A blue circle with white text and a letter

Description automatically generated

**COMPUTER NETWORK ASSIGNMENT**

**Introduction**

The Network Traffic Monitoring System is a comprehensive solution for real-time network analysis, combining low-level packet capture capabilities with an intuitive web-based visualization interface. This system provides detailed insights into network traffic patterns, protocol distributions, and connection statistics.

**Project Objectives**

- Real-time network traffic monitoring

- Protocol-level traffic analysis

- User-friendly visualization of network metrics

- Efficient data storage and retrieval

**System Requirements**

- Windows operating system with Npcap installed

- C++ development environment

- Python with Flask framework

- SQLite database

**System Architecture**

A diagram of a software server

Description automatically generated

Figure: System Architecture Diagram showing component interaction

The system consists of three main components:

1. Packet Capture Engine (C++)

2. Data Storage Layer (SQLite)

3. Web Visualization Interface (Flask/HTML)

**Component Details**

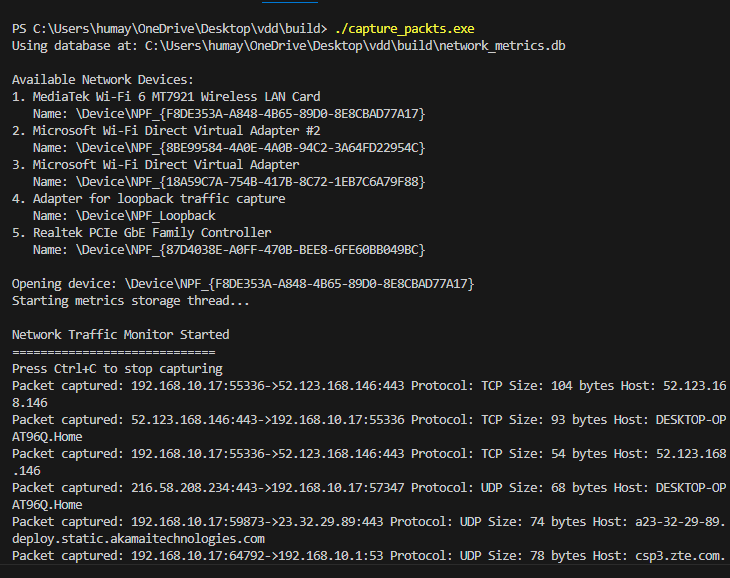
**Packet Capture Engine**

- Built in C++ using WinPcap/Npcap library

- Performs real-time packet analysis

- Calculates network metrics

- Handles protocol detection



A screenshot of a computer

Description automatically generated

Figures: Packet Capture Engine in Operation

**Data Storage Layer**

- SQLite database for metric storage

- Real-time data updates

- Thread-safe operations

- Efficient query handling

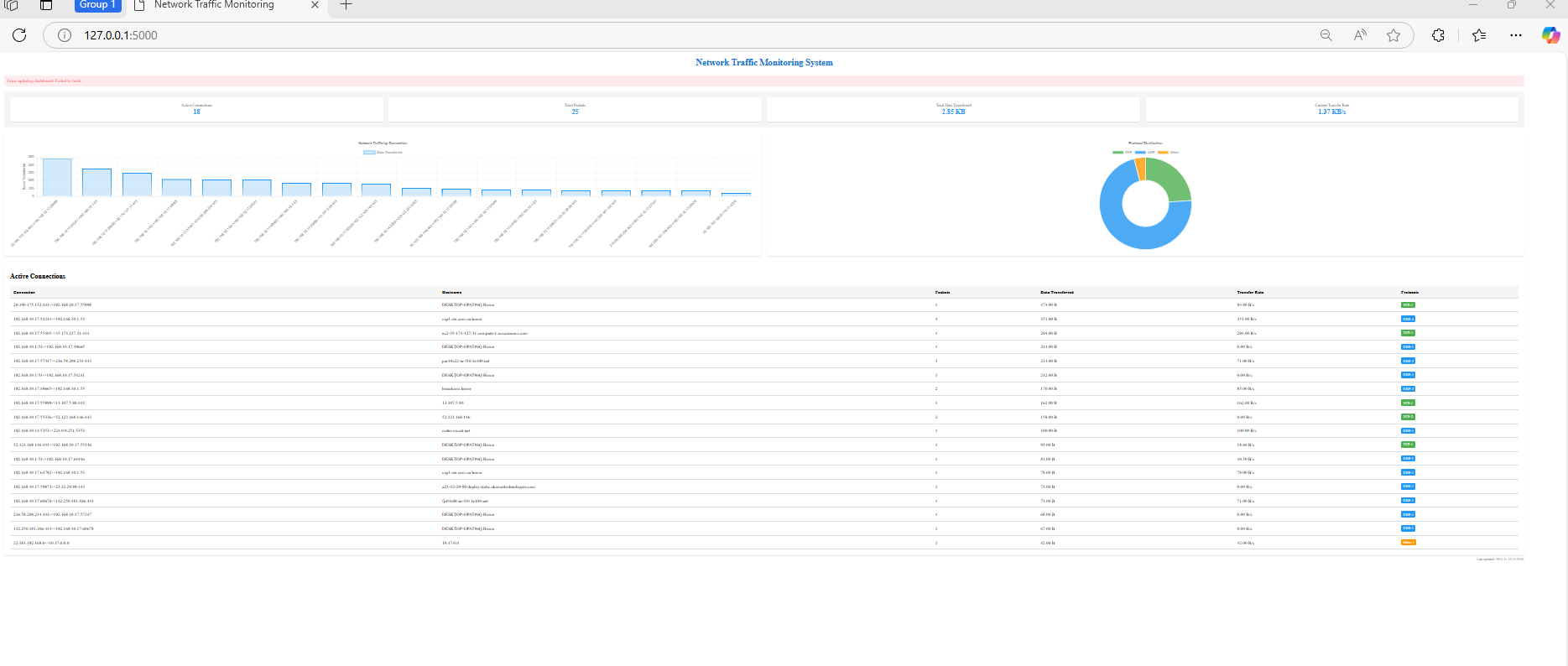
**Web Visualization Interface**

- Flask-based backend

- Real-time dashboard

- Interactive charts and tables

- Responsive design



A screenshot of a computer

Description automatically generatedFigures: Web Interface Dashboard

**Implementation Details**

**Packet Capture Component**

//cpp

*// Key data structures*

struct ProtocolMetrics {

int tcp\_count = 0;

int udp\_count = 0;

int other\_count = 0;

std::string hostname;

int bytes\_transferred = 0;

};

struct ConnectionMetrics {

int packet\_count = 0;

ProtocolMetrics protocol\_metrics;

float bytes\_per\_second = 0.0;

};

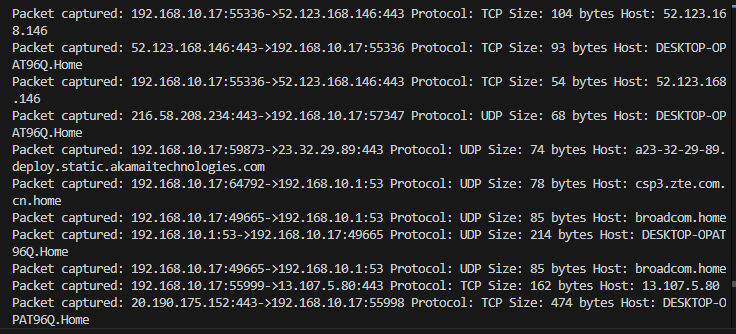


Figure: Real-time Packet Capture Results

**Web Interface Implementation**

```python

def fetch\_metrics():

try:

with sqlite3.connect(DB\_PATH) as conn:

conn.row\_factory = sqlite3.Row

cursor = conn.cursor()

*# Fetch and process metrics*

return processed\_metrics

except Exception as e:

log\_error(e)

return default\_metrics

```

**Component Interaction**

**Data Flow Process**

**1. Packet Capture**

- Network interface monitoring

- Protocol detection

- Metric calculation

**2. Data Storage**

- Real-time updates

- Efficient indexing

- Concurrent access handling

**3. Web Visualization**

- Regular data polling

- Dynamic updates

- User interaction handling

**Communication Protocols**

- Database interactions

- HTTP/JSON data exchange

- WebSocket updates

**Conclusion**

**Achievements**

- Successful implementation of real-time monitoring

- Efficient protocol detection

- Responsive user interface

- Reliable data storage

*# Compilation Command*

*#To build c++ code*

**Ctrl + shift + b**

*# Running the System*

**./capture\_packts.exe**

#To run flask

**python app.py**