



## King Abdulaziz University

Faculty of Computing & Information Technology

CPCS 371

Instructor: Dr. Naif Rajkhan

Section: CS1

## GROUP PROJECT

### TCP/IP Packet Demultiplexer

1	Ali Abdullah Al-Hebshi	[Redacted]
2	Abdullah Alhalees	[Redacted]
3	Abdulrahman Alfrihidi	[Redacted]
4	Umair Thasleem	[Redacted]

# TABLE OF CONTENTS

03	
INTRODUCTION.....	
OVERVIEW OF THE TCP/IP PROTOCOL SUITE.....	
IMPORTANCE OF PACKET DEMULTIPLEXING IN NETWORK ANALYSIS.....	
04	
GOALS AND OBJECTIVES OF THE PROJECT.....	
REAL-WORLD APPLICATIONS OF PACKET DEMULTIPLEXING.....	
05	
SYSTEM DESIGN.....	
ARCHITECTURE.....	
COMPONENTS.....	
INTERACTION.....	
DATA FLOW.....	
06	
TESTING AND VALIDATION.....	
UNIT TESTING.....	
07	
INTEGRATION TESTING.....	
VALIDATION.....	
08	
DOCUMENTATION.....	
CODE DOCUMENTATION .....	
USER MANUAL.....	
TECHNICAL REPORT.....	
09	
SAMPLE INPUT/OUTPUT.....	

## INTRODUCTION

- **OVERVIEW OF THE TCP/IP PROTOCOL SUITE**

The TCP/IP protocol suite, also known as the Internet protocol suite, is the foundation of modern computer networking and communication across the Internet. It defines a set of standardized protocols that enable devices to connect, communicate, and share data reliably over various types of networks. TCP/IP stands for Transmission Control Protocol / Internet Protocol, which are the two core protocols of the suite.

The suite is structured in four layers, each with its own responsibilities:

1. Application Layer – Provides services and interfaces for user applications such as HTTP (web), FTP (file transfer), and SMTP (email).
2. Transport Layer – Ensures reliable or connectionless data transmission between devices. The main protocols are TCP (reliable, connection-oriented) and UDP (faster, connectionless).
3. Internet Layer – Handles logical addressing and routing through the IP protocol, ensuring data packets reach their destination across networks.
4. Network Access Layer – Manages physical connections and data framing across network interfaces like Ethernet or Wi-Fi.

Together, these layers define how data is packaged, addressed, transmitted, routed, and received. TCP/IP's layered design allows interoperability among different hardware and operating systems, making it the global standard for data communication.

- **IMPORTANCE OF PACKET DEMULTIPLEXING IN NETWORK ANALYSIS**

Packet demultiplexing is essential in network analysis because modern networks transmit massive amounts of interleaved traffic from multiple applications, users, and connections. When packets arrive at a device, they do not appear in a neatly organized sequence—packets from different TCP or UDP sessions are mixed together.

Demultiplexing separates packets by examining source and destination IP addresses, port numbers, and protocol identifiers, allowing each flow to be isolated and analyzed independently. This separation is critical for diagnosing performance issues, understanding communication patterns, detecting security threats, and troubleshooting application-level behavior. Without effective packet demultiplexing, captured network data would appear chaotic and extremely difficult to interpret, making accurate monitoring and detailed analysis nearly impossible.

## **• GOALS AND OBJECTIVES OF THE PROJECT**

The main goal of this project is to develop a functional TCP/IP Packet Demultiplexer capable of capturing, separating, and organizing TCP packets based on their source and destination addresses, ports, and protocol information. The project aims to provide a practical tool that helps students and analysts understand low-level TCP/IP communication and how individual network flows can be isolated for analysis.

The key objectives of the project are:

- Capture TCP packets from a selected network interface using Pcap4J.
- Extract connection identifiers such as source IP, destination IP, source port, and destination port.
- Demultiplex packets by grouping them into their corresponding TCP flows.
- Store raw packets in separate output files for each connection.
- Demonstrate a clear system structure showing the interaction between packet capture, demultiplexing, and storage.
- Enable testing and verification by allowing captured packets to be reviewed and analyzed.
- Provide documentation describing the system design, functionality, and usage.

Together, these objectives ensure that the project demonstrates how packet demultiplexing operates in real network environments while offering a useful tool for learning, analysis, and troubleshooting.

## **• REAL-WORLD APPLICATIONS OF PACKET DEMULTIPLEXING**

Packet demultiplexing is widely used in many real-world networking systems and tools. Network monitoring platforms rely on it to separate traffic into individual flows for performance analysis and troubleshooting. Security systems, such as intrusion detection and firewall inspection engines, use demultiplexing to analyze suspicious connections and detect malicious activity.

Packet analysis tools like Wireshark depend on demultiplexing to organize captured packets into readable TCP or UDP sessions. Additionally, routers, servers, and operating systems use this process internally to route incoming packets to the correct application or protocol handler. These applications highlight the importance of demultiplexing in ensuring efficient communication, accurate analysis, and strong network security.

## SYSTEM DESIGN

- **ARCHITECTURE OVERVIEW**

The packet demultiplexer is designed as a modular system that captures TCP packets from a network interface, separates them based on connection information, and stores them for later analysis. The architecture consists of three main components: Packet Capture, Demultiplexer, and Data Storage.

This modular approach improves scalability and makes each component easier to test, maintain, and extend.

- **COMPONENT**

**1. Packet Capture:** Responsible for listening to a selected network interface and collecting incoming TCP packets. It filters network traffic and forwards each packet to the next stage for processing.

**2. Demultiplexer:** Analyzes each captured packet and determines which connection flow it belongs to by examining source/destination IP addresses and port numbers. It ensures packets are grouped correctly for later analysis.

**3. Data Storage:** Stores packets belonging to each identified flow in separate files. Organizing packets this way allows analysts to examine individual TCP sessions independently.

- **INTERACTION**

The Packet Capture module collects live TCP packets from the network interface and forwards them to the Demultiplexer, which identifies the corresponding flow by analyzing IP and port information. Once the flow is determined, packets are passed to the Data Storage module, where they are written into the appropriate per-flow file for later analysis.

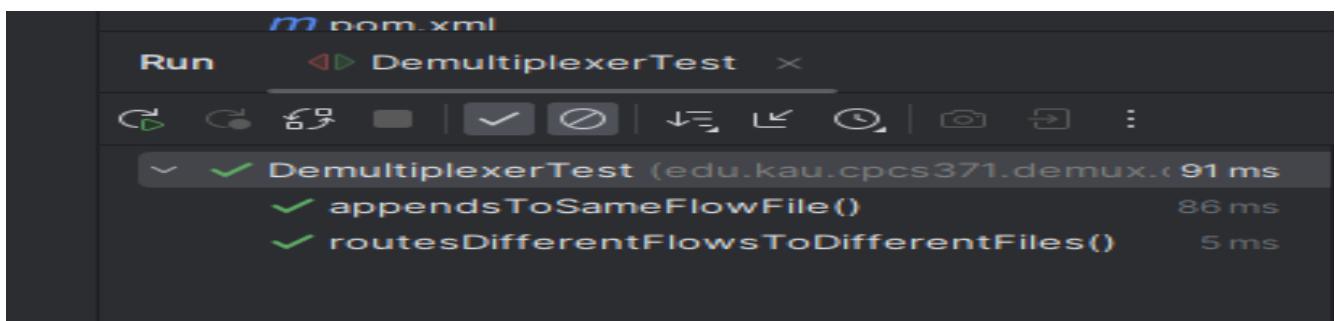
- **DATA FLOW**

Network Interface → Packet Capture Module → Demultiplexer (Flow Identification & Routing)  
→ Data Storage Module → Per-Flow Output Files

## TESTING AND VALIDATION

### • UNIT TESTING

Unit tests were created to verify that each core component behaves correctly. The IpPortKey tests confirm that flow identifiers are generated consistently and that equality and hashing work as expected. The FlowFileWriter tests ensure that output files are created with the correct names and that packet data is written successfully. Finally, the Demultiplexer tests check that packets belonging to the same flow are appended to the same file, while packets from different flows are correctly separated into different files. All tests passed without errors, confirming the correctness of these modules.



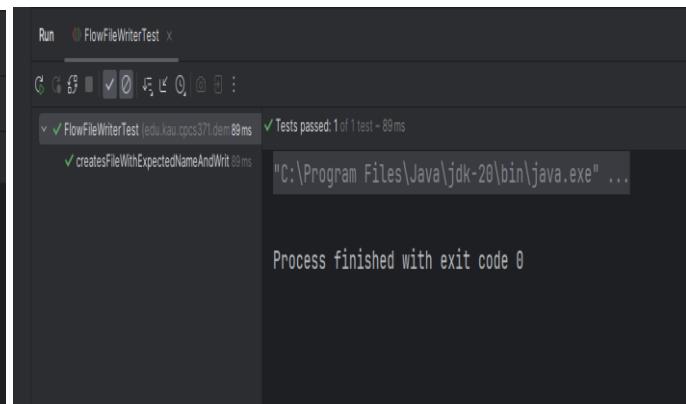
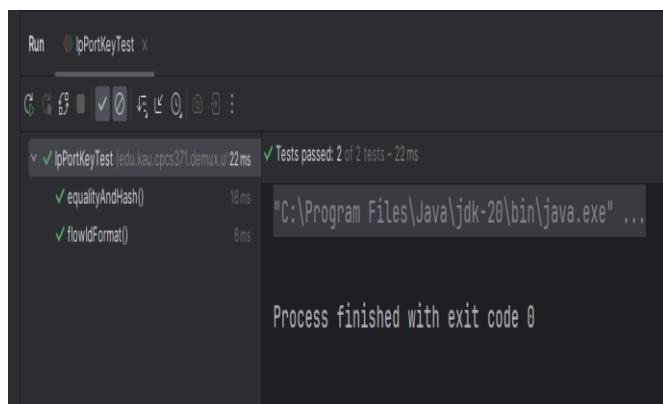
✓ Tests passed: 2 of 2 tests - 91ms

```
"C:\Program Files\Java\jdk-20\bin\java.exe" ...
```

```
Opened flow file: C:\Users\ABOALI~1\AppData\Local\Temp\junit18400887903562143015\[20251125_193931_868]10.0.0.2.2222-93.184.216.34.443
```

```
Opened flow file: C:\Users\ABOALI~1\AppData\Local\Temp\junit16752828112077606482\[20251125_193931_894]10.0.0.1.1111-93.184.216.34.80
```

```
Opened flow file: C:\Users\ABOALI~1\AppData\Local\Temp\junit16752828112077606482\[20251125_193931_895]10.0.0.1.1112-93.184.216.34.80
```



- **INTEGRATION TESTING**

Integration tests verified the entire pipeline by capturing real TCP traffic and confirming that packets were routed into the correct per-flow files. The console logs and generated output files show that capture, demultiplexing, and storage all work together without errors.

```
== TCP Packet Demultiplexer ==
Found 8 network interface(s):
[0] \Device\NPF_{88AC18C2-134E-430D-8F53-B54898C0F775} (name=\Device\NPF_{88AC18C2-134E-430D-8F53-B54898C0F775}, description=WAN Miniport (Network Monitor))
[1] \Device\NPF_{617E76DF-BAC3-413F-8512-684F35A539A9} (name=\Device\NPF_{617E76DF-BAC3-413F-8512-684F35A539A9}, description=WAN Miniport (IPv6))
[2] \Device\NPF_{1A5BF15F-834D-4A85-9A15-8326E7909BDA} (name=\Device\NPF_{1A5BF15F-834D-4A85-9A15-8326E7909BDA}, description=WAN Miniport (IP))
[3] \Device\NPF_{17AAD8D-9469-46F6-B3C5-48BCA22E0E64} (name=\Device\NPF_{17AAD8D-9469-46F6-B3C5-48BCA22E0E64}, description=Bluetooth Device (Personal Area Network))
[4] \Device\NPF_{37DF239F-9846-4749-8FFD-AF5F9F751608} (name=\Device\NPF_{37DF239F-9846-4749-8FFD-AF5F9F751608}, description=Intel(R) Wi-Fi 7 BE201 320MHz)
[5] \Device\NPF_{B5E5F5A2-2DD2-4DEA-B718-59974A2A9ADB} (name=\Device\NPF_{B5E5F5A2-2DD2-4DEA-B718-59974A2A9ADB}, description=Intel(R) Wi-Fi 7 BE201 320MHz #5)
[6] \Device\NPF_{5BD463D0-7793-417C-898D-62B8A70A6C1C} (name=\Device\NPF_{5BD463D0-7793-417C-898D-62B8A70A6C1C}, description=Intel(R) Wi-Fi 7 BE201 320MHz #4)
[7] \Device\NPF_Loopback (name=\Device\NPF_Loopback, description=Adapter for loopback traffic capture)
Using interface index: 7
Max packets: 200
Output dir: C:\Users\d7oom\OneDrive\Documents\Network Project\Network Project\captures
Opening: Adapter for loopback traffic capture
Capturing TCP -> writing to: C:\Users\d7oom\OneDrive\Documents\Network Project\Network Project\captures (maxPackets=200)
Opened flow file: C:\Users\d7oom\OneDrive\Documents\Network Project\Network Project\captures\[20251125_194327_184]127.0.0.1.55686-127.0.0.1.5037
Opened flow file: C:\Users\d7oom\OneDrive\Documents\Network Project\Network Project\captures\[20251125_194327_227]127.0.0.1.5037-127.0.0.1.55686
...captured 50 packets
```

- **VALIDATION**

We validated correctness by generating plain HTTP traffic to [neverssl.com](https://neverssl.com), then inspecting the per-flow files: the outbound file contained GET / HTTP/1.1 and Host: headers, and the inbound file contained HTTP/1.1 200 OK with standard response headers—confirming accurate capture and demultiplexing.

#### REQUEST:

```
HTTP/1.1 200 OK
Host: neverssl.com
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

#### RESPONSE:

```
HTTP/1.1 200 OK
Date: Sun, 16 Nov 2025 15:45:23 GMT
Server: Apache/2.4.62 ()
Upgrade: h2,h2c
Connection: Upgrade, Keep-Alive
Last-Modified: Wed, 29 Jun 2022 00:23:33 GMT
ETag: "f79-5e28b29d38e93-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1900
Keep-Alive: timeout=5, max=100
Content-Type: text/html; charset=UTF-8
```

## DOCUMENTATION

### • CODE DOCUMENTATION

The source code of the packet demultiplexer is organized into small, focused classes, each responsible for a specific part of the workflow. Brief documentation is included in the code to clarify the purpose of each component:

- PacketCapturer : captures live TCP packets from the selected network interface and forwards them for processing.
- Demultiplexer : Uses the flow identifier to route each packet to the correct output file.
- FlowFileWriter : creates and writes packet data to per-flow output files using the required naming format.
- IpPortKey : defines a stable identifier for each TCP connection based on IP addresses and ports.
- Main: detects available interfaces, reads program arguments, and starts the capture process.

The code uses descriptive method names and brief comments to explain intent and high-level behavior. This simple and modular structure makes the system easy to understand, test, and extend.

### • USER MANUAL

The system includes a user manual that explains how to install and run the packet demultiplexer. It outlines the required environment, including Java, Pcap4J, and Npcap (Windows packet-capture driver). The manual guides users through launching the program and configuring capture settings. **User able to choose:**

- Network Interface – The program lists all available interfaces for selection.
- Capture Mode – Command-line execution or a simple interactive text menu.
- Packet Limit – A fixed number of packets or continuous capture.
- Output Directory – The folder where the per-flow files will be stored.

The manual also explains how output files are organized by flow, how IP/port identifiers are used, and includes short troubleshooting tips for common issues such as missing drivers, permission errors, or inactive interfaces. Its purpose is to ensure that any user can set up the environment, run the tool, and understand the generated files easily.

- **TECHNICAL REPORT**

This report presents the design and functionality of a Java-based TCP packet demultiplexer developed as part of a network analysis project. The system captures live TCP traffic from a selected network interface, identifies individual flows using source and destination IP addresses and ports, and stores each flow in a separate output file for further inspection. The implementation uses a modular architecture built with a Maven project structure and integrates the Npcap library through the Pcap4J framework to enable packet capture at the OS level. The demultiplexing logic is implemented through the PacketCapturer, Demultiplexer, and FlowFileWriter components, each responsible for a specific stage of packet handling.

Comprehensive testing was performed to confirm that packets were correctly captured, classified, and written to the appropriate per-flow files, ensuring accurate separation of concurrent network connections. Along with the accompanying documentation and user manual, this report provides a complete explanation of the system's purpose, design decisions, implementation details, and practical usage.

## SAMPLE INPUT/OUTPUT

**INPUT:**

```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:== TCP Packet Demultiplexer (bootstrap) ===
Found 9 network interface(s):
[.] \Device\NPF_{1A068BC0-3171-4109-83B0-A14C738C04D2}
[.] \Device\NPF_{798ADD0E-D32B-45F3-9853-B72710F2A020}
[.] \Device\NPF_{FA6933A5-D769-4E05-89A8-6FF18DA8FBB9}
[.] \Device\NPF_{9982D164-5CBA-4E9C-9D2A-4BA7463691D1}
[.] \Device\NPF_{50DD57F8-B78D-4665-9E46-80144A0FE324}
[.] \Device\NPF_{69BE72E0-ABBC-47DC-97ED-EC0054045A10}
[.] \Device\NPF_{2A8EFF06-E06A-44B3-A1EE-B940975EA635}
[.] \Device\NPF_{B0C12811-6623-4025-BE9D-5514077AB954}
[.] \Device\NPF_Loopback (name=\Device\NPF_Loopback, d
Opening: Intel(R) Ethernet Connection (7) I219-V
Capturing TCP packets... (will stop after 20 packets)
192.168.8.100:51183 -> 15.197.167.90:443 len=316
15.197.167.90:443 -> 192.168.8.100:51183 len=262
192.168.8.100:51183 -> 15.197.167.90:443 len=134
15.197.167.90:443 -> 192.168.8.100:51183 len=60
192.168.8.100:51191 -> 86.51.81.80:443 len=66
86.51.81.80:443 -> 192.168.8.100:51191 len=66
192.168.8.100:51191 -> 86.51.81.80:443 len=54
```

**OUTPUT:**

```
Capturing TCP -> writing to: C:\testingAA\tcp-demux\.captures (maxPackets=200)
Opened flow file: C:\testingAA\tcp-demux\.captures\[20251125_195145_682]162.159.134.234.443-192.168.8.100.57527
Opened flow file: C:\testingAA\tcp-demux\.captures\[20251125_195145_706]192.168.8.100.57527-162.159.134.234.443
Opened flow file: C:\testingAA\tcp-demux\.captures\[20251125_195145_770]2a02_9b0_4015_c1d6_d073_f5fe_2c9f_15b7.57
Opened flow file: C:\testingAA\tcp-demux\.captures\[20251125_195145_830]2606_4700_7_0_0_0_a29f_80eb.2087-2a02_9b0_
Opened flow file: C:\testingAA\tcp-demux\.captures\[20251125_195148_100]2a02_9b0_4015_c1d6_d073_f5fe_2c9f_15b7.57
Opened flow file: C:\testingAA\tcp-demux\.captures\[20251125_195148_177]2606_4700_7_0_0_0_a29f_8aea.2083-2a02_9b0_
Opened flow file: C:\testingAA\tcp-demux\.captures\[20251125_195149_313]192.168.8.100.59446-23.40.158.218.80
Opened flow file: C:\testingAA\tcp-demux\.captures\[20251125_195149_405]23.40.158.218.80-192.168.8.100.59446
Opened flow file: C:\testingAA\tcp-demux\.captures\[20251125_195149_498]192.168.8.100.59447-142.250.201.227.80
Opened flow file: C:\testingAA\tcp-demux\.captures\[20251125_195149_545]142.250.201.227.80-192.168.8.100.59447
Opened flow file: C:\testingAA\tcp-demux\.captures\[20251125_195149_622]192.168.8.100.59448-23.204.89.56.80
Opened flow file: C:\testingAA\tcp-demux\.captures\[20251125_195149_653]23.204.89.56.80-192.168.8.100.59448
...captured 50 packets
```