Vivekanand Education Society's Institute Of Technology

Department Of Information Technology

DSA miniProject

A.Y. 2025-26

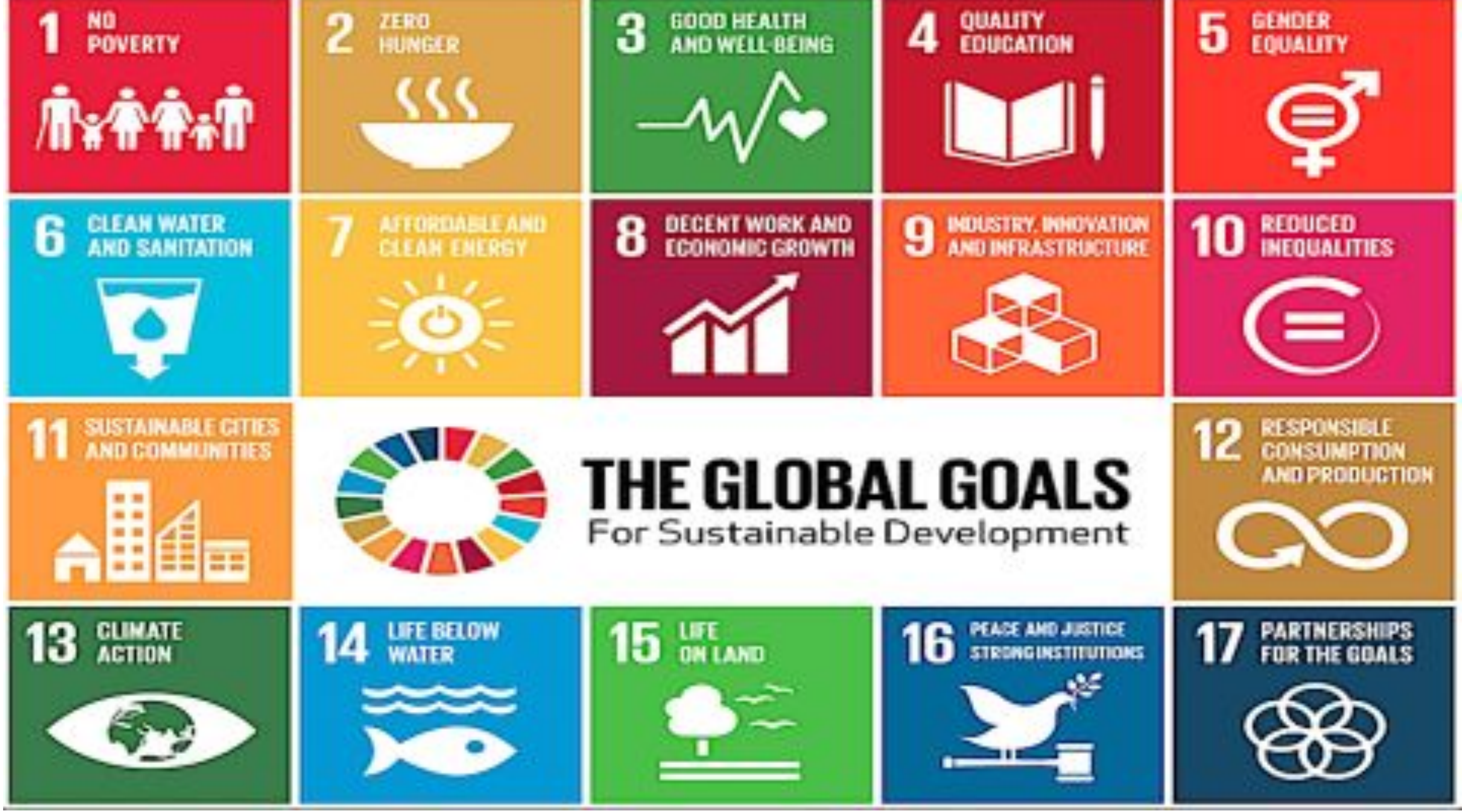
Title:Network Packet Analyser

Sustainability Goal : Contributing to open source Network Security and Energy Aware Processing

Domain: **Data Structures & Algorithms**

Member: **Rishabh Kumar**

Mentor Name: **Kajal Jewani**



1 NO POVERTY

2 ZERO HUNGER

3 GOOD HEALTH AND WELL-BEING

4 QUALITY EDUCATION

5 GENDER EQUALITY

6 CLEAN WATER AND SANITATION

7 AFFORDABLE AND CLEAN ENERGY

8 DECENT WORK AND ECONOMIC GROWTH

9 INDUSTRY, INNOVATION AND INFRASTRUCTURE

10 REDUCED INEQUALITIES

11 SUSTAINABLE CITIES AND COMMUNITIES

THE GLOBAL GOALS

For Sustainable Development

12 RESPONSIBLE CONSUMPTION AND PRODUCTION

13 CLIMATE ACTION

14 LIFE BELOW WATER

15 LIFE ON LAND

16 PEACE AND JUSTICE STRONG INSTITUTIONS

17 PARTNERSHIPS FOR THE GOALS

Introduction to Project

This project implements a Network Packet Analyser using Python, leveraging core Data Structures and Algorithms (DSA). It functions as a custom tool to capture, process, and inspect data packets travelling across a network in real-time.Key operations include packet capture via socket programming, efficient parsing of protocol headers (like Ethernet, IP, TCP) using structured data types, and filtering/statistics managed by algorithms for quick data retrieval and organisation.

This project demonstrates the practical use of Python and DSA in low-level network programming and cybersecurity.

Content

1.Introduction to the Project

2.Problem Statement

3.Objectives of the Project

4.Scope of the Project

5.Requirements of the System (Hardware, Software)

6.ER Diagram of the Proposed System

7.Data Structure & Concepts Used

8.Algorithm Explanation

9.Time and Space Complexity

10.Front End

11.Implementation

12.Gantt Chart

13. Test Cases

14.Challenges and Solutions

15. Future Scope

16. Code

17. Output Screenshots

18. Conclusion

Problem Statement

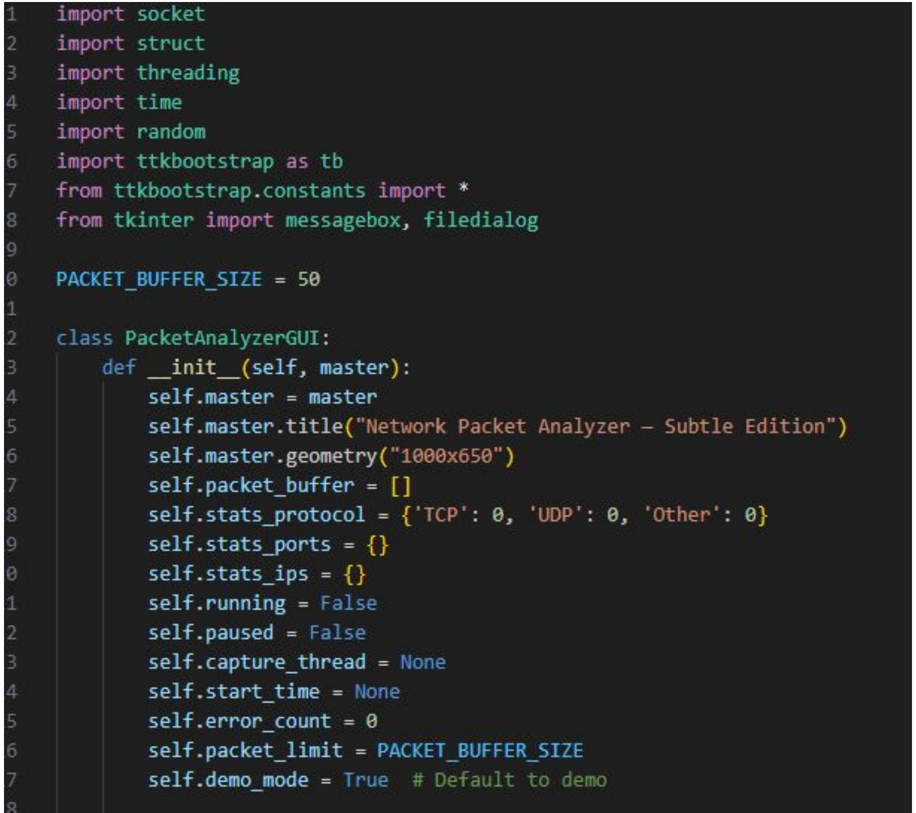
Manual network monitoring is impractical for large-scale networks.
Analyzing packet data efficiently requires:

- Real-time processing of high-volume traffic
- Memory-efficient storage of packet data
- Quick statistical analysis and pattern recognition
- Hierarchical organization of network data

Objectives of the project

- Implement a simulated packet capture system using DSA concepts
- Use Circular Buffer for efficient memory management
- Apply Hash Tables for O(1) protocol statistics
- Organize packets hierarchically using Tree structures
- Demonstrate real-time analysis and search capabilities

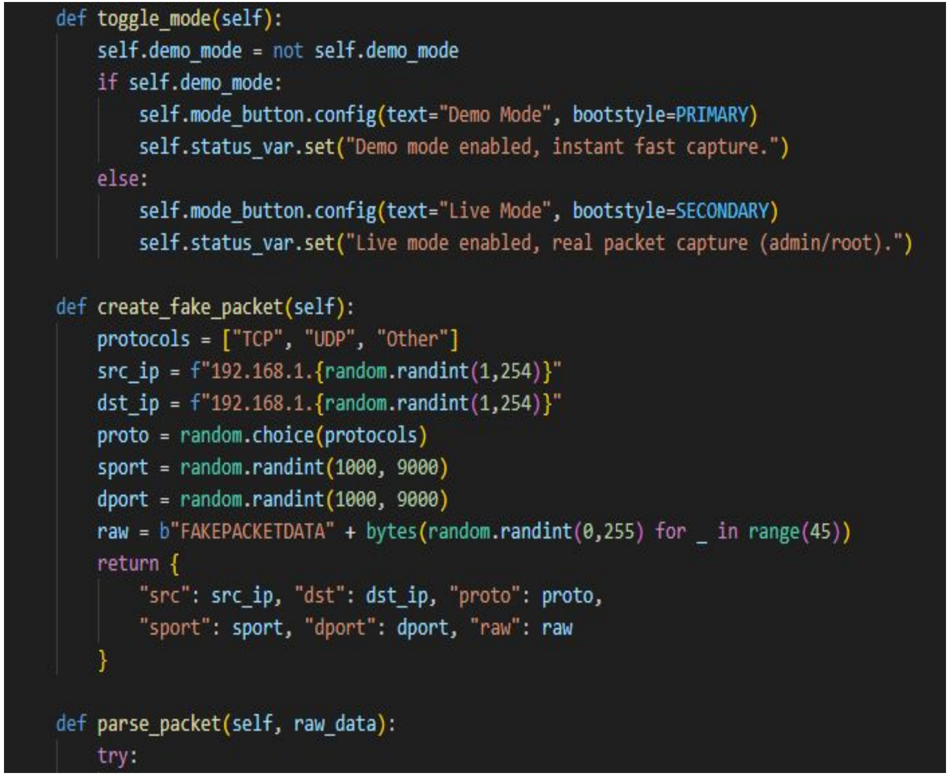
Implementation



```python
import socket
import struct
import threading
import time
import random
import ttkbootstrap as tb
from ttkbootstrap.constants import *
from tkinter import messagebox, filedialog

PACKET_BUFFER_SIZE = 50

class PacketAnalyzerGUI:
    def __init__(self, master):
        self.master = master
        self.master.title("Network Packet Analyzer – Subtle Edition")
        self.master.geometry("1000x650")
        self.packet_buffer = []
        self.stats_protocol = {'TCP': 0, 'UDP': 0, 'Other': 0}
        self.stats_ports = {}
        self.stats_ips = {}
        self.running = False
        self.paused = False
        self.capture_thread = None
        self.start_time = None
        self.error_count = 0
        self.packet_limit = PACKET_BUFFER_SIZE
        self.demo_mode = True  # Default to demo
```

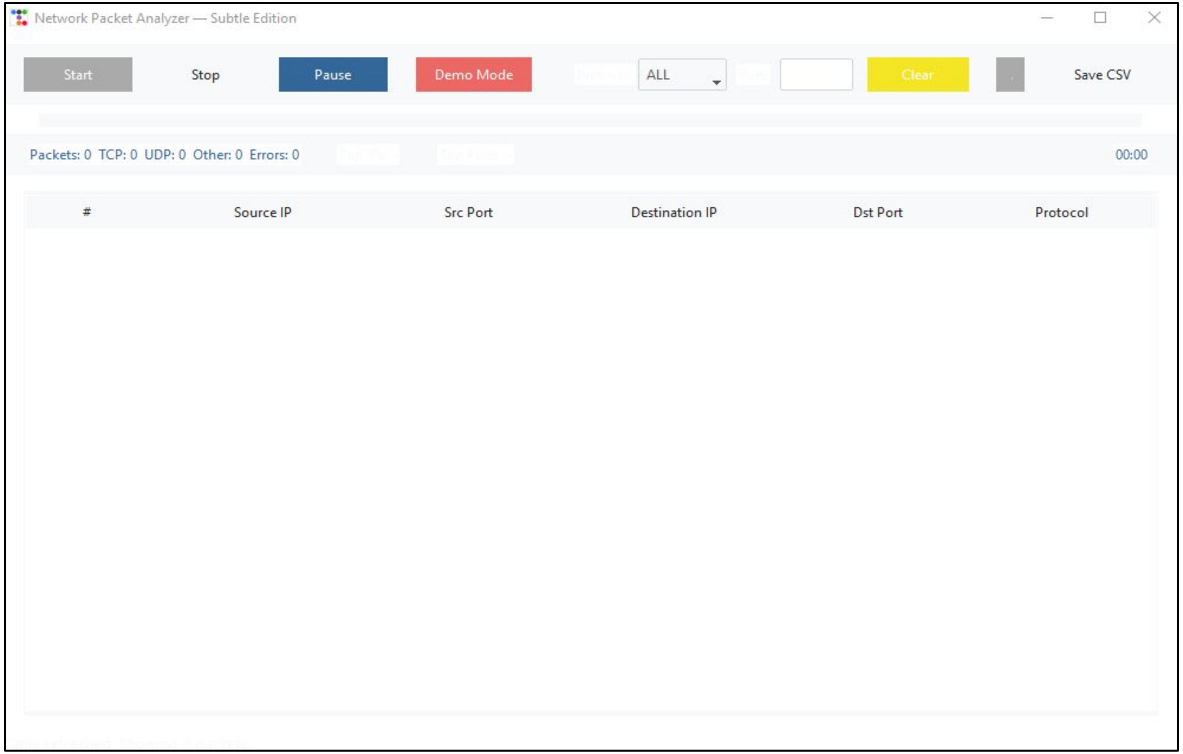


```python
def toggle_mode(self):
    self.demo_mode = not self.demo_mode
    if self.demo_mode:
        self.mode_button.config(text="Demo Mode", bootstyle=PRIMARY)
        self.status_var.set("Demo mode enabled, instant fast capture.")
    else:
        self.mode_button.config(text="Live Mode", bootstyle=SECONDARY)
        self.status_var.set("Live mode enabled, real packet capture (admin/root).")

def create_fake_packet(self):
    protocols = ["TCP", "UDP", "Other"]
    src_ip = f"192.168.1.{random.randint(1,254)}"
    dst_ip = f"192.168.1.{random.randint(1,254)}"
    proto = random.choice(protocols)
    sport = random.randint(1000, 9000)
    dport = random.randint(1000, 9000)
    raw = b"FAKEPACKETDATA" + bytes(random.randint(0,255) for _ in range(45))
    return {
        "src": src_ip, "dst": dst_ip, "proto": proto,
        "sport": sport, "dport": dport, "raw": raw
    }

def parse_packet(self, raw_data):
    try:
```

Outputs

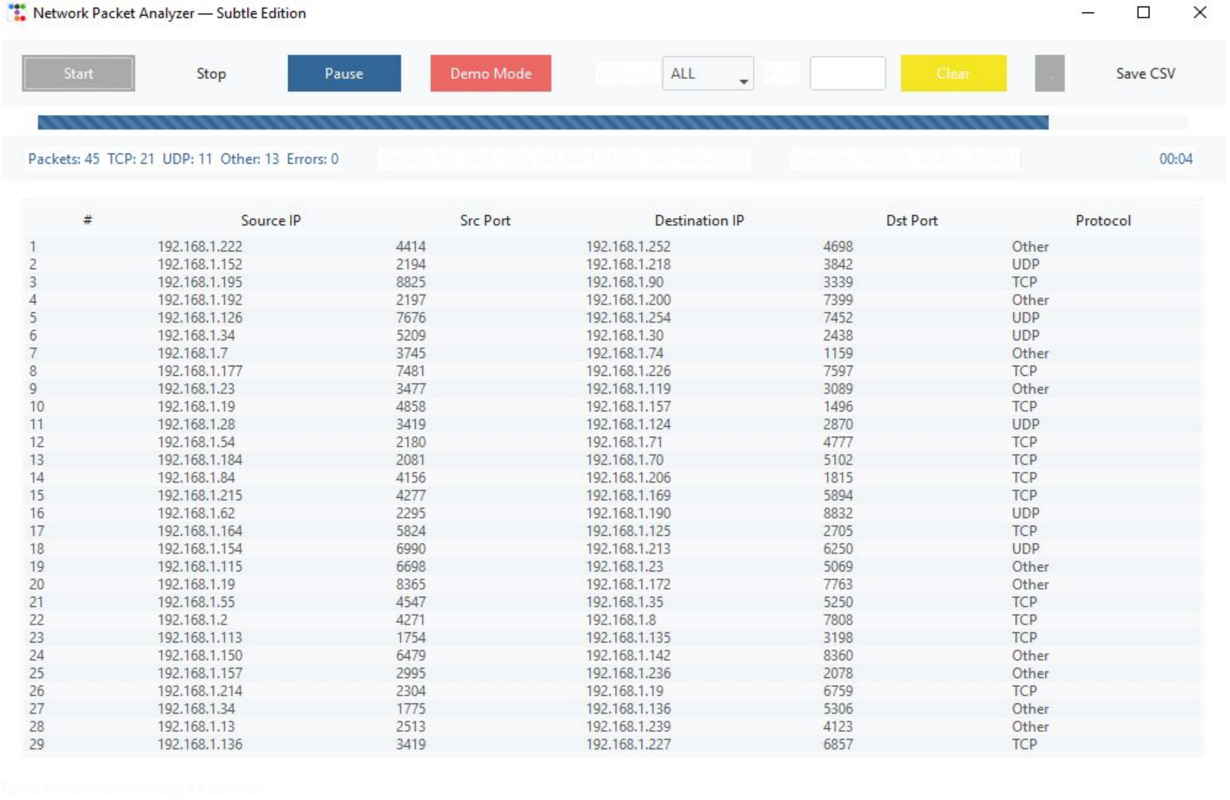


Packets: 0 TCP: 0 UDP: 0 Other: 0 Errors: 0

#	Source IP	Src Port	Destination IP	Dst Port	Protocol

Table refreshed. Showing 0 packets.

Outputs



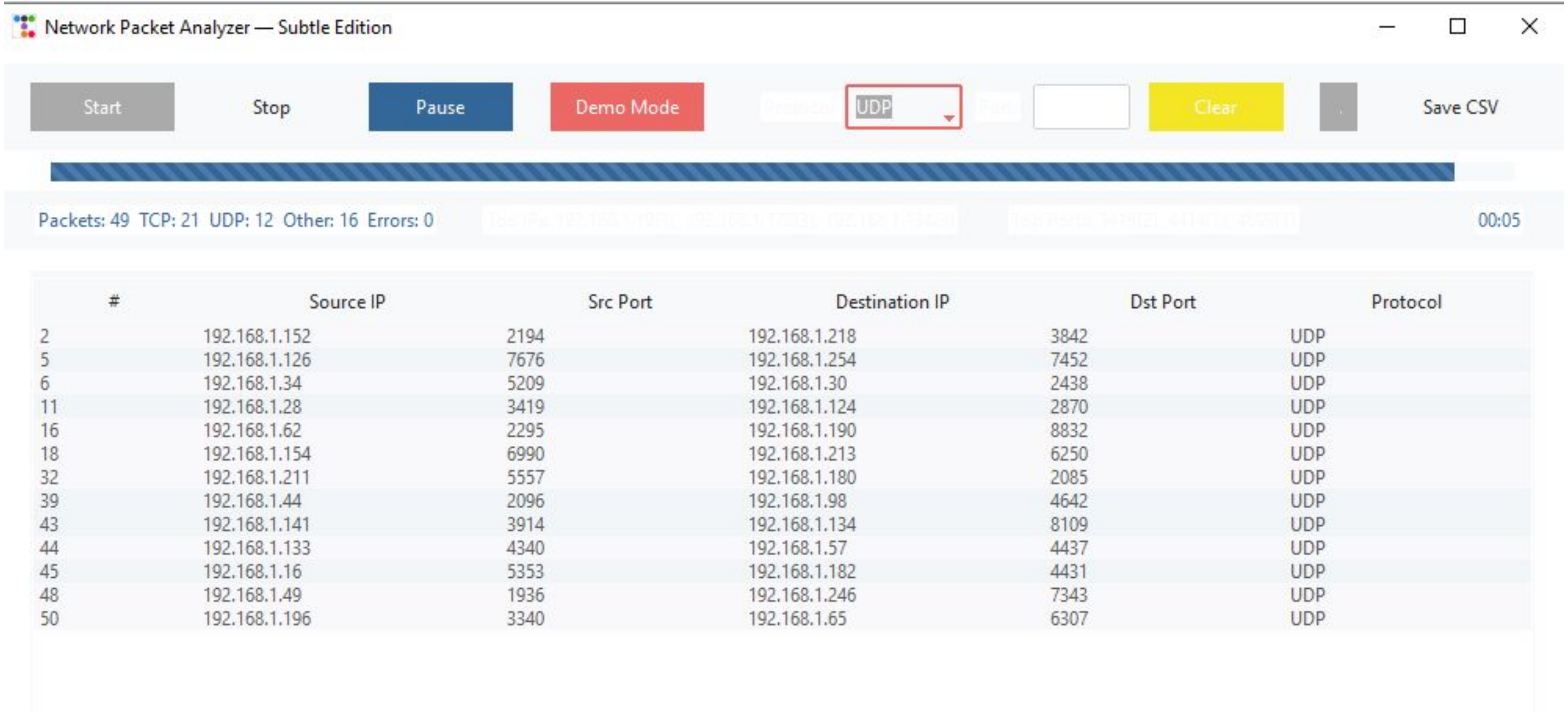
Packets: 45 TCP: 21 UDP: 11 Other: 13 Errors: 0

00:04

#	Source IP	Src Port	Destination IP	Dst Port	Protocol
1	192.168.1.222	4414	192.168.1.252	4698	Other
2	192.168.1.152	2194	192.168.1.218	3842	UDP
3	192.168.1.195	8825	192.168.1.90	3339	TCP
4	192.168.1.192	2197	192.168.1.200	7399	Other
5	192.168.1.126	7676	192.168.1.254	7452	UDP
6	192.168.1.34	5209	192.168.1.30	2438	UDP
7	192.168.1.7	3745	192.168.1.74	1159	Other
8	192.168.1.177	7481	192.168.1.226	7597	TCP
9	192.168.1.23	3477	192.168.1.119	3089	Other
10	192.168.1.19	4858	192.168.1.157	1496	TCP
11	192.168.1.28	3419	192.168.1.124	2870	UDP
12	192.168.1.54	2180	192.168.1.71	4777	TCP
13	192.168.1.184	2081	192.168.1.70	5102	TCP
14	192.168.1.84	4156	192.168.1.206	1815	TCP
15	192.168.1.215	4277	192.168.1.169	5894	TCP
16	192.168.1.62	2295	192.168.1.190	8832	UDP
17	192.168.1.164	5824	192.168.1.125	2705	TCP
18	192.168.1.154	6990	192.168.1.213	6250	UDP
19	192.168.1.115	6698	192.168.1.23	5069	Other
20	192.168.1.19	8365	192.168.1.172	7763	Other
21	192.168.1.55	4547	192.168.1.35	5250	TCP
22	192.168.1.2	4271	192.168.1.8	7808	TCP
23	192.168.1.113	1754	192.168.1.135	3198	TCP
24	192.168.1.150	6479	192.168.1.142	8360	Other
25	192.168.1.157	2995	192.168.1.236	2078	Other
26	192.168.1.214	2304	192.168.1.19	6759	TCP
27	192.168.1.34	1775	192.168.1.136	5306	Other
28	192.168.1.13	2513	192.168.1.239	4123	Other
29	192.168.1.136	3419	192.168.1.227	6857	TCP

Table refreshed. Showing 45 packets.

Outputs



Packets: 49 TCP: 21 UDP: 12 Other: 16 Errors: 0

00:05

#	Source IP	Src Port	Destination IP	Dst Port	Protocol
2	192.168.1.152	2194	192.168.1.218	3842	UDP
5	192.168.1.126	7676	192.168.1.254	7452	UDP
6	192.168.1.34	5209	192.168.1.30	2438	UDP
11	192.168.1.28	3419	192.168.1.124	2870	UDP
16	192.168.1.62	2295	192.168.1.190	8832	UDP
18	192.168.1.154	6990	192.168.1.213	6250	UDP
32	192.168.1.211	5557	192.168.1.180	2085	UDP
39	192.168.1.44	2096	192.168.1.98	4642	UDP
43	192.168.1.141	3914	192.168.1.134	8109	UDP
44	192.168.1.133	4340	192.168.1.57	4437	UDP
45	192.168.1.16	5353	192.168.1.182	4431	UDP
48	192.168.1.49	1936	192.168.1.246	7343	UDP
50	192.168.1.196	3340	192.168.1.65	6307	UDP

Gantt Chart

Task	Week 1	Week 2	Week 3
Requirement Analysis	✔		
Design (ER/Flowch	✔		
Coding		✔	
Testing & Debugging		✔	
Documentation		✔	

Test Case	Input	Expected Output	Result Pass
Add Customer	Name=Raj, Service=Haircut	Added successfully	Pass
Serve Customer	Queue not empty	First customer served	Pass
Serve Customer	Queue empty	“No customers” message	Pass
View Queue	3 customers	Display all in order	

Conclusion

What I Learned:

- DSA is not just academic - it's the foundation of efficient software systems
- Network analysis requires smart data organization for real-time processing
- Algorithm efficiency directly impacts application performance

Future Enhancements:

- Real-time intrusion detection systems
- Advanced traffic visualization dashboard
- Deep packet inspection for security analysis
- Machine learning integration for anomaly detection

This Network Packet Analyzer is more than just a project - it's proof that when we master fundamental data structures and algorithms, we gain the power to not just use technology, but to understand, analyze, and secure the digital world around us. Thank you.

References

1 A. S. Tanenbaum and D. J. Wetherall, Computer Networks, 5th ed. Upper Saddle River, NJ, USA: Prentice Hall, 2011.

2 E. Horowitz, S. Sahni, and S. Anderson-Freed, Fundamentals of Data Structures in C, 2nd ed. Silicon Press, 2008

3 DCCN Course Material