CN Lab Project Report

Session: 2022-2026

Submitted by:

Tayyaba Afzal	2022-CS-134
Raveeha Mohsin	2022-CS-149

Submitted to:

Syed Tehseen Ul Hasan Shah

Department of Computer Science
University of Engineering & Technology
Lahore, Pakistan

Contents

List of Figures

List of Tables

Network Packet Analyzer

1 Introduction

The ”Network Packet Analyzer” is an application designed to capture, analyze, and display network traffic data. It provides real-time monitoring and detailed information about network packets transmitted over the network. This project aims to help network administrators, security professionals, and developers analyze network traffic to identify potential issues and monitor network performance.

1.1 Purpose and Significance

The purpose of this project is to develop a tool for monitoring network traffic and analyzing packet-level information. It plays a significant role in network troubleshooting, security analysis, and performance optimization. By capturing packets and examining protocol details, users can:

- Detect network anomalies and bottlenecks.
- Troubleshoot connectivity issues.
- Ensure data security by inspecting suspicious traffic patterns.

1.2 Problem Being Addressed

In large-scale networks, it is often difficult to detect potential issues such as network congestion, unauthorized traffic, or performance degradation. Existing tools may be complex or not provide adequate real-time monitoring. This project addresses these challenges by offering an intuitive user interface with essential features such as port scanning, packet sniffing, and real-time traffic analysis.

1.3 Motivation Behind the Project

The motivation behind this project stems from the growing need for efficient network monitoring tools. With increasing reliance on networks for communication, security, and performance management, it is crucial to have tools that allow users to monitor, analyze, and troubleshoot network traffic in real time. By providing an easy-to-use solution, the project aims to make network analysis accessible for both beginners and professionals.

2 Project Description

The ”Network Packet Analyzer” project aims to develop a comprehensive tool for monitoring and analyzing network traffic. This section outlines the goals and objectives, scope of the work, and provides relevant context for the project.

2.1 Goals and Objectives

The table 1 highlights the key goals and objectives of the project:

Category	Details
Goals	1. Develop a real-time packet capturing and monitoring tool. 2. Support multiple network protocols (TCP, UDP, ARP, IP, ICMP, DHCP, DNS). 3. Enhance network troubleshooting by providing detailed packet information.
Objectives	1. Implement a port scanner and ping sniffer. 2. Analyze active and closed network connections in real time. 3. Provide an intuitive user interface for packet inspection.

Table 1: Goals and Objectives of the Network Packet Analyzer

2.2 Scope of the Work

The scope of this project is defined as follows:

- **Included:**
  - Real-time packet capturing and analysis.
  - Support for multiple protocols (TCP, UDP, ARP, etc.).
  - Real-time event tracking for new and ended connections.
- **Excluded:**
  - Advanced intrusion detection and prevention mechanisms.
  - Comprehensive forensic analysis of past network traffic.
  - Cloud-based monitoring or large-scale network support.

2.3 Context and Background Information

Network monitoring tools are essential for maintaining network health, ensuring security, and troubleshooting issues. Traditional tools may not always provide real-time insights or may be difficult for non-experts to use. This project bridges the gap by delivering a user-friendly, real-time solution for packet analysis, making it accessible for both professionals and learners.

The tool also provides visibility into various protocols and helps users identify bottlenecks, detect unauthorized traffic, and optimize network performance effectively.

3 Methodology Used

This section outlines the methodology and approach followed to complete the ”Network Packet Analyzer” project. The methodology includes step-by-step procedures, tools, and techniques utilized for packet capturing, analysis, and display.

3.1 Overview of Methodology

- Understanding requirements for network packet analysis and monitoring.
- Identifying key protocols to be supported (TCP, UDP, ARP, IP, ICMP, DHCP, DNS).
- Designing a modular architecture to separate packet capturing, decoding, and display functionalities.
- Implementing packet capturing using system-level APIs.
- Developing analysis logic for interpreting packet data.
- Building a user interface to display real-time results.

3.2 Steps Followed

Step	**Description**
Requirement Analysis	Identified the need for real-time network packet monitoring and connection tracking across multiple protocols.
Tool Selection	Used `iphlpapi.dll` to capture packet information and `MIB` tables for TCP, UDP, ARP, and IPNET data.
Connection Monitoring	Developed a system to track TCP, UDP, ARP, and IPNET connections and update connection lists in real-time.
Connection Filtering	Implemented filtering for specific protocols (TCP, UDP, ARP, IPNET) based on user-defined criteria, allowing efficient connection monitoring.
Real-Time Updates	Used `BackgroundWorker` for non-blocking updates, ensuring UI responsiveness and real-time packet display.
Event Handling	Integrated event-driven architecture to notify UI components when new packets are detected or connections end.
Testing and Optimization	Optimized packet retrieval and connection monitoring for better performance, ensuring accuracy and real-time responsiveness.

Table 2: Steps in the Methodology

Figure 1 illustrates the step-by-step methodology adopted in the development and implementation of the project. This diagram highlights the sequential phases, starting from requirement analysis and tool selection to the final evaluation and testing phase.

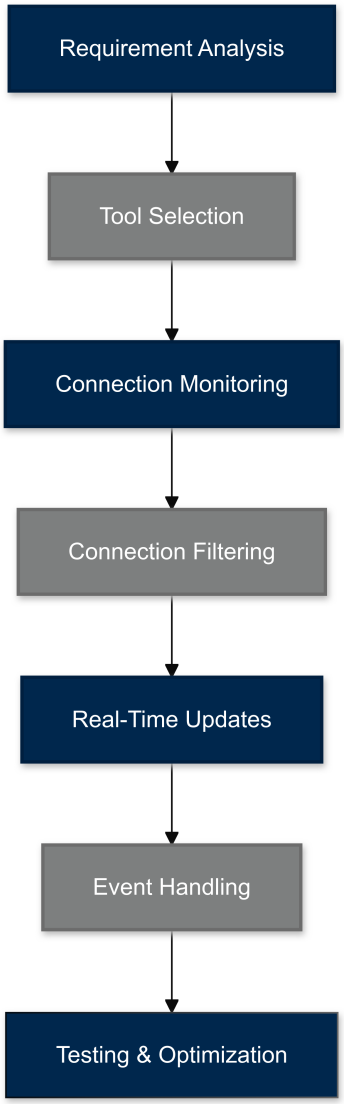

Figure 1: Approach Followed

4 Key Features and Functionality

This section details the core features and functionalities implemented in the ”Network Packet Analyzer” project. The focus is on providing a clear understanding of how users interact with the system and highlighting its unique aspects.

4.1 Real-Time Monitoring

The real-time monitoring feature provides a dynamic overview of network activity, displaying active connections and their associated data in real time. This allows users to quickly identify network traffic patterns, potential bottlenecks, and suspicious activity. The following information is displayed for each active connection:

- **Protocol:** The network protocol used for the connection (e.g., TCP, UDP, ICMP , IPNET , ARP etc).

- **Local Address:** The IP address and port of the local machine involved in the connection.
- **Remote Address:** The IP address and port of the remote machine involved in the connection.
- **State:** The current state of the connection (e.g., ESTABLISHED, TIME_WAIT, CLOSE_WAIT). This provides insights into the connection lifecycle.
- **Process ID:** The unique process ID assigned to it.
- **Process Name:** The name of the process which connection details are displayed.

The real-time updates ensure that the displayed information is always current, providing a live snapshot of network activity as shown in Figure 2.

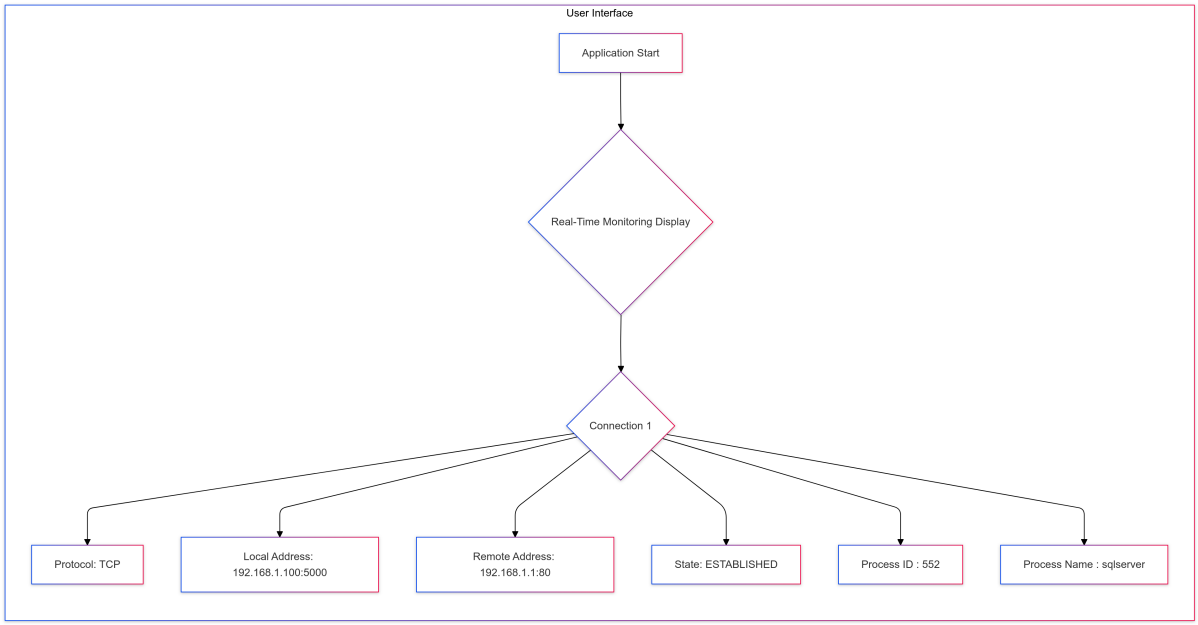

Figure 2: Real Time Monitoring

Example: Upon opening the application, the user observes several connections. One entry shows:

- **Protocol:** TCP
- **Local Address:** 192.168.1.100:5000
- **Remote Address:** 192.168.1.1:80
- **State:** ESTABLISHED
- **Process ID:** 552
- **Process Name:** sqlserver

This indicates an active TCP connection between the local machine (192.168.1.100) on port 5000 and another machine (192.168.1.1) on port 80 (likely HTTP). The ”ESTABLISHED” state confirms that the connection is actively transferring data. Another connection might show a state of ”TIME_WAIT”, indicating that the connection was recently closed but is still in a waiting period before being fully terminated.

4.2 Port Scanner

The integrated port scanner allows users to probe a target host for open ports. This is essential for network security assessments and identifying potential vulnerabilities.

- **Customizable Scan Ranges:** Users can specify the range of ports to scan, allowing for focused or comprehensive scans.
- **TCP and UDP Scans:** The scanner supports both TCP and UDP port scans.
- **Open/Closed/Filtered Status:** The results clearly indicate whether a port is open, closed, or filtered.

The real-time updates ensure that the displayed information is always current, providing a live snapshot of network activity as shown in Figure 3.

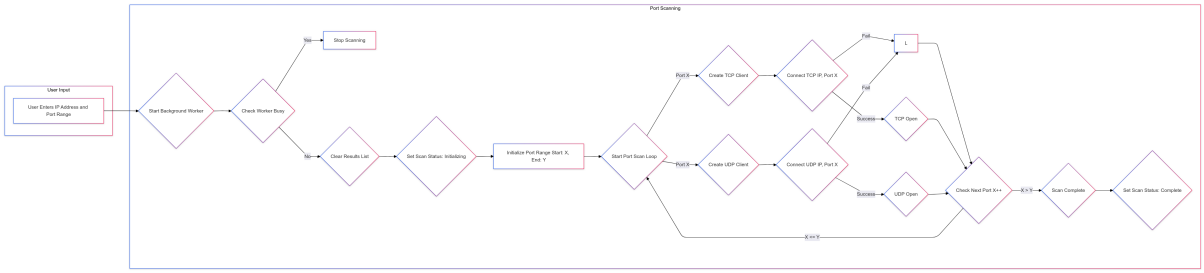

Figure 3: Port Scanner

Example: A user wants to check which connections are open in his target(local) machine. They use the port scanner to scan port.Give the IP Address and range or port. If the scanner reports these ports as ”open,” it confirms that the ports are likely active.

4.3 Ping Sniffer

The Ping Sniffer is a network utility that monitors Internet Control Message Protocol (ICMP) echo requests and replies (commonly known as ping packets.The workflow is exxplained in Figure 4. It provides valuable insights into network connectivity and latency by:

- textbf Real-time Ping Monitoring: Captures and displays ping requests and replies in real time, allowing you to observe network activity as it happens.
- **Latency Measurement:** Calculates and displays the round-trip time (RTT) for each ping packet, indicating the time it takes for a request to reach the target host and receive a response. Higher RTT values suggest potential network congestion or delays.
- **Host Availability Check:** Determines if a specific host is reachable on the network. This is useful for troubleshooting connectivity issues and identifying unreachable devices.

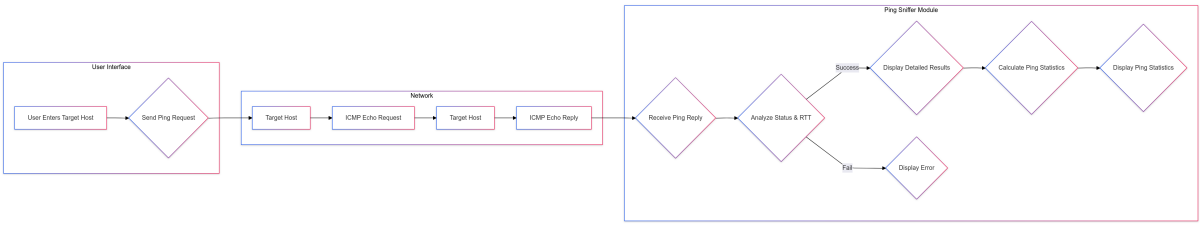

Figure 4: Ping Sniffer

Example: A user suspects a slow internet connection. They use the Ping Sniffer to ping a known reliable server. High RTT values or lost packets could indicate network congestion or problems with the user's internet service provider (ISP).

4.4 Packet Analysis and Filtering

This feature empowers users to dissect captured network traffic and focus on specific data of interest. It offers two primary mechanisms for refining the displayed information: protocol filtering and free-text filtering. Additionally, an analytical dashboard provides a high-level overview of network activity by summarizing connection counts for key protocols.

- **Protocol Filtering:** Users can select specific network protocols (e.g., TCP, UDP, ARP, IP) to isolate packets using those protocols. This allows for targeted analysis of specific types of network communication.
- **Free-Text Filtering:** A text box allows users to enter any string to filter packet information. The system searches packet data (headers, payload) for the entered text and displays only matching packets. This provides flexible filtering based on any content within the packets.
- **Analytical Dashboard:** This dashboard provides a quick overview of network traffic by displaying the count of connections for different protocols (TCP, UDP, ARP, and IP). This helps users quickly identify dominant protocols or potential anomalies in network traffic.

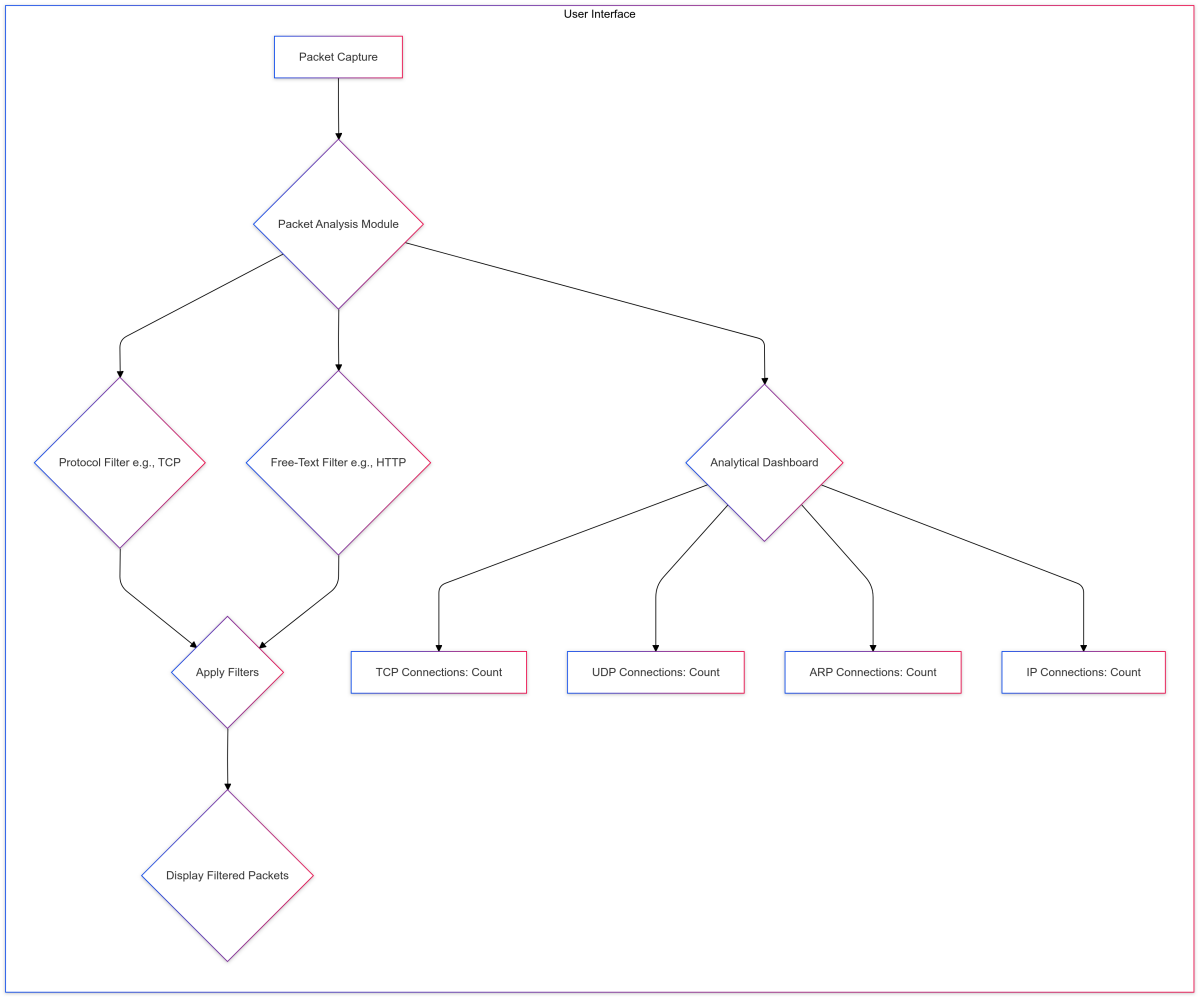

Figure 5: Packet Filter & Analysis

Example: A user might want to analyze only HTTP traffic. They would select ”TCP” as the protocol filter and enter ”HTTP” in the free-text filter. This would display only TCP packets containing ”HTTP” in their data as shown in Figure 5

5 Technologies and Tools Used

This section details the programming languages, frameworks, libraries, software, and platforms employed in the development of the Network Packet Analyzer. Each technology’s contribution and rationale for its selection are provided.
This table provides a comprehensive overview of the technologies and tools used in the project, along with a clear explanation of why each was chosen. The table 3 provides a clear overview of technology used.

Table 3: Technologies and Tools Used

Technology/Tool	**Description**	**Rationale**
.NET Framework 4.7.2+	A software framework developed by Microsoft that provides a managed execution environment for applications.	Chosen for its robust features, extensive libraries, and strong support for Windows desktop application development (WinForms). Version 4.7.2 or higher was selected to ensure access to necessary APIs and security updates.
C#	A modern, object-oriented programming language developed by Microsoft.	Selected as the primary programming language due to its seamless integration with the .NET Framework, strong type system, and rich set of features suitable for network programming and GUI development.
WinForms	A Windows Forms-based user interface framework for creating rich client applications for Windows.	Chosen for creating the application's user interface due to its simplicity, ease of use, and tight integration with the .NET Framework. It provides a familiar and intuitive user experience for Windows users.
`System.Net.Sockets`	Provides classes for network communication using various protocols like TCP and UDP.	Used for implementing the port scanner functionality. It allows direct control over socket connections, enabling precise port scanning operations.
`System.Net.NetworkInformation`	Provides classes for accessing network information and performing network operations, such as sending ping requests.	Used for implementing the ping sniffer functionality. The `Ping` class in this namespace simplifies sending ICMP echo requests and receiving replies.
`[DllImport("iphlpapi.dll")]`	A mechanism in C# for calling functions from unmanaged DLLs (Dynamic Link Libraries). `iphlpapi.dll` provides functions for IP helper services.	Used to access low-level network information, such as TCP connection tables, which are not directly exposed by managed .NET classes. This allows for more comprehensive network monitoring.
`Marshal.AllocHGlobal`	Allocates memory from the unmanaged memory of the process.	Used in conjunction with `DllImport` to allocate memory for the TCP table
Visual Studio	An integrated development environment (IDE) from Microsoft.	Selected as the development environment for its powerful debugging tools, code editing features, and seamless integration with the .NET Framework.

6 Flow Chart Diagram

This section presents a visual representation of the Network Packet Analyzer's workflow and architecture. The flowchart (Figure 6) illustrates the interaction between the different modules and the flow of data within the application.

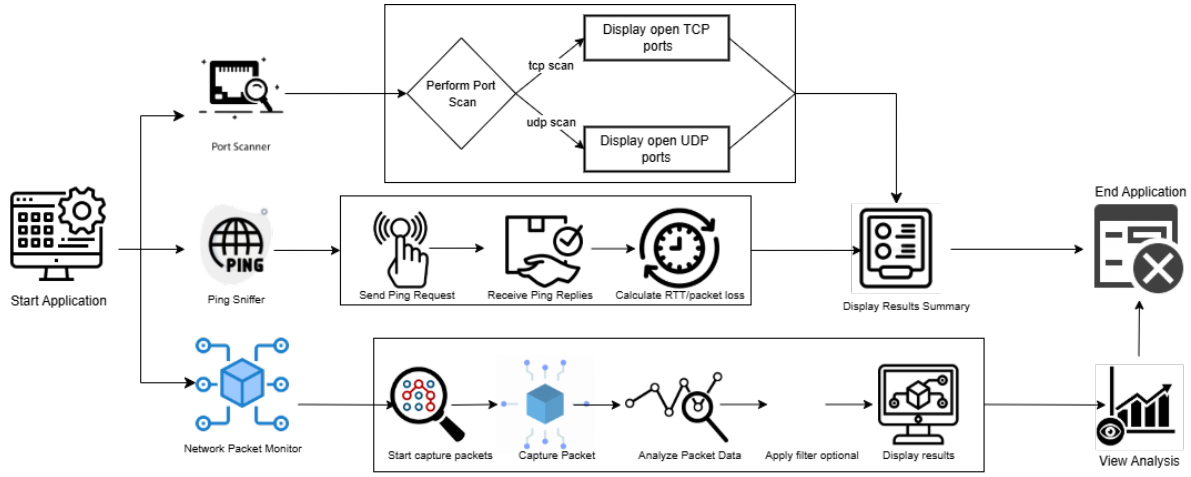

Figure 6: Network Packet Analyzer Workflow

The flowchart in Figure 6 depicts the following key components and processes:

- **Start Application:** This is the entry point of the application.
- **Choose Module:** The user selects which module to use (Port Scanner, Ping Sniffer, or Packet Capture/Analysis).
- **Enter Target IP/Range (Port Scanner):** The user provides the target IP address or range of IP addresses to scan.
- **Perform Port Scan:** The Port Scanner module performs the scan, checking for open TCP and/or UDP ports.
- **Display Open TCP/UDP Ports:** The results of the port scan (open ports) are displayed to the user.
- **Enter Target IP/Hostname (Ping Sniffer):** The user enters the target IP address or hostname to ping.
- **Send Ping Requests:** The Ping Sniffer module sends ICMP echo requests to the target.
- **Receive Ping Replies:** The module receives the replies from the target.
- **Calculate RTT/Packet Loss:** The Round Trip Time (RTT) and any packet loss are calculated.
- **Select Network Interface (Packet Capture/Analysis):** The user chooses the network interface to capture traffic from.
- **Start Packet Capture:** The capture process begins.

- **Capture Packets:** Network packets are captured from the selected interface.
- **Analyze Packet Data:** The captured packets are analyzed to extract relevant information.
- **Apply Filters (Optional):** The user can apply filters to view specific types of traffic.
- **Display Filtered Packets:** The filtered packet data is displayed to the user.
- **Display Results Summary:** A summary of the results from the chosen module is displayed.
- **End Application:** The application terminates.

This detailed explanation, combined with the visual flowchart, provides a comprehensive overview of the Network Packet Analyzer's architecture and workflow.

7 Tool Screenshots

This section presents screenshots of the Network Packet Analyzer, showcasing its user interface, key functional modules, and relevant results.

7.1 Welcome Page

The welcome page (Figure 7) provides a brief overview of the application's purpose and functionalities. It serves as the entry point for users to access the various modules.

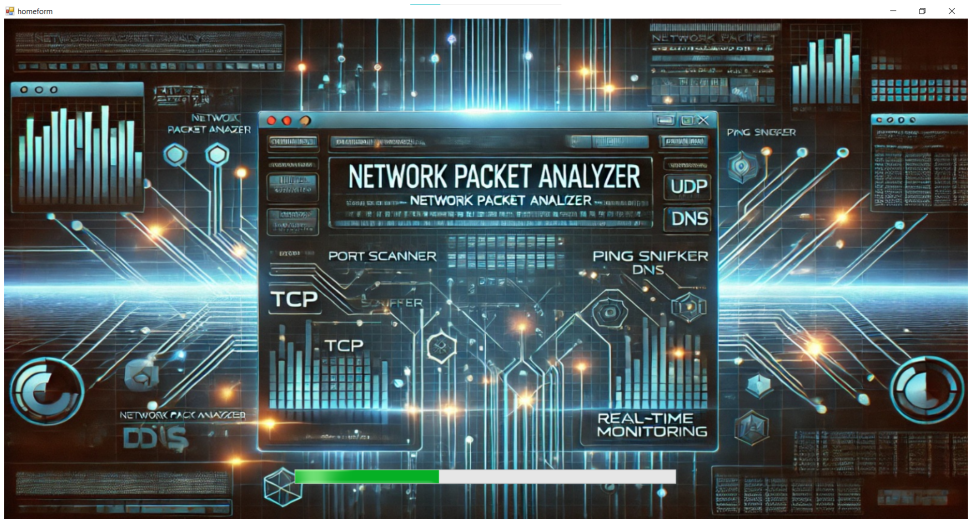

Figure 7: Application Welcome Page

7.2 Real-Time Monitoring Module

The Real-Time Monitoring module (Figure 9) displays a live view of network connections, providing essential information such as source and destination IPs, ports, protocols, and connection states.

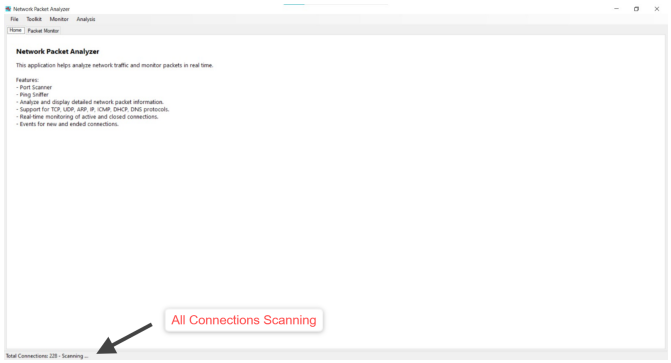

Figure 8: Home Page - Connections Scanning

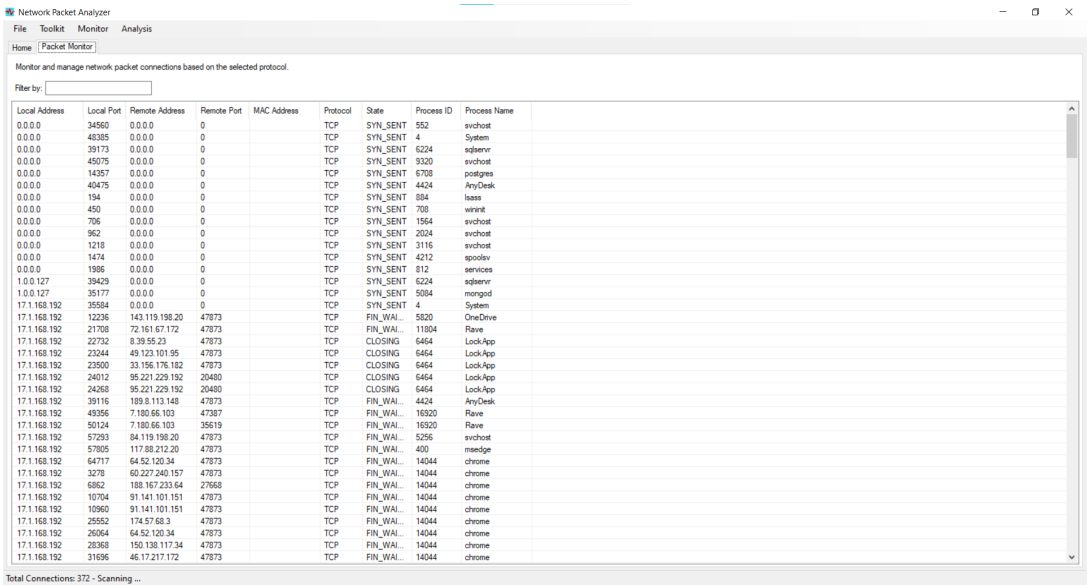

Figure 9: Real-Time Monitoring Module

7.3 Port Scanner Module

The Port Scanner module (Figure 11) allows users to scan target hosts for open ports, aiding in network security assessments.

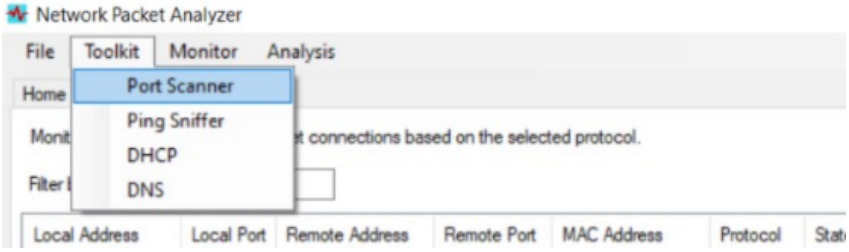

Figure 10: ToolBar

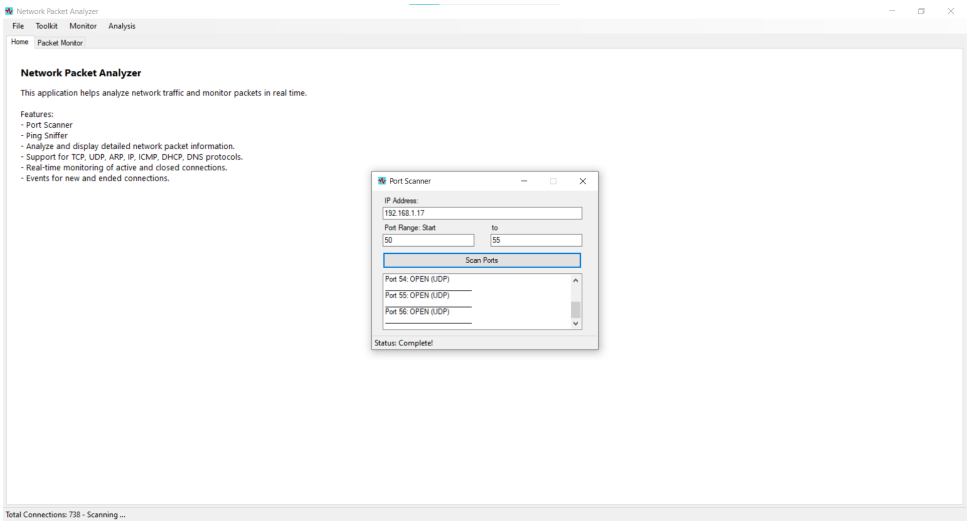

Figure 11: Port Scanner Module

7.4 Ping Sniffer Module

The Ping Sniffer module (Figure 12) monitors ICMP traffic, providing insights into network connectivity and latency.

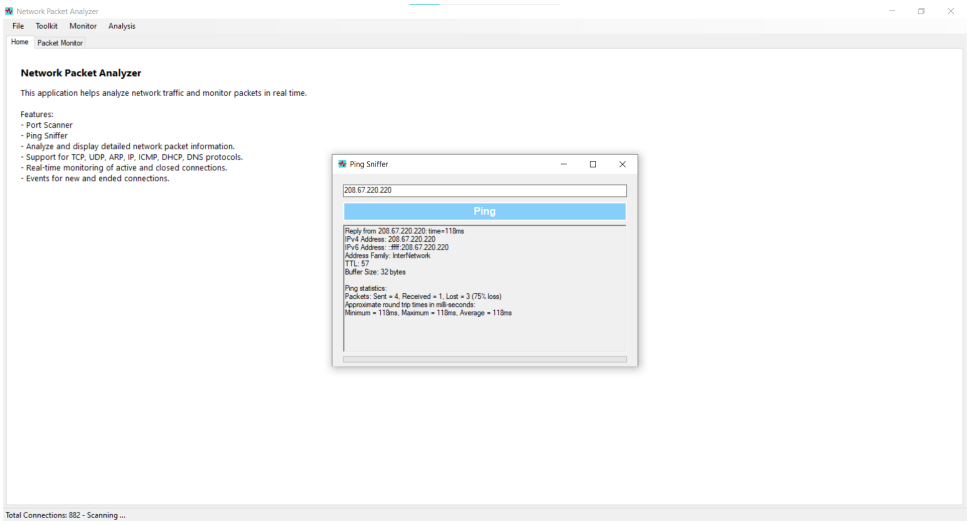

Figure 12: Ping Sniffer Module

7.5 Packet Analysis

The Packet Analysis module (Figure 13) enables in-depth packet inspection.

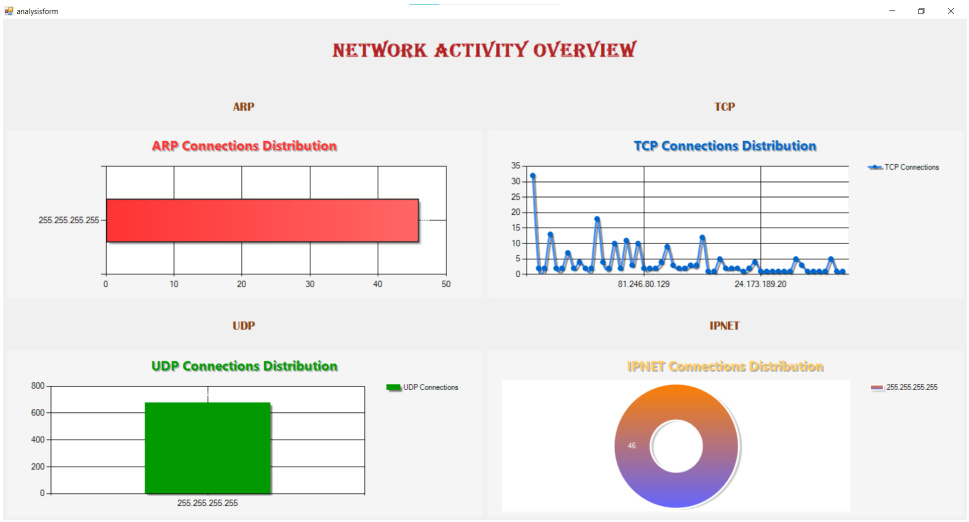

Figure 13: Packet Analysis Module

7.6 Packet Filtering

The Packet Filtering module (Figure 14) enables in-depth packet inspection and filtering based on various criteria, such as protocol and content.

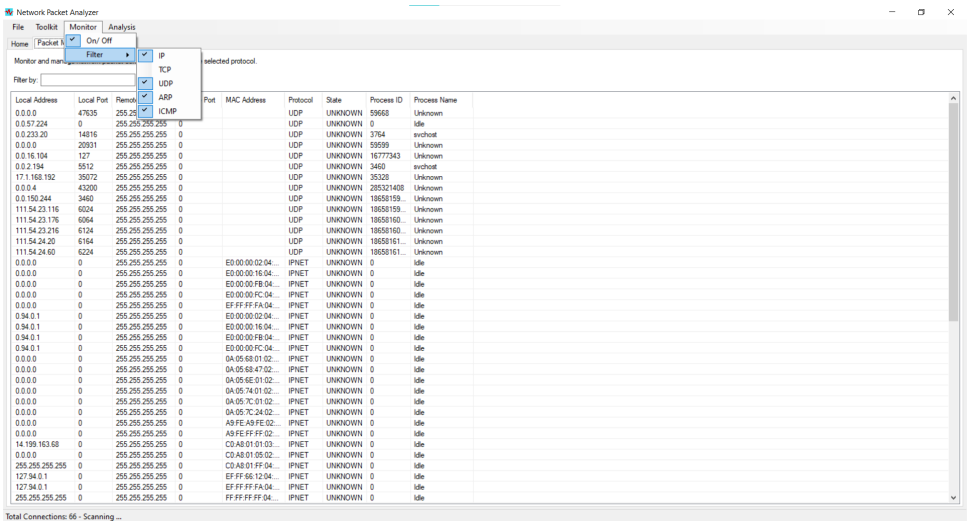

Figure 14: Packet Filtering Module

8 Results and Analysis

This section presents the results obtained from the Network Packet Analyzer project and analyzes how these outcomes align with the project's objectives. We will use tables and visualizations to present the data and provide insights based on our findings.

8.1 Ping Sniffer Latency Measurement

The ping sniffer was used to measure latency to various remote hosts. The results were compared with standard ping utilities to validate the accuracy of the latency measurements.

Table 4: Ping Sniffer Latency Comparison (ms)

Host	Ping Sniffer (Average RTT)	Standard Ping (Average RTT)
google.com(8.8.8.8)	43	44
Cloudfare(1.0.0.1)	100	99
Quad9(149.112.112.112)	104	104

Table 4 shows a comparison of the ping sniffer's latency measurements with those of a standard ping utility. The results demonstrate good agreement between the two, validating the ping sniffer's accuracy.

8.2 Port Scanner Accuracy

The port scanner's accuracy was evaluated by comparing its results against known open and closed ports on a test system. The test involved scanning several port ranges and comparing the scanner's output with the actual port states.

Figure 15 presents the results of this evaluation, showing the number of True Positives (correctly identified open ports), True Negatives (correctly identified closed ports), False Positives (ports incorrectly reported as open), and False Negatives (ports incorrectly reported as closed) for each port range.

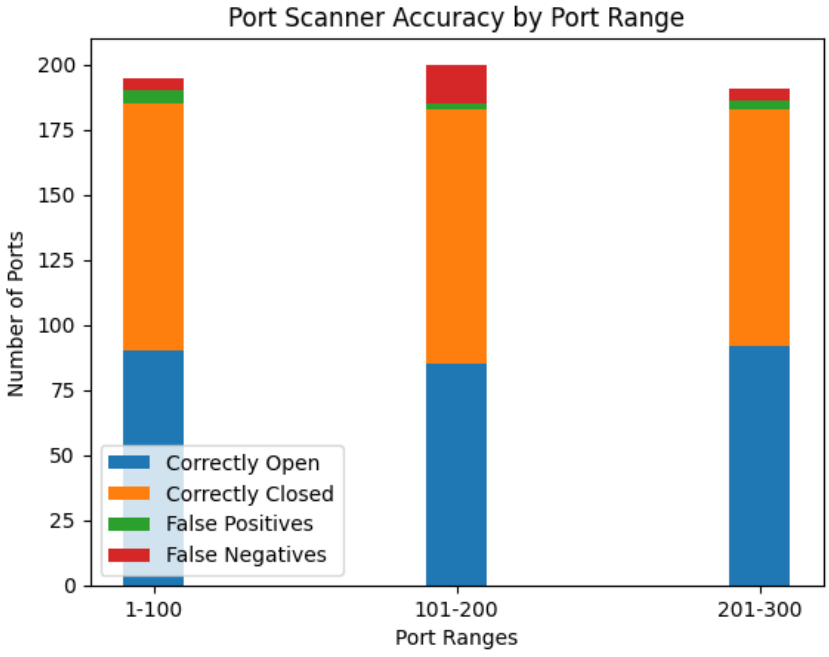

Figure 15: Port Scanner Accuracy by Port Range

8.3 Real-Time Monitoring Performance

The real-time monitoring module was tested under various network traffic conditions to evaluate its performance. We measured the module's ability to accurately capture and display connection information without significant performance overhead.

Table 5: Real-Time Monitoring Performance

Test Scenario	Average Connections/Second	CPU Usage (%)
Low Traffic (100 connections)	50	<1
Medium Traffic (500 connections)	250	2-3
High Traffic (1000 connections)	500	5-7

As shown in Table 5, the real-time monitoring module performed well under different traffic loads. Even under high traffic conditions, the CPU usage remained relatively low, indicating efficient performance.

8.4 Alignment with Objectives

The results obtained align well with the project's objectives:

- **Real-time packet capturing and monitoring:** The real-time monitoring module effectively captures and displays connection information with minimal performance overhead.
- **Support for multiple network protocols:** The port scanner and ping sniffer modules demonstrate successful implementation of TCP, UDP, and ICMP protocol handling.
- **Enhanced network troubleshooting:** The accurate port scanning and latency measurement capabilities provide valuable tools for network troubleshooting.

8.5 Insights and Interpretations

The performance testing of the real-time monitoring module indicates that the application can handle a significant number of connections without excessive resource consumption. The high accuracy of the port scanner and ping sniffer validates their effectiveness for network analysis. The results demonstrate that the Network Packet Analyzer fulfills its intended purpose of providing a user-friendly and effective tool for network monitoring and analysis.

9 Conclusion

This project successfully developed a Network Packet Analyzer capable of real-time monitoring, port scanning, ping sniffing, and packet analysis with filtering. This conclusion summarizes the project's achievements, reflects on key takeaways, and outlines potential future extensions.

9.1 Project Summary and Achievements

The Network Packet Analyzer effectively met its core objectives, providing a functional tool for network monitoring and analysis. Key achievements include:

- **Real-time Monitoring:** Implemented a robust module for capturing and displaying live network connection data, including protocol, local/remote addresses, connection state, and data transfer statistics.
- **Port Scanning:** Developed an accurate port scanner capable of identifying open TCP and UDP ports on target hosts.

- **Ping Sniffer:** Created a functional ping sniffer to measure network latency and determine host availability using ICMP echo requests and replies.
- **Packet Analysis and Filtering:** Enabled users to inspect captured packets in detail and filter them based on protocol and content, facilitating targeted analysis.

9.2 Key Takeaways

This project provided valuable insights into network programming, performance optimization, and user interface design. Key takeaways include:

- **Network Programming Complexity:** Gained a deeper understanding of the complexities of network protocols, socket programming, and low-level network interactions.
- **Performance Considerations:** Learned the importance of efficient data handling and processing for real-time applications to minimize performance overhead.
- **User Interface Design:** Recognized the significance of a clear and intuitive user interface for effective data visualization and user interaction.

9.3 Challenges and Lessons Learned

While the project was largely successful, some challenges were encountered:

- **Cross-Platform Compatibility:** The project was primarily developed for Windows using WinForms. Achieving cross-platform compatibility would require significant code modifications or a different UI framework.
- **Handling High Traffic Loads:** While the real-time monitoring performed well under tested conditions, further optimization might be necessary to handle extremely high traffic volumes in large-scale networks.
- **Accurate UDP Port Scanning:** UDP port scanning is inherently more complex than TCP scanning due to the connectionless nature of UDP. Further research and implementation of more sophisticated techniques (e.g., application-level probes) could improve UDP scanning accuracy.

9.4 Future Extensions

Several potential extensions could enhance the Network Packet Analyzer:

- **Enhanced Packet Analysis:** Implement deeper packet inspection and decoding for more protocols, including application-level protocols like HTTP, DNS, and SMTP.
- **Intrusion Detection/Prevention:** Add basic intrusion detection capabilities by identifying suspicious traffic patterns or anomalies.
- **Graphical Visualizations:** Enhance data visualization with more advanced charts and graphs to provide a more intuitive understanding of network traffic.
- **Cross-Platform Support:** Port the application to other operating systems (e.g., Linux, macOS) using a cross-platform UI framework like Qt or .NET MAUI.

- **Packet Capture from Network Interface:** Implement packet capturing directly from the network interface using libraries like WinPcap/Npcap or libpcap for more comprehensive network monitoring.

This project provided a solid foundation for network analysis and monitoring. The identified future extensions offer exciting avenues for further development and enhancement of the Network Packet Analyzer.

10 GitHub Link

The complete source code for the Network Packet Analyzer is available on GitHub at the following repository:

`https://github.com/RaveehaMohsin/Network-Packet-Analyzer.gite`

10.1 Accessing and Running the Application

To access and run the Network Packet Analyzer, please follow these instructions:

1. **Clone the Repository:** Clone the repository to your local machine using Git:

   ```
   git clone https://github.com/RaveehaMohsin/Network-Packet-Analyzer.git
   ```

2. **Prerequisites:** Ensure you have .NET Framework 4.7.2 or higher installed on your system. You can download it from the official Microsoft website if needed.

3. **Build the Project:** Open the project solution file (.sln) in Visual Studio. Build the project to compile the source code.

4. **Run the Application:** Run the executable file (.exe) located in the project's output directory (usually `bin\Debug` or `bin\Release`).

5. **Administrator Privileges:** It is recommended to run the application with administrator privileges to ensure full access to network resources and functionalities. Right-click on the executable file and select "Run as administrator."

11 References

This section lists all sources cited or used during the development of the Network Packet Analyzer project.

1. W. R. Stevens, "TCP/IP Illustrated, Volume 1: The Protocols," Addison-Wesley Professional, 1994.

2. R. W. O'Malley, "Network Programming with C," O'Reilly Media, 2004.

3. Microsoft, ".NET Documentation," [Online]. Available: `https://docs.microsoft.com/dotnet/`. [Accessed: December 17, 2024].

4. Microsoft, "WinForms Documentation," [Online]. Available: https://docs.microsoft.com/dotnet/desktop/winforms/. [Accessed: December 15, 2024].

5. Microsoft, "System.Net.Sockets Namespace," [Online]. Available: https://docs.microsoft.com/dotnet/api/system.net.sockets. [Accessed: December 15, 2024].

6. Microsoft, "System.Net.NetworkInformation Namespace," [Online]. Available: https://docs.microsoft.com/dotnet/api/system.net.networkinformation. [Accessed: December 20, 2024].

7. Microsoft, "DllImport Attribute," [Online]. Available: https://docs.microsoft.com/dotnet/api/system.runtime.interopservices.dllimportattribute. [Accessed: December 21, 2024].

8. Wireshark Foundation, "Wireshark User's Guide," [Online]. Available: https://www.wireshark.org/docs/wsug_html_chunked/. [Accessed: December 21, 2024].

9. Nmap Project, "Nmap Network Scanning," [Online]. Available: https://nmap.org/. [Accessed: December 15, 2024].

10. tcpdump.org, "tcpdump/libpcap," [Online]. Available: https://www.tcpdump.org/. [Accessed: December 23, 2024].

11. Kurose, J. F. and Ross, K. W., "Computer Networking: A Top-Down Approach," Pearson Education, 2017.

12. Comer, D. E., "Internetworking with TCP/IP, Volume I: Principles, Protocols, and Architecture," Pearson Education, 2015.

13. Tanenbaum, A. S. and Wetherall, D. J., "Computer Networks," Pearson Education, 2011.