed.
  - Current processing speed.

## Key Features

1. Real-Time Updates:
  - Use matplotlib to render pie charts updated every second with new statistics.
  - Ensure charts are visually engaging with appropriate labels, legends, and color coding.
2. User Feedback:
  - Provide a notification or log area to display errors, packet counts, or alerts.
3. Responsive Design:
  - The UI adjusts automatically for different screen resolutions.

## Technology Stack

- Python Libraries:
  - matplotlib for visualizations.
  - scapy for packet parsing.
  - netfilterqueue for interfacing with nfqueue

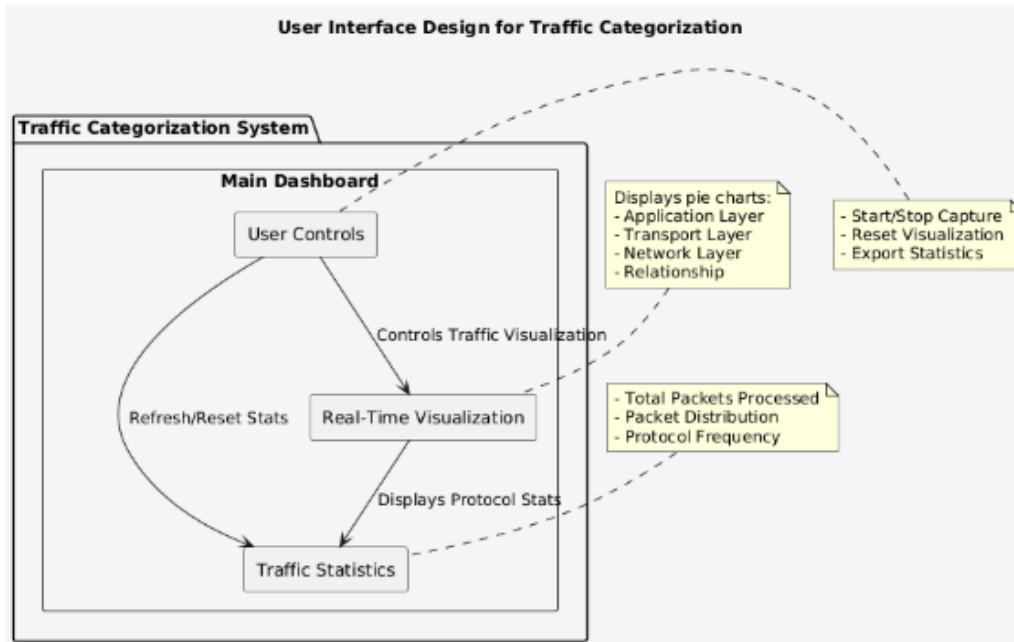


Fig 6.1 User interface design

In Fig. 6.1 The diagram illustrates a UI design for traffic categorization:

Components:

- Packet Capture: Buttons to start and stop packet capture.
- Protocol Display: Shows counts for TCP, UDP, ICMP, and other protocols.
- Real-time Pie Chart: Visualizes protocol distribution dynamically.
- Flow: Capturing packets updates the Protocol Display and Pie Chart in real-time

# CHAPTER 7

## TESTING

### 7.1 RESULT

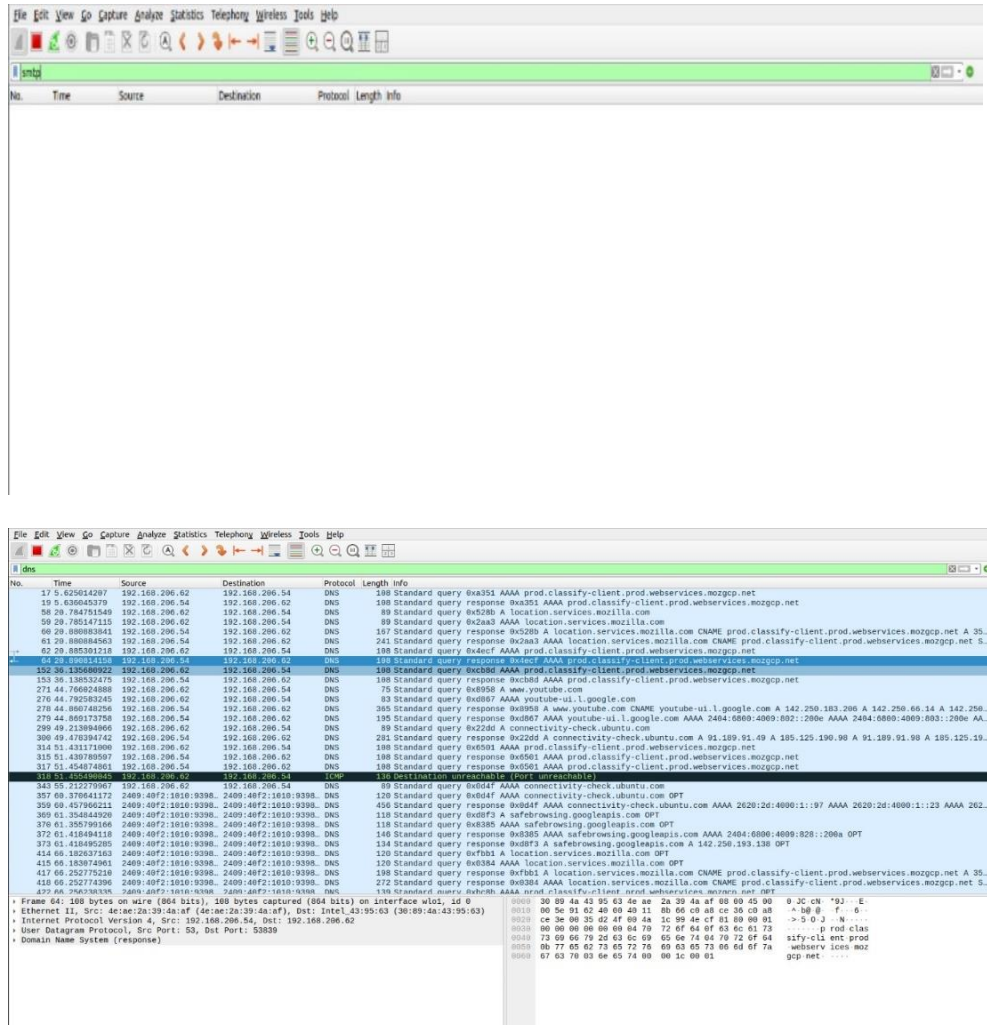


Fig 7.1 Wireshark Testing Interface

In Fig. 7.1 Purpose: To validate DNS and SMTP traffic using Wireshark and compare with the pie chart outputs.

DNS Traffic: In Wireshark: DNS packets are present, using UDP (port 53) for domain name resolution.

In Pie Charts:

- Application Layer: DNS is 9.3%.

- Transport Layer: Part of the 15.6% UDP traffic.
- Network Layer: Uses both IPv4 and IPv6.

SMTP Traffic: In Wireshark: No SMTP packets were captured, meaning no email activity occurred.

In Pie Charts: SMTP is absent, consistent with Wireshark.

Conclusion: DNS traffic is correctly captured and categorized, SMTP traffic is absent, matching the expected activity.

## 7.2 TESTING REPORT

Table 7.1 Testing Report

Test ID	Description	Steps	Expected Output	Result
TC-01	Capture network packets	<ul style="list-style-type: none"> <li>- Start the application</li> <li>- Generate network traffic (e.g., open a website)</li> </ul>	Packets should be captured successfully and sent to nfqueue.	Pass
TC-02	Handle no network traffic	<ul style="list-style-type: none"> <li>- Start the application</li> <li>- Ensure no active network traffic</li> </ul>	The system should wait without errors and not crash.	Pass
TC-03	Redirect packets to nfqueue	<ul style="list-style-type: none"> <li>- Set up iptables rules</li> <li>- Start the application</li> <li>- Check nfqueue</li> </ul>	Packets should be redirected to nfqueue without loss or errors.	Pass
TC-04	Remove iptables rules	<ul style="list-style-type: none"> <li>- Remove iptables rules</li> <li>- Generate network traffic</li> </ul>	No packets should be processed by the application.	Pass

TC-05	Identify common protocols	<ul style="list-style-type: none"> <li>- Capture packets (e.g., HTTP, HTTPS, DNS)</li> <li>- Start application</li> </ul>	Protocols should be correctly identified.	Pass
TC-06	Handle unknown protocols	<ul style="list-style-type: none"> <li>- Capture packets with unknown protocol types</li> </ul>	Unknown packets should be discarded or flagged without crashing.	Pass
TC-07	Classify packets by protocol	<ul style="list-style-type: none"> <li>- Capture packets</li> <li>- Start application</li> </ul>	Packets should be classified and stored correctly by protocol type.	Pass
TC-08	Display real-time statistics	<ul style="list-style-type: none"> <li>- Capture packets</li> <li>- Check traffic visualization (e.g., pie chart)</li> </ul>	The pie chart should update in real time with accurate data.	Pass

## 7.3 TESTING TOOL

Below is the adapted version of testing tools specifically tailored for your **Traffic Categorization Project** using Python, iptables, nftables, and real-time visualization:

### 1. Pytest (Python Testing Framework)

#### ○ Purpose:

- To test the Python scripts used for packet classification and visualization (e.g., `classify_packet`, `update_pie_charts`).

#### ○ Example Use Cases:

- Validate the packet parsing logic.
- Check if application, transport, and network layer statistics are updated correctly.

## 2. Wireshark (Traffic Analysis)

- **Purpose:**
  - To analyze and verify packet flow between iptables and nfqueue.
- **Example Use Cases:**
  - Monitor whether the packets are being correctly captured and forwarded to the Python script.
  - Verify that specific traffic (e.g., HTTP, HTTPS) is being categorized accurately.

## 3. NetfilterQueue Debugging

- **Purpose:**
  - Test the integration of iptables and the Python script.
- **Example Use Cases:**
  - Validate that iptables rules are directing packets to nfqueue.
  - Ensure the Python script processes packets in real-time without dropping them.

## 4. Matplotlib Testing

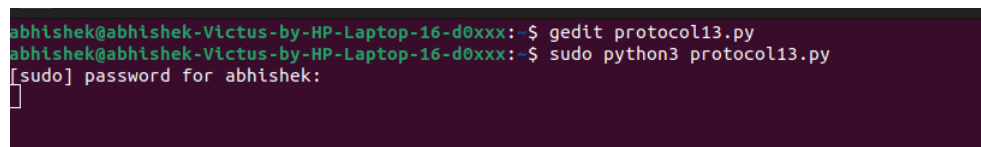
- **Purpose:**
  - Validate the correctness of pie chart updates for real-time visualization.
- **Example Use Cases:**
  - Check that the total percentage across all charts equals 100%.
  - Verify that the displayed labels correspond to the collected statistics.

## 5. Confusion Matrix and Metrics Analysis (Optional)

- **Purpose:**
  - To evaluate the accuracy of packet classification if a machine learning model is added later.
- **Example Use Cases:**
  - Compare the predicted protocol type against actual packet data

## RESULTS AND OUTCOMES

### Snapshots of the Project Execution



```
abhishek@abhishek-Victus-by-HP-Laptop-16-d0xxx:~$ gedit protocol13.py
abhishek@abhishek-Victus-by-HP-Laptop-16-d0xxx:~$ sudo python3 protocol13.py
[sudo] password for abhishek:

```

Fig 7.2 Running the Python script

In Fig. 7.2 The image shows the execution of the protocol13.py script in Ubuntu using the command `sudo python3 protocol13.py`. This runs the Python script with root privileges to capture and classify network traffic by protocol (TCP, UDP, ICMP) in real-time, potentially displaying the results in the terminal.



```
abhishek@abhishek-Victus-by-HP-Laptop-16-d0xxx:~$ sudo iptables -L
[sudo] password for abhishek:
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
abhishek@abhishek-Victus-by-HP-Laptop-16-d0xxx:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

```

Fig 7.3 Checking the iptables rules

In Fig 7.3 Sudo ip6tables -L lists the rules in the IPv6 firewall.

Displays three default chains: INPUT, FORWARD, and OUTPUT.

All policies are set to ACCEPT, and there are no additional rules configured.

Sudo iptables -L lists the rules in the IPv4 firewall.

Similar output to the IPv6 command, showing the default chains with policies set to ACCEPT.

No custom rules are added.

This indicates that both IPv4 and IPv6 firewalls currently have no active custom rules, and all traffic is allowed (default ACCEPT policy).

```

abhishek@abhishek-Victus-by-HP-Laptop-16-d0xxx:~$ sudo ip6tables -A INPUT -j NFQUEUE --queue-num 1
abhishek@abhishek-Victus-by-HP-Laptop-16-d0xxx:~$ sudo iptables -A INPUT -j NFQUEUE --queue-num 1
abhishek@abhishek-Victus-by-HP-Laptop-16-d0xxx:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
NFQUEUE    all  --  anywhere               anywhere               NFQUEUE num 1

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
abhishek@abhishek-Victus-by-HP-Laptop-16-d0xxx:~$ sudo ip6tables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
NFQUEUE    all  --  anywhere               anywhere               NFQUEUE num 1

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
abhishek@abhishek-Victus-by-HP-Laptop-16-d0xxx:~$

```

Fig 7.4 Adding Iptables rules

In Fig 7.4 Commands executed: Sudo iptables -A INPUT -j NFQUEUE --queue-num 1

Adds a rule to the IPv4 firewall in the INPUT chain, directing all incoming packets to NFQUEUE number 1.

Sudo ip6tables -A INPUT -j NFQUEUE --queue-num 1

Adds a similar rule for the IPv6 firewall in the INPUT chain.

Sudo iptables -L output: Shows the updated INPUT chain for IPv4 with the NFQUEUE target, directing packets to queue number 1.

Sudo ip6tables -L output: Similarly, displays the updated INPUT chain for IPv6 with the NFQUEUE target, sending packets to queue number 1.



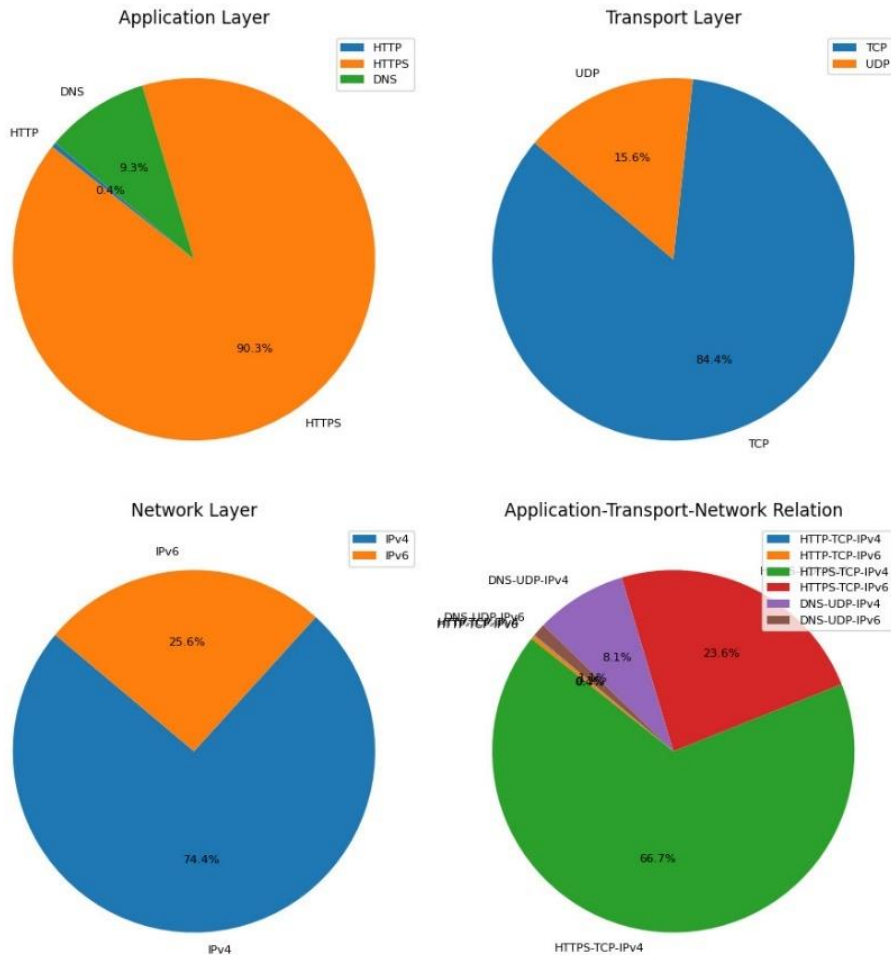


Fig 7.5 Pie chart

In Fig 7.5 The image shows four pie charts visualizing network packet classification across different protocol layers: Application Layer, Transport Layer, Network Layer, and a combined relationship of Application-Transport-Network layers.

#### Application Layer:

- HTTPS dominates with 90.3% of the traffic.
- HTTP accounts for 9.3%.
- DNS has a minimal share of 0.4%.

#### Transport Layer:

- TCP is the major protocol, representing 84.4% of the traffic.

- UDP accounts for 15.6%.

Network Layer:

- IPv4 is prevalent, contributing 74.4% of the traffic.
- IPv6 makes up 25.6%.

Application-Transport-Network Relation:

- Shows the combined distribution:
- HTTPS-TCP-IPv4 is the largest group at 66.7%.
- HTTPS-TCP-IPv6 follows with 23.6%.
- Other combinations (e.g., DNS-UDP-IPv4, HTTP-TCP-IPv4) have smaller shares, each under 8.1%.

## CONCLUSION

The testing process ensures that all components of the Traffic Categorization Project work seamlessly. Pytest validates Python logic, Wireshark confirms packet flow correctness, and debugging utilities for Net filter Queue ensure packet handling accuracy. Comprehensive testing guarantees real-time traffic classification and visualization function reliably under varying conditions.

## REFERENCES

- [1] Velan, P., Čermák, M., Čeleda, P., & Drasar, M. (2015). A survey of methods for encrypted traffic classification and analysis. *International Journal of Network Management*, 25(5), 355-374.
- [2] Khalife, H., Yuan, D., & Lahoud, S. (2014). Towards an optimal traffic classification model: A taxonomy for encrypted and obfuscated traffic. *IEEE Communications Surveys & Tutorials*, 16(4), 1946-1970.
- [3] Piskač, R. (2012). Network traffic classification based on time characteristics analysis. Master's Thesis, Faculty of Electrical Engineering and Computing, University of Zagreb.
- [4] Valenti, S., Rossi, D., & Dainotti, A. (2013). Reviewing traffic classification. *Proceedings of the International Conference on Communication Systems and Networks*, 1-7.