MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Mini Project Report

of

Computer Networks LAB

IP Address Validator

SUBMITTED
BY

Abhinav Modi 200905280 49 D

Pushkar Anand 200905414 63 D

Table of Contents

Acknowledgement

Our gratitude goes out first and foremost to our mentor, **Dr Roopalakshmi R,** who was a continual source of inspiration. Throughout the project, she encouraged us to think imaginatively and to proceed without hesitation. Her extensive knowledge, experience, and professional competency in Computer Networks enabled us to accomplish this project. Her assistance and supervision were crucial to the success of this endeavour. As students, we could not have asked for a more exemplary mentor.

In addition, we would like to express our gratitude to our Head of Department, **Dr Ashalatha Nayak**, and the college for offering us the opportunity to work on this project.

Finally, we would like to thank everyone who has contributed directly or indirectly to this project.

Abstract

IP addresses are the identifiers that allow information to be transmitted between network devices: they contain location information and allow devices to communicate with one another. The internet requires a method to distinguish between various computers, routers, and webpages. IP addresses enable this and are a critical component of the internet's operation. An IP address is a numeric string separated by periods. IP addresses are composed of four digits. Each integer in the set can have a value ranging from 0 to 255. As a result, the whole IP addressing range is 0.0.0.0 to 255.255.255.255.

A Subnet Mask is a 32-bit combination that specifies which component of an address relates to the subnet and which to the host.

- The tool accepts an IP address and a mask as independent inputs.
- The utility determines whether or not the specified IP address is legitimate.
- It then displays the first, last, and number of addresses in the block while keeping the size of this block constant and providing three more blocks of the same size with their network addresses.

Introduction

IP address validation is a process of verifying the validity of an IP address. The process includes checking the subnet mask, network mask and prefix length.

IP address validation is a necessary step in any networking setup. It's also used to verify that an IP address is valid before it can be used for internet access or other purposes.

The subnet mask is a 32-bit number in the octet (8-bit) format that tells which part of the IP address should be considered as network and which part should be considered as host.

The network mask tells us what section of the IP address we should consider as a network, and what section we should consider as a host.

The validation process starts by checking if the first octet, or set of numbers, in the IP address matches the first octet in the network mask. If they match, then they continue on to check if all other octets match as well.

If there are any discrepancies between these two numbers, then it will be considered an invalid IP Address and should not be used on any device that connects to a network.

Problem

If we are developing some web applications in Django, Flask, Node or any other backend framework, we may need to verify that a user's IP address is correct. Among other things, IP address validation is essential for preventing fraud and providing targeted location services for users.

The IP address validation problem is a common issue that can make it difficult to access web pages or services on the internet. IP addresses identify computers, servers, and other devices connected to the internet.

One of the most common errors that people make when configuring their network is to input an invalid IP address. This can be caused by incorrectly typing the IP address, entering a subnet mask that does not match the subnet size, or entering an incorrect gateway.

The IP address validation process is used to verify that an IP address belongs to a specific network.

Objectives

The goal of our Computer Networks project is to create an IP address validator that takes an IP address as input and notifies the user whether or not the IP address is genuine.

The project also displays the total amount of addresses within the block that hosts can use to join to the network.

In addition to validating the IP address, our solution uses the provided network mask to generate a random address block of the same size as the entered IP address.

Methodology

The program takes the IP address and mask from the user as input.

Using this information the program first validates if the entered IP Address is valid or not. If it is invalid, it exits the program and displays an error message.

The input is taken in string format which is then parsed into int for easier calculations and comparisons.

There are 3 functions in our program:-

- One main function that takes input from the user, and displays the first and last IP address of the network entered along with three random IP addresses in the network.
- The other function is isValid() which is simply used to validate the entered IP address.
- The last function called generator is used to convert each octet of the IP address to a String, which is initially stored in an array.
- This function converts each of the four IP address segments into binary octets which is later used to do computation and generate random address blocks.

Implementation

I. Code:-

- For the project implementation, we have used JAVA as the language.

```
import java.util.*;
import java.io.*;
import java.lang.Math;
import java.lang.*;
class validator {
   //function to check if entered IP is valid
   static boolean isvalid(String[] arr) {
      int x;
      for (int i = 0; i < 4; i++) {
         x = Integer.parseInt(arr[i]);
         if (x < 0 || x >= 255)
            return false;
      }
      return true;
   }
   //function to convert each octet of the entered IP in array form to a
String
   static String func(int[] arr) {
      int val = 0;
      String bottom = "";
      int i = 0;
      for (int j = 7; j >= 0; j--) {
         double temp = Math.pow(2, i);
         int x = (int) temp;
         val += (x * arr[j]);
         i++;
      }
      String f = Integer.toString(val);
      bottom += f;
      bottom += ".";
      val = 0;
      i = 0;
      for (int j = 15; j > 7; j--) {
        double temp = Math.pow(2, i);
        int x = (int) temp;
        val += (x * arr[j]);
        i++;
    }
    f = Integer.toString(val);
    bottom += f;
    bottom += ".";
    val = 0;
    i = 0;
    for (int j = 23; j > 15; j--) {
        double temp = Math.pow(2, i);
        int x = (int) temp;
        val += (x * arr[j]);
        i++;
    }
    f = Integer.toString(val);
    bottom += f;
    bottom += ".";
    val = 0;
    i = 0;
    for (int j = 31; j > 23; j--) {
        double temp = Math.pow(2, i);
        int x = (int) temp;
        val += (x * arr[j]);
        i++;
    }
    f = Integer.toString(val);
    bottom += f;
    val = 0;
    i = 0;
    return bottom;
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("\nEnter IP address-");
    String str1;
    str1 = sc.nextLine();
    System.out.println("Enter mask-");
    int mask;
    mask = sc.nextInt();
    String[] valid = str1.split("\\.");
    if (isvalid(valid)) {
        System.out.println("Valid Ip entered");
    } else {
```

```
    System.out.println("Invalid Ip entered");
    System.exit(0);
}
int[] arr1 = new int[32];
for (int i = 0; i < 4; i++) {
    int temp = Integer.parseInt(valid[i]);
    String binary = Integer.toBinaryString(temp);
    if (i == 0) {
        int j = 7;
        int x = binary.length();
        x--;
        while (x >= 0) {
            char temp2 = binary.charAt(x);
            if (temp2 == '0')
                arr1[j--] = 0;
            else
                arr1[j--] = 1;
            x--;
        }
    } else if (i == 1) {
        int j = 15;
        int x = binary.length();
        x--;
        while (x >= 0 && j > 7) {
            char temp2 = binary.charAt(x);
            if (temp2 == '0')
                arr1[j--] = 0;
            else
                arr1[j--] = 1;
            x--;
        }
    } else if (i == 2) {
        int j = 23;
        int x = binary.length();
        x--;
        while (x >= 0 && j > 15) {
            char temp2 = binary.charAt(x);
            if (temp2 == '0')
                arr1[j--] = 0;
            else
                arr1[j--] = 1;
            x--;
        }
    } else if (i == 3) {
        int j = 31;
        int x = binary.length();
        x--;
        while (x >= 0 && j > 23) {
            char temp2 = binary.charAt(x);
            if (temp2 == '0')
                arr1[j--] = 0;
            else
                arr1[j--] = 1;
            x--;
        }
    }
}
int[] first = new int[32];
int[] last = new int[32];
for (int i = 0; i < mask; i++) {
    first[i] = arr1[i];
    last[i] = arr1[i];
}
for (int i = mask; i < 32; i++) {
    first[i] = 0;
    last[i] = 1;
}
int[] randomAddFirst = new int[32];
int[] randomAddLast = new int[32];
String first_address = func(first);
System.out.println("First address- " + first_address);
String last_address = func(last);
System.out.println("\nLast address- " + last_address);
double range = Math.pow(2, 32 - mask);
int r = (int) range;
System.out.println("\nNumber of addresses in the block- " +
    range + "\n");
//generating three blocks with the same block size
System.out.println("\n3 RANDOM ADDRESS BLOCKS: ");
for (int j = 0; j < 3; j++) {
    for (int i = 0; i < 7; i++) {
        randomAddFirst[i] = randomAddLast[i] =
            (int) Math.floor(Math.random() * (1 - 0 + 1) + 0);
    }
    for (int i = 8; i < 32; i++) {
        randomAddFirst[i] = first[i];
        randomAddLast[i] = last[i];
    }
    String testFirst = func(randomAddFirst);
    String testLast = func(randomAddLast);
```

```
            System.out.print("First Address: ");
            System.out.println(testFirst);
            System.out.print("Last Address: ");
            System.out.println(testLast + "\n");
        }
    }
}
```

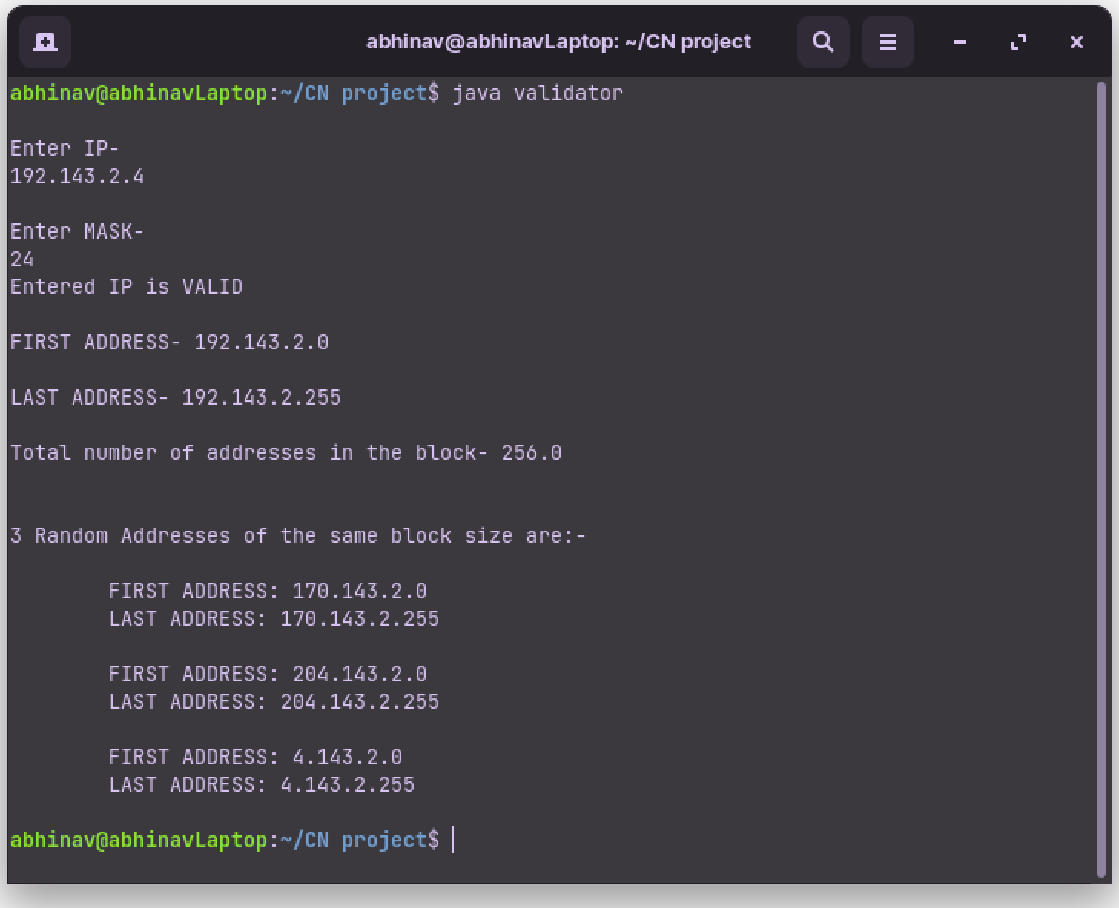
II. Output:-

Depending on the mask number entered, the output will have multiple cases, but eventually, there will only be valid and invalid results

Here we have shown 3 different output cases:-

1. Case 1:-

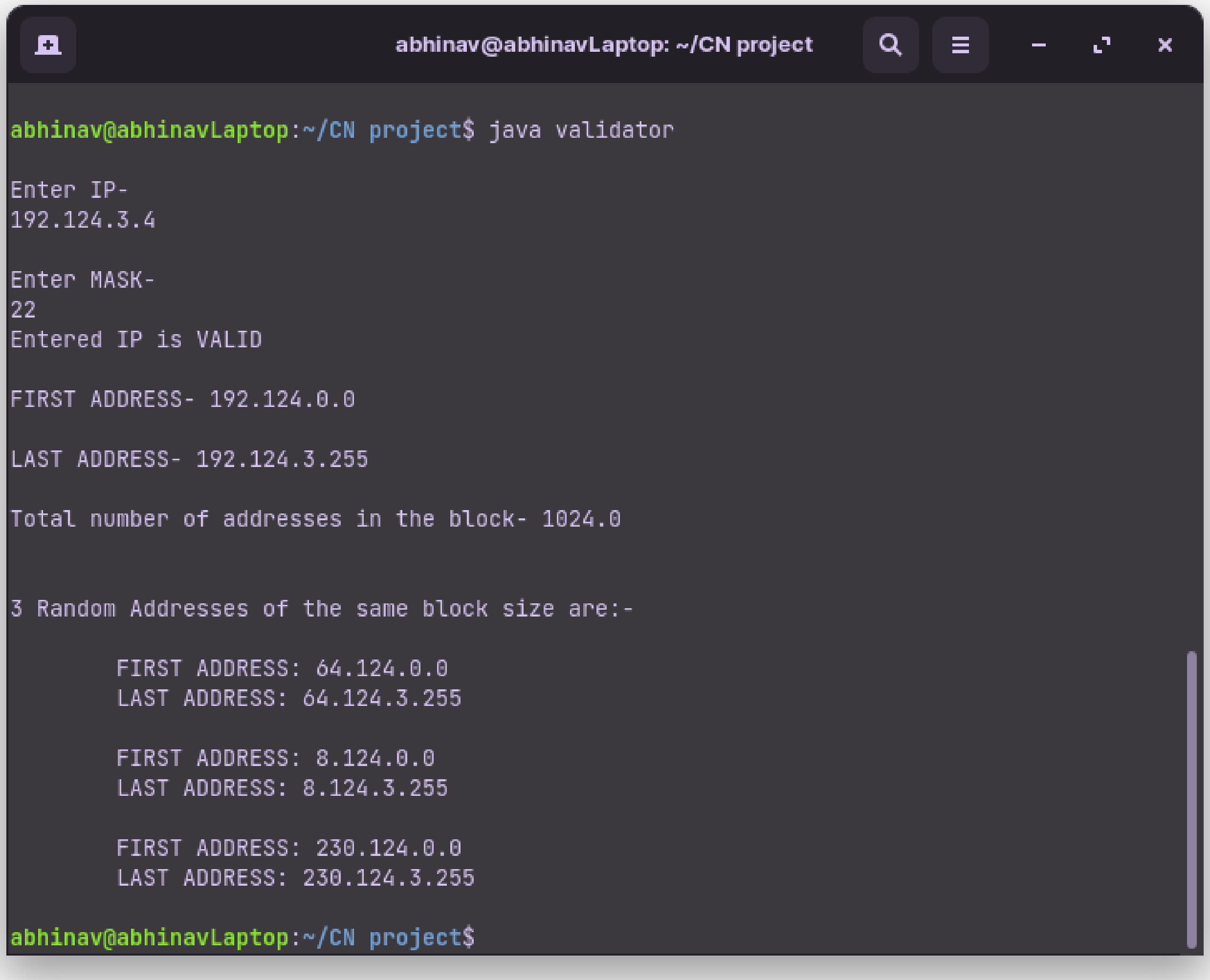
The blocks of the provided subnet mask have 2(32 - 24) = 256 addresses. As we can see, the application created three Random Address Blocks of the same size (256) with the same subnet mask (255.255.255.0).



```
abhinav@abhinavLaptop:~/CN project$ java validator

Enter IP-
192.143.2.4

Enter MASK-
24
Entered IP is VALID

FIRST ADDRESS- 192.143.2.0

LAST ADDRESS- 192.143.2.255

Total number of addresses in the block- 256.0


3 Random Addresses of the same block size are:-

        FIRST ADDRESS: 170.143.2.0
        LAST ADDRESS: 170.143.2.255

        FIRST ADDRESS: 204.143.2.0
        LAST ADDRESS: 204.143.2.255

        FIRST ADDRESS: 4.143.2.0
        LAST ADDRESS: 4.143.2.255

abhinav@abhinavLaptop:~/CN project$ 
```

2. Case 2:-

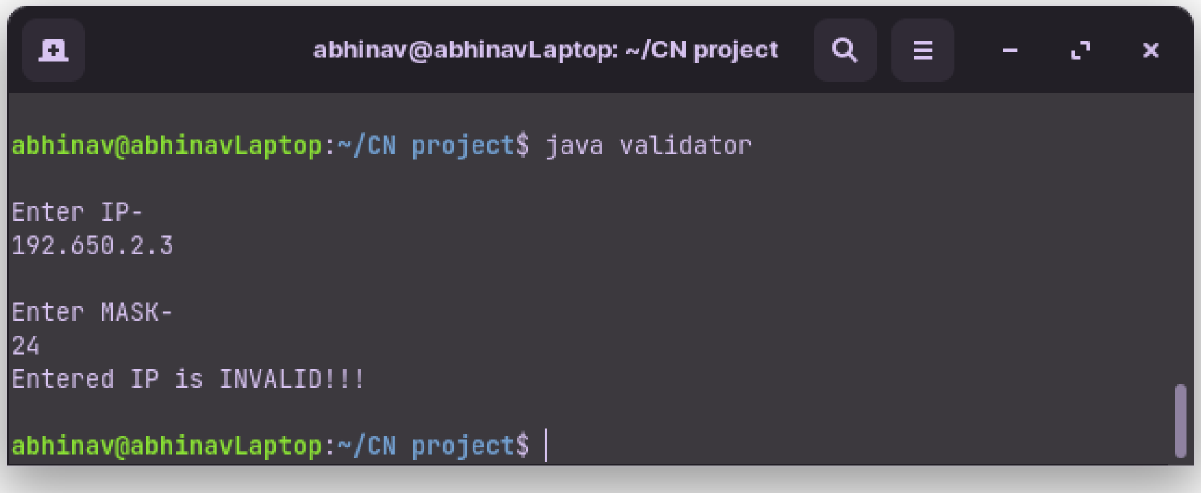
The blocks of the provided subnet mask have 2(32 - 22) = 1024 addresses. As we can see, the application created three Random Address Blocks of the same size (1024) with the same subnet mask (255.255.255.240).



```
abhinav@abhinavLaptop:~/CN project$ java validator

Enter IP-
192.124.3.4

Enter MASK-
22
Entered IP is VALID

FIRST ADDRESS- 192.124.0.0

LAST ADDRESS- 192.124.3.255

Total number of addresses in the block- 1024.0


3 Random Addresses of the same block size are:-

        FIRST ADDRESS: 64.124.0.0
        LAST ADDRESS: 64.124.3.255

        FIRST ADDRESS: 8.124.0.0
        LAST ADDRESS: 8.124.3.255

        FIRST ADDRESS: 230.124.0.0
        LAST ADDRESS: 230.124.3.255

abhinav@abhinavLaptop:~/CN project$
```

3. Case 3:-

Since the Octet values should range from 0 to 255 (inclusive), an error is thrown in the following output.



```
abhinav@abhinavLaptop:~/CN project$ java validator

Enter IP-
192.650.2.3

Enter MASK-
24
Entered IP is INVALID!!!

abhinav@abhinavLaptop:~/CN project$ 
```

Contribution Summary

The project's implementation part was done through a discussion among the team members; we made a structured plan and discussed the pseudocode together

Once the pseudocode part was done we divided the work

Abhinav: responsible for writing the main function that takes user input and generates 3 random address blocks and converts the entered IP address into binary form for computation.

Also responsible for making the Objectives, Problem and Implementation part of the report

Pushkar: responsible for the generator function and valid function that validates IP and converts the Octets into a string and generates the first and last address.

Also responsible for making Abstract, Acknowledgement, Introduction and Methodology part of the report.

References

https://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13788-3.html

https://www.techiedelight.com/validate-ip-address-java/

https://www.omnisecu.com/tcpip/how-to-find-network-address-and-broadcast-address-of-a-subnetted-ipv4-address.php

https://www.geeksforgeeks.org/supernetting-in-network-layer/