LINUX NETWORK PACKET STATISTICS DISPLAY PROJECT

**Table of Contents**

[Overview 3](#_Toc167976550)

[Objectives 3](#_Toc167976552)

[Declaration and Includes](#_Toc167976554) 3

[Code Functionality .......5](#_Toc167976555)

[Debbuging 7](#_Toc167976556)

[Conclusion](#_Toc167976557) 7

# **Overview**

# This documentation covers the implementation of a network packet statistics display application. The application captures and analyzes network packets, storing the results in shared memory. It provides a real-time display of the packet statistics in either tabular or graphical format. The display format can be customized through command-line arguments. This project demonstrates the use of shared memory for inter-process communication (IPC) and multi-threaded programming in C.

# **Objectives**

The primary objective of this application is to:

1. Capture and analyze network packets.
2. Store the analyzed data in shared memory.
3. Display the packet statistics in real-time.
4. Allow the user to choose between tabular and graphical display formats.
5. Implement the above functionality using multi-threading for concurrent packet capture and UI display.

# **Declaration and Includes**

## **Global Variables:**

**Shared Memory Key and Size:**

const key\_t shm\_key = 9768;

const size\_t shm\_size = sizeof(PacketStatistics);

**Display Control variables:**

enum DisplayFormat {

TABULAR,

GRAPH

};

enum DisplayFormat display\_format = TABULAR;

bool show\_tcp = true;

bool show\_udp = true;

bool show\_icmp = true;

**Includes**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <pthread.h>

#include <string.h>

#include <stdbool.h>

#include <time.h>

# **Code Functionality**

### **Packet Capture and Analysis**

**Functionality:**

* The capture\_and\_analyze\_packets function simulates the capture and analysis of network packets.
* It writes the packet statistics to shared memory for real-time access by the UI display thread.

**Components:**

**Shared Memory Setup:**

int shm\_id = shmget(shm\_key, shm\_size, IPC\_CREAT | 0666);

PacketStatistics \*shared\_stats = (PacketStatistics \*)shmat(shm\_id, NULL, 0);

memset(shared\_stats, 0, shm\_size);

**Packet Simulation:**

while (1) {

shared\_stats->tcp\_packet\_count += 10;

// ... (similar for UDP and ICMP)

usleep(500000); // Simulate delay

}

### **UI Display**

**Functionality:**

* The display\_ui function continuously reads packet statistics from shared memory and displays them.
* It supports both tabular and graphical display formats.

**Components:**

**Shared Memory Access:**

**int shm\_id = shmget(shm\_key, shm\_size, 0666);**

**PacketStatistics \*shared\_stats = (PacketStatistics \*)shmat(shm\_id, NULL,SHM\_RDONLY);**

**Display Functions:**

* The display\_ui function continuously reads packet statistics from shared memory and displays them.
* It supports both tabular and graphical display formats.

### **Argument Parsing**

**Functionality:**

* The parse\_arguments function processes command-line arguments to set the display format and packet types to be shown.

**Usage:**

void parse\_arguments(int argc, char \*argv[]) {

// Code to parse arguments and set display\_format, show\_tcp, show\_udp, show\_icmp

}

**Debugging**

* Use the command in terminal “ ps -ef | grep server ” to find the process.
* Then use “ pstree -p <pid> ” to view the process tree.

# **Conclusion**

This project demonstrates the effective use of shared memory and multi-threading in a C application to manage real-time data display. By simulating packet capture and analyzing network traffic, the application showcases practical use cases for inter-process communication and synchronization, ensuring data consistency and real-time updates. The flexible UI display options enhance the usability of the application, providing clear and customizable views of the packet statistics.