CHAPTER 1 :INTRODUCTION

CHAPTER 1.1 PROBLEM DEFINITION

The internet is a massive web of computers that communicate by sending small packets of data back and forth. Each packet contains useful information such as where it came from, where it's going, and what kind of data it holds. Tools that help us monitor and analyze these packets are called packet sniffers. This project is about creating a simple packet sniffer using Java.
Packet sniffers are widely used by network administrators, security professionals, and developers to troubleshoot issues, optimize performance, and study network behavior. The tool we have built helps us understand how these packets work in a real-world scenario by letting us observe live traffic.

Chapter 1.2 Importance of Packet Sniffing

Packet sniffing is an essential task in the field of network security and diagnostics. It helps:

- Detect security threats like unauthorized access
- Identify performance bottlenecks in a network
- Debug networking issues during development
- Learn about how data travels across systems

CHAPTER 1.3 Scope of the Project

This project is designed to help students and beginners get familiar with packet capturing using Java. By using libraries like Pcap4J and tools like JavaFX, we provide a hands-on tool that is functional and easy to understand

CHAPTER 1.4 Project Highlight

- Real-time packet capture using Pcap4J
- Header parsing (IP, Protocol)
- Protocol filtering using BPF (e.g., TCP)
- JavaFX-based GUI

- Log output to a text file

CHAPTER 2: Objective

2.1 Primary Goals

The main aim of this project is to create a basic yet powerful packet sniffer application that allows the user to:

- Monitor live network traffic
- Extract key packet-level details
- Save captured packet data for later analysis
- Display packet information in a user-friendly GUI

2.2 Secondary Goals

- Filter captured data by protocol (e.g., TCP only)
- Use a modular structure for separation of concerns
- Provide a foundation for future enhancements

3. LITERATURE SURVEY

3.1 Existing Tools

There are several tools in the market today like:

- **Wireshark:** A full-featured open-source packet sniffer
- **Tcpdump:** A command-line packet analyzer
- **Microsoft Message Analyzer:** An advanced tool for Windows

3.2 Why a Java-based Sniffer?

- Platform-independent development
- Simple UI building with JavaFX
- Availability of Pcap4J to interact with native packet capturing libraries

3.3 Research References

- [1] https://pcap4j.github.io/
- [2] https://npcap.com/
- [3] JavaFX Official Documentation

4. SOFTWARE REQUIREMENTS

4.1 System Requirements

- Operating System: Windows 10/11
- RAM: Minimum 4 GB
- Disk Space: 500 MB free space
- Processor: Intel Core i3 or higher

4.2 Tools Used

- Java JDK 17
- Maven
- Npcap
- JavaFX SDK
- Visual Studio Code / IntelliJ IDEA

4.3 External Libraries

- Pcap4J Core
- Pcap4J Packet Factory Static
- SLF4J Logging API

---

5. MODULE DESCRIPTION

5.1 Module 1: Interface Selection

- Lists all available interfaces
- Allows program to choose based on index (default: Wi-Fi)

5.2 Module 2: Packet Capture

- Uses a handle to listen to live network packets
- Loop captures a specified number of packets (e.g., 50)

5.3 Module 3: Packet Parsing

- Extracts IPv4 headers
- Retrieves source IP, destination IP, protocol type

5.4 Module 4: File Logging

- Uses BufferedWriter to write data to packets_log.txt
- Includes all parsed data for reference

5.5 Module 5: Graphical UI

- JavaFX GUI displays real-time data
- Includes Start Sniffing button
- TableView to show packet logs

---

6. IMPLEMENTATION DETAILS

6.1 Code Structure

- PacketSniffer.java: Captures and processes packets
- PacketInfo.java: Holds information like IPs and protocol
- PacketSnifferApp.java: Creates the GUI and handles user interaction

6.2 Interface Selection

Code uses Pcap4J to list interfaces:

```
List<PcapNetworkInterface> interfaces = Pcaps.findAllDevs();

PcapNetworkInterface nif = interfaces.get(3);
```

6.3 Capturing Packets

```
handle.loop(50, packet -> {
  // parsing logic
});
```

6.4 Parsing Headers

```
IpV4Packet ip = packet.get(IpV4Packet.class);
if (ip != null) {
   ip.getHeader().getSrcAddr();
   ip.getHeader().getDstAddr();
}
```

6.5 Logging to File

```
BufferedWriter writer = new BufferedWriter(new FileWriter("packets_log.txt"));
writer.write(...);
```

6.6 JavaFX UI

- Button to start capture
- Table to display timestamp, source IP, dest IP, protocol

---

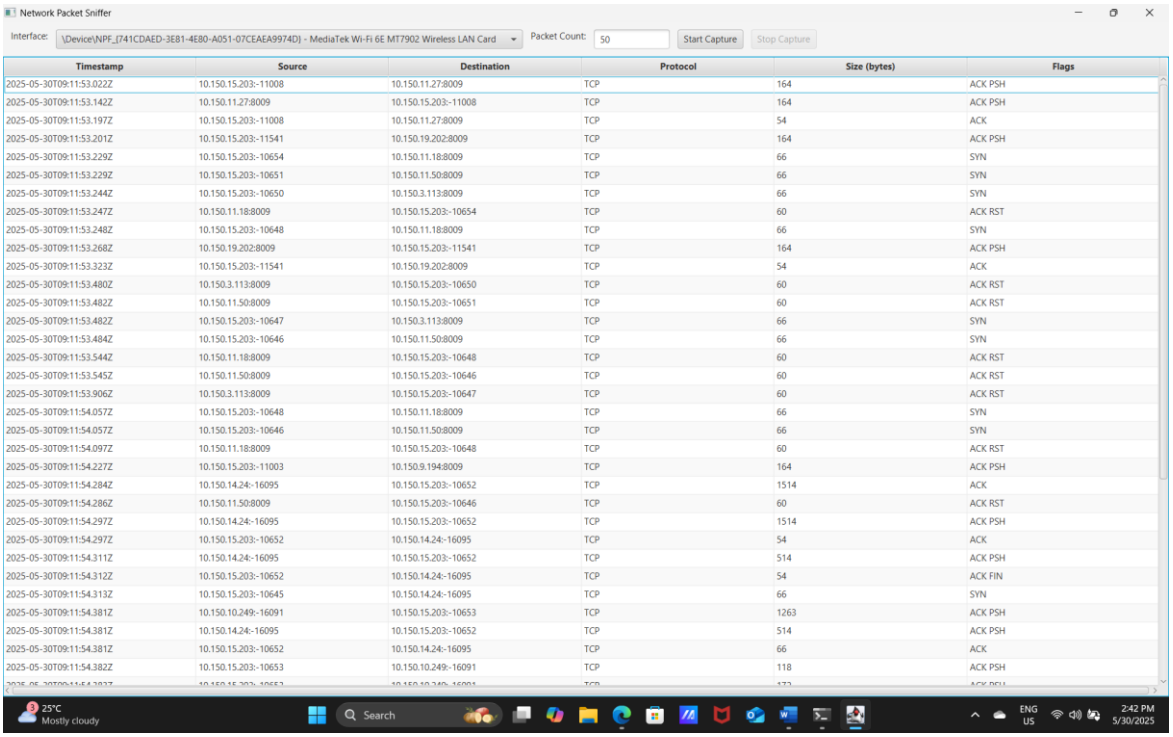
7. OUTPUT AND RESULTS

7.1 Console Output

```
Interface 3 selected: MediaTek Wi-Fi
Timestamp: 2025-05-28 15:22:17
Source IP: 192.168.0.101
Destination IP: 142.250.194.78
```

Protocol: TCP

7.2 GUI Screenshot



7.3 File Output

packets_log.txt

Timestamp: 2025-05-28 15:22:17

Source IP: 192.168.0.101

Destination IP: 142.250.194.78

Protocol: TCP

---

8. ADVANTAGES

- Lightweight and fast
- Simple to use with real output

- Educational for networking students
- Works on most systems with Java and Npcap

---

9. LIMITATIONS

- Admin access required
- No advanced protocol parsing
- GUI cannot visualize packet content deeply
- Fixed capture size (not continuous)

---

10. FUTURE ENHANCEMENTS

- Dynamic interface selection via dropdown
- Chart/graph support for live stats
- HTTP header parsing
- Export in .pcap format
- Filtering options in UI

---

11. CONCLUSION

The project was successfully implemented and achieved its goal of building a basic yet functional packet sniffer using Java. It helped demonstrate the use of external libraries, file handling, GUI development, and real-time data capture. The application provides a base for anyone interested in learning about packet structures and how network sniffing tools work.

---

12. REFERENCES

- https://pcap4j.github.io/
- https://npcap.com/

- https://openjfx.io/
- Java Documentation
