Project Report

On

IMPLEMENT IP SUBNETTING

Submitted in partial fulfilment of the requirements for the award of

BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE & ENGINEERING

(Artificial Intelligence & Machine Learning)

by

Ms. V. KEAVALYA(22WH1A6618)

Ms. K. LAXMI PRASANNA (22WH1A6620)

Ms. E. LAHARI(22WH1A6637)

Ms. A. GAYATHRI (22WH1A6656)

Under the esteemed guidance of

Ms. P Anusha

Assistant Professor, CSE(AI&ML)

BVRIT HYDERABAD College of Engineering for Women

(UGC Autonomous Institution | Approved by AICTE | Affiliated to JNTUH)

(NAAC Accredited - A Grade | NBA Accredited B.Tech. (EEE, ECE, CSE and IT)

Bachupally, Hyderabad – 500090

2024-25

ABSTRACT

This program implements a foundational IP Subnetting Calculator designed to assist in dividing a network into smaller, manageable subnetworks. Its primary goal is to provide a simple yet effective method of calculating key subnetting parameters such as the network address, broadcast address, and the range of usable IP addresses.

The program is built around a straightforward command-line interface (CLI) that allows users to input an IP address and subnet mask. By leveraging efficient binary arithmetic, the system ensures accurate calculations and provides essential information for network configuration and management..

1. Initialization

    - This initialization phase eliminates the need for external libraries or databases, ensuring a lightweight and portable implementation suitable for local network setups.

2. Address calculation

    - The network address is determined using a logical AND operation between the IP address and the subnet mask, which identifies the starting point of the subnet.
    - The broadcast address is computed using a logical OR operation with the complement of the subnet mask, marking the endpoint of the subnet.

3. Usable IP range

    - After determining the network and broadcast addresses, the system calculates the range of usable IP addresses by incrementing the network address and decrementing the broadcast address.
    - This range provides all valid host addresses within the subnet, excluding the network and broadcast addresses themselves. This information is critical for configuring devices and assigning IPs in a network.

PROBLEM STATEMENT

This project is centred on creating a C program that serves as an IP Subnetting Calculator, designed to simplify subnetting tasks in computer networks. Users can input an IP address and subnet mask in standard dotted-decimal notation, which the program validates to ensure correctness. Once validated, the program performs binary operations to compute critical subnetting parameters, such as the network address, broadcast address, and the range of usable IP addresses, displaying results with clear and user-friendly messages.

The program emphasizes accuracy and traceability by leveraging bitwise operations for precise calculations and maintaining a log file named subnet_calculations.log. This log records successful calculations, including user inputs, results, and timestamps, facilitating audits and reviews. It also manages file operation errors gracefully, informing users of issues like the inability to write to the log file. This ensures a seamless and reliable user experience.

Secure resource management is a priority, with the program ensuring proper handling of files and memory to avoid resource leaks. Designed for accuracy, reliability, and ease of use, this tool is invaluable for network engineers, IT professionals, and students. It not only simplifies complex subnetting tasks but also provides an educational platform for understanding foundational networking concepts.

FUNCTIONAL REQUIREMENTS

1. **User Input Validation**:
   The system must validate the IP address and subnet mask entered by the user to ensure they are in the correct dotted-decimal format and within valid ranges.

2. **Subnet Calculation**:
   The system should compute and display the following based on the valid IP address and subnet mask:
   - **Network Address**
   - **Broadcast Address**
   - **Usable IP Range**

3. **Error Handling for Invalid Input**:
   If the user provides an invalid IP address or subnet mask, the system must deny further calculations and prompt the user to re-enter valid inputs.

4. **Logging Results**:
   After successful computation, the program must log the following details into a file named subnet_calculations.log: User-provided IP address and subnet mask. Calculated network address, broadcast address, and usable IP range.

5. **File Operation Handling:**
   Handle file-related errors gracefully with appropriate messages.

6. **Program Termination**:
   Exit only after successful logging and computation.

NON-FUNCTIONAL REQUIREMENTS:

1. **Usability**:

The system should provide a simple and intuitive command-line interface with clear prompts for user input and concise error messages for invalid entries.

2. **Performance**:

The system should perform calculations and respond to user inputs promptly, ensuring efficient handling of subnetting tasks.

3. **Reliability**:

The program should operate consistently without crashes or unexpected behavior even when processing invalid inputs or encountering file operation errors.

4. **Accuracy**:

The system must perform precise binary arithmetic to ensure that all calculated results, such as the network address, broadcast address, and usable IP range, are correct.

5. **Security**:

The program should handle sensitive operations securely, such as preventing unintended

6. **Maintainability**:

The code should be clean, modular, and well-documented, making it easy to update or extend functionality, such as supporting additional subnetting features.

SOURCE CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

// Function to convert an IP string to an integer array
void ipToArray(const char *ip, int arr[4]) {
    sscanf(ip, "%d.%d.%d.%d", &arr[0], &arr[1], &arr[2], &arr[3]);
}

// Function to calculate the network address
void calculateNetworkAddress(int ip[4], int mask[4], int network[4]) {
    for (int i = 0; i < 4; i++) {
        network[i] = ip[i] & mask[i];
    }
}

// Function to calculate the broadcast address
void calculateBroadcastAddress(int ip[4], int mask[4], int broadcast[4]) {
    for (int i = 0; i < 4; i++) {
        broadcast[i] = ip[i] | (~mask[i] & 255);
    }
}

// Function to print an IP address
void printIP(const char *label, int ip[4]) {
    printf("%s: %d.%d.%d.%d\n", label, ip[0], ip[1], ip[2], ip[3]);
}
```

```
int main() {
    char ipStr[16], maskStr[16];
    int ip[4], mask[4], network[4], broadcast[4];

    // Input IP address and subnet mask
    printf("Enter IP address (e.g., 192.168.1.10): ");
    scanf("%15s", ipStr);
    printf("Enter Subnet Mask (e.g., 255.255.255.0): ");
    scanf("%15s", maskStr);

    // Convert strings to integer arrays
    ipToArray(ipStr, ip);
    ipToArray(maskStr, mask);

    // Calculate network address and broadcast address
    calculateNetworkAddress(ip, mask, network);
    calculateBroadcastAddress(ip, mask, broadcast);

    // Print results
    printIP("IP Address", ip);
    printIP("Subnet Mask", mask);
    printIP("Network Address", network);
    printIP("Broadcast Address", broadcast);

    // Calculate the range of usable IP addresses
    int firstUsable[4], lastUsable[4];
    for (int i = 0; i < 4; i++) {
        firstUsable[i] = network[i];
        lastUsable[i] = broadcast[i];
    }
```

```
    firstUsable[3] += 1; // First usable IP
    lastUsable[3] -= 1;  // Last usable IP

    printIP("First Usable IP", firstUsable);
    printIP("Last Usable IP", lastUsable);

    return 0;
}
```

OUTPUT

```
Enter IP address (e.g., 192.168.1.10): 192.168.1.1:8090
Enter Subnet Mask (e.g., 255.255.255.0): IP Address: 192.168.1.1
Subnet Mask: 0.0.286052357.0
Network Address: 0.0.1.0
Broadcast Address: 255.255.251.255
First Usable IP: 0.0.1.1
Last Usable IP: 255.255.251.254


=== Code Execution Successful ===
```