Subnet Calculator with IP Validation

A Mini Project Report

Submitted By

Prodeep Ghosh 200905384

In partial fulfilment for the award of the degree of

Bachelor's in Technology (B. Tech)

IN

Computer Science & Engineering

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
A Constituent Institution of Manipal University

Department of Computer Science & Engineering

Department of Computer Science & Engineering

BONAFIDE CERTIFICATE

Certified that this project report "**Subnet Calculator with IP Validation**"
is the bonafide work of:

Prodeep Ghosh 200905384
who carried out the mini project work under my supervision.

Dr. Ashalatha Nayak	Radhakrishna Bhat
Head of Department, CSE	Supervisor

Submitted to the Viva voce Examination held on

_____

EXAMINER 1 **EXAMINER 2**

ABSTRACT

This document gives a brief description of how my project was designed and developed . The idea of my project was to create a fully working Subnetwork Calculator tool which takes as input the ip address along with subnet mask in CIDR format , checks the validity of IP address and displays the network configuration details such as first address/Network ID, last address/Broadcast ID, IP class , range of IP address, total number of usable hosts per network and total number of addresses.Moreover , it will create subnets of equal block sizes from the given IP address as input and show all the relevant details including the number of hosts in that network and first and last address.

The program will work with three classes of IP address, such as :A,B and C. The program will calculate and show you the network address , first and last address both in binary and decimal notation along with total number of addresses and usable hosts in that particular network

To implement the project , I used c++ programming language which has inbuild STL library

INTRODUCTION

An Internet Protocol address (IP address) is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication. An IP address serves two main functions: host or network interface identification and location addressing. There are 2 types/categories of IP addressees namely:

IPV4 and IPV6

Internet Protocol version 4 (IPv4) defines an IP address as a 32-bit number.IPV4 Addresses are canonically represented in dot-decimal notation, which consists of four decimal numbers, each ranging from 0 to 255, separated by dots, e.g., 172.16.254.1 However, because of the growth of the Internet and the depletion of available IPv4 addresses, a new version of IP(IPv6), using 128 bits for the IP address, was standardized in 1998. IPv6 deployment has been ongoing since the mid-2000s. The size of the routing prefix ofthe address is designated in CIDR notation by suffixing the address with the number of significant bits .

Public IP Addresses:

- A public IP address is an address where one primary address is associated with your whole network. In this type of IP address, each of the connected devices has the same IP address.

- This type of public IP address is provided to your router by your ISP.

Private IP Addresses:

- A private IP address is a unique IP number assigned to every device that connects to your home internet network, which includes devices like computers, tablets, smart phones, which is used in your household.

- It also likely includes all types of Bluetooth devices you use, like printers or printers, smart devices like TV, etc. With a rising industry of internet of things(IOT) products, the number of private IP addresses you are likely to have inyour own home is growing.

Dynamic IP address:

- Dynamic IP addresses always keep changing. It is temporary and are allocated to a device every time it connects to the web. Dynamic IPs can trace their origin to a collection of IP addresses that are shared across many computers.

- Dynamic IP addresses are another important type of internet protocol addresses.It is active for a specific amount of time; after that, it will expire.

Static IP Addresses:

- A static IP address is an IP address that cannot be changed. In contrast, adynamic IP address will be assigned by a Dynamic Host Configuration Protocol

(DHCP) server, which is subject to change. Static IP address never changes, but it can be altered as part of routine network administration.

Static IP addresses are consistent, which is assigned once, that stays the same over the years. This type of IP also helps you procure a lot of information about a device.

Some Important Terms

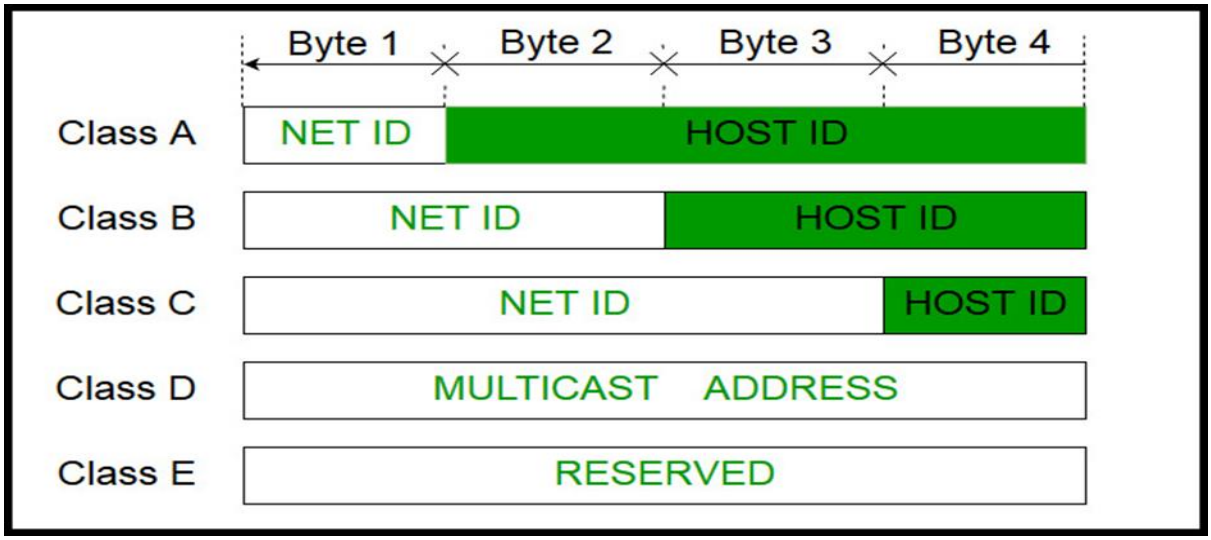
Default Mask:
An address mask determines which portion of an IP address represents network number and which part represents host number, e.g., IP address, the mask has four octets. If a given bit of the mask is 1, the corresponding bit of the IP address is the in-network portion and if the given bit of the mask is 0, the corresponding bit of the IP address is in the host portion.

CIDR:
Classless Inter-Domain Routing uses slash(/) notation to specify the mask with IPv4 address. The address is given as x.y.z.t/n where x.y.z.t is the IP address and n is the number of 1's in the default mask.

Network address:
The first address of a network is a network address. It is obtained by ANDing the mask with the IP address (Both in binary form). Another method is to set the last 32-n bits of the IP address to 0.

Broadcast address:
The last address of a network is a broadcast address. It is obtained by ORing the complement mask with the IP address(Both in binary form). Another method is to set the last 32-n bits of the IP address to 1.

Class:
The class of an address is identified by the first byte of the address. There are currently five classes A, B, C, D, and E. The range of the first byte of each class is:



PROBLEM DEFINITION

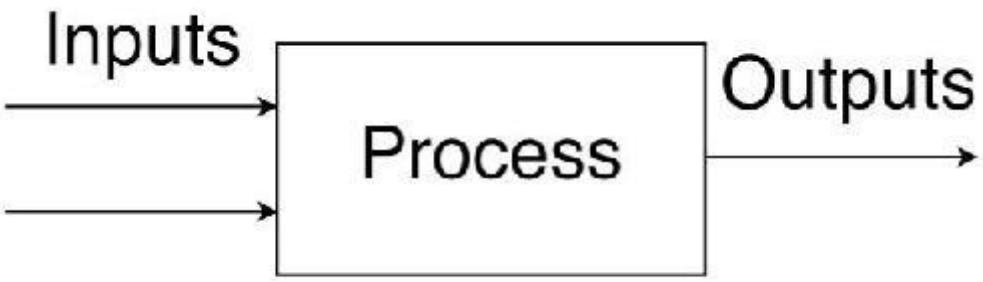
Design a tool which, take ip along with mask as input and, then, give the validity of the ip,
along with first address, last address and the number of address in the block. Keeping the size of this block as constant, it should give at least other three blocks available of the same size with their network address.

OBJECTIVES

The purpose or objective of this mini project is to develop an easy -to-use IP subnetwork calculator that lets individuals using it to calculate every aspect of subnet configuration

This project will help any individual blocks providing them decreased network traffic, flexibility, improved troubleshooting and utilization of addresses

METHODOLOGY



Features and flow of Subnet Calculator:

a.) Take a user IP Address and Subnet Mask

b.) Checks Validity of IP Address

c.) If it is not a valid IP address, it displays the result not valid

d.) If it is a valid IP address , it shows the ip address in both decimal and binary notation

e.) Determines IP Address Class

f.) Shows First Address in binary and dotted decimal representation

g.) Shows Last Address in binary and dotted decimal representation

h.) Range of IP address

i.) Shows Total Number of Addresses

j.) Shows Total Number of usable hosts per network

k.) Shows the number of hosts,first and last address of ‘n’ subnets

IMPLEMENTATION

The program is divided into 2 parts . The first part includes checking the validity of IP address using “.” as delimiter and the second part is the calculation part which includes conversion of decimal to binary , string to integers , setting up of host bits as 0 or 1 as per requirements .

Below is the logic for checking validity of Ipv4 address

step 1)

Parse string with “.” as delimiter using ‘strok()’ function.

step 2)

A) If ptr contains any character which is not digit then return 0

B) Convert “ptr” to decimal number say ‘NUM’

C) If NUM is not in range of 0-255 return 0

D) If NUM is in range of 0-255 and ptr is non-NULL increment “dot_counter” by 1

E) if ptr is NULL goto step 3 else goto step 1

step 3)

If dot_counter != 3

return 0

else return 1

The second part includes the subnet calculation where in we take the number of networks as input. Based on this input, we calculate the Subnet bits required for the action (greatest power of 2) and display the various subnet details.

CODE SNIPPETS

```
bool valid_part(char* s)
{
    int n = strlen(s);


    if (n > 3)
        return false;


    for (int i = 0; i < n; i++)
        if ((s[i] >= '0' && s[i] <= '9') == false)
            return false;
    string str(s);


    if (str.find('0') == 0 && n > 1)
        return false;

    stringstream geek(str);
    int x;
    geek >> x;

    return (x >= 0 && x <= 255);
}
```

```
int is_valid_ip(char* ip_str)
{

    if (ip_str == NULL)
        return 0;
    int i, num, dots = 0;
    int len = strlen(ip_str);
    int count = 0;


    for (int i = 0; i < len; i++)
        if (ip_str[i] == '.')
            count++;
    if (count != 3)
        return false;

    char *ptr = strtok(ip_str, DELIM);
    if (ptr == NULL)
        return 0;

    while (ptr) {


        if (valid_part(ptr))
        {

            ptr = strtok(NULL, ".");
            if (ptr != NULL)
                ++dots;
        }
        else
            return 0;
    }

    if (dots != 3)
        return 0;
    return 1;
}
```

```
vector<int> bina(vector<string> str)
    {
    vector<int> re(32,0);
        int a, b, c, d, i, rem;
        a = b = c = d = 1;
        stack<int> st;

            a = stoi(str[0]);
            b = stoi(str[1]);
            c = stoi(str[2]);
            d = stoi(str[3]);

        // convert first number to binary
        for (i = 0; i <= 7; i++)
        {
            rem = a % 2;
            st.push(rem);
            a = a / 2;
        }

        // Obtain First octet
        for (i = 0; i <= 7; i++) {
            re[i] = st.top();
            st.pop();
        }

        for (i = 8; i <= 15; i++) {
            rem = b % 2;
            st.push(rem);
            b = b / 2;
        }

        for (i = 8; i <= 15; i++) {
            re[i] = st.top();
            st.pop();
        }
```

```
        for (i = 16; i <= 23; i++) {
            rem = c % 2;
            st.push(rem);
            c = c / 2;
        }

        for (i = 16; i <= 23; i++) {
            re[i] = st.top();
            st.pop();
        }

        for (i = 24; i <= 31; i++) {
            rem = d % 2;
            st.push(rem);
            d = d / 2;
        }

    for (i = 24; i <= 31; i++) {
            re[i] = st.top();
            st.pop();
        }

        return (re);
    }
```

```
char cls(vector<string> str)
    {
        int a = stoi(str[0]);
        if (a >= 0 && a <= 127)
            return ('A');
        else if (a >= 128 && a <= 191)
            return ('B');
        else if (a >= 192 && a <= 223)
            return ('C');
        else if (a >= 224 && a <= 239)
            return ('D');
        else
            return ('E');
    }
```

```
vector<int> deci(vector<int> bi)
    {
        vector<int> arr(4,0);
        int a, b, c, d, i, j;
        a = b = c = d = 0;
        j = 7;

        for (i = 0; i < 8; i++) {

            a = a + (int)(pow(2, j)) * bi[i];
            j--;
        }

        j = 7;
        for (i = 8; i < 16; i++) {

            b = b + bi[i] * (int)(pow(2, j));
            j--;
        }

        j = 7;
        for (i = 16; i < 24; i++) {

            c = c + bi[i] * (int)(pow(2, j));
            j--;
        }
        j = 7;
        for (i = 24; i < 32; i++) {

            d = d + bi[i] * (int)(pow(2, j));
            j--;
        }
        arr[0] = a;
        arr[1] = b;
        arr[2] = c;
        arr[3] = d;
        return arr;
    }
```

```
    void printSubnetInfo(u_int32_t* addressOctets, int* CIDR, int* subnetBits) {

    u_int32_t netAddress;
    u_int32_t netMask;

    netMask = (0xFFFFFFFF << (32 - (*CIDR + *subnetBits)) & 0xFFFFFFFF);
    netAddress = *addressOctets & netMask;

    // Unpack and display the network address
    printf("\nNetwork address: %d.%d.%d.%d/%d\n", (netAddress >> 24) & 0xFF, (netAddress >> 16) & 0xFF,
                        (netAddress >> 8) & 0xFF, (netAddress) & 0xFF, *CIDR + *subnetBits);

    // Subtract the network address from the broadcast address and take one from the result for total hosts
    printf("Total hosts: %d\n", ((netAddress | ~netMask) - netAddress) - 1);

    // Display the first host address by adding to each of our unpacked octets
    printf("First host address: %d.%d.%d.%d\n", ((netAddress + 1) >> 24) & 0xFF, ((netAddress + 1) >> 16) & 0xFF,
                        ((netAddress + 1) >> 8) & 0xFF, (netAddress + 1) & 0xFF);

    // Bitwise OR the address int with the negated mask to get the broadcast address in the variable
    netAddress = netAddress | ~netMask;

    // Subtract from the from the broadcast address for the final host address
    printf("Last host address: %d.%d.%d.%d\n", ((netAddress - 1) >> 24) & 0xFF, ((netAddress - 1) >> 16) & 0xFF,
                        ((netAddress - 1) >> 8) & 0xFF, (netAddress - 1) & 0xFF);

    // Unpack and display the broadcast address
    printf("Broadcast address: %d.%d.%d.%d\n", (netAddress >> 24) & 0xFF, (netAddress >> 16) & 0xFF,
                        (netAddress >> 8) & 0xFF, (netAddress) & 0xFF);
}
```

```
int main()
    {   cout<<"-----------------------------------------------"<<endl;
        cout<<"------SUBNET CALCULATOR WITH IP VALIDATOR---"<<endl;
         cout<<"-----------------------------------------------"<<endl;
         cout<<endl;cout<<endl;cout<<endl;

        string ipr = "192.168.100.0/26";
                char ip1[]="192.168.100.0";

                char ipAddress[]="192.168.512.0";
                    ipAddress[strlen(ipAddress)-1] = '\0';

        cout<<"IP address CIDR format is:"<< ipr;
        cout<<endl;

        if(is_valid_ip(ip1) )
        cout<<"Valid IP address\n" ;

        else if(is_valid_ip(ip1)==0)
        {   cout<<"Not valid IP address\n";
            exit(-1);
        }

        // Separate IP address and n
        string str1 = "";
        int idx = 0;
        int len = ipr.size();
        len -= 3;
        //will run 14 times for above ip
        while(len--){
            str1 += ipr[idx];
            idx++;
        }
        cout<<endl;
        cout<<"IP Address : " <<str1<<endl;//stores ip without network mask

        string str2 = "";
        idx++;
        str2 += ipr[idx];
        idx++;
        str2 += ipr[idx];
        cout<<endl;
        cout<<"Value of n : "<< str2<<endl;//stores network mask;
        // IP address
        string tr = str1;
```

```
// Split IP address into 4 subparts x, y, z, t

vector<string> str;

string temp;
int n = tr.size();
for(int i = 0; i < n; i++){

        //0...9, then append string to temp
    if(tr[i] >= 48 && tr[i] <= 57)
        temp +=tr[i];
    else{       //if . encountered , push entire string till dot in vector and clear temp
        str.push_back(temp);
        temp = "";
    }
}
str.push_back(temp);//


vector<int> b;

cout<<endl;
b = bina(str);

cout<<"IP address in binary:"<<endl;

    //printing ip ad in binary
```

```
  n = stoi(str2);
  vector<int> ntwk(32,0);
  vector<int> brd(32,0);
  int t = 32 - n;

  for (int i = 0; i <= (31 - t); i++) {

      ntwk[i] = b[i];
      brd[i] = b[i];
  }

  // Set 32-n bits to 0
  for (int i = 31; i > (31 - t); i--) {

      ntwk[i] = 0;
  }
  cout<<endl;
  cout<<"First address in binary:"<<endl;
  for(int i=0;i<8;i++){
          cout<<ntwk[i];
  }
  cout<<"  ";
      for(int i=8;i<16;i++){
          cout<<ntwk[i];
  }cout<<"  ";
      for(int i=16;i<24;i++){
          cout<<ntwk[i];
  }cout<<"  ";
      for(int i=24;i<32;i++){
          cout<<ntwk[i];
  }cout<<"  ";
  cout<<endl;


  // setting 32-n bits of broadcast address to 1
  for (int i = 31; i > (31 - t); i--) {

      brd[i] = 1;
  }

  cout<<endl;
```

```
    int subnetNumber;
    int subnetBits = 0;
    int totalSubnets = 0;
    u_int32_t currentSubnet;
    u_int32_t addressOctets;
    unsigned char* octetArray;
    octetArray = new unsigned char[4];
    int x=ipVerify(ipAddress, octetArray);

    char buffer[256];

    addressOctets = (octetArray[0] << 24) | (octetArray[1] << 16) | (octetArray[2] << 8) | (octetArray[3]);

        do {
    printf("Enter number of required networks, or q to quit: ");
    fgets(buffer, 4, stdin);
    subnetNumber = atoi(buffer);

    if (subnetNumber == 0) {
        printf("Exiting...\n");
        exit(0);
    }

    // Determine the amount of bits required to contain the required networks
    while (subnetNumber > totalSubnets) {
        subnetBits++;
        totalSubnets = pow(2, subnetBits);
    }

    // Check we have the required amount of bits to subnet successfully
    if ((n + subnetBits) > 31) {
        printf("Amount of networks too large to be accommodated.\n");
    }
} while ((n + subnetBits) > 31);

printf("\nTotal subnets to be created: %d\n--------------------------------", totalSubnets);
        for (int i=0; i<totalSubnets; i++) {
    currentSubnet = (addressOctets & ((0xFFFFFFFF << (32 - n)) & 0xFFFFFFFF))
            | i << (32 - (n + subnetBits));
    printSubnetInfo(&currentSubnet, &n, &subnetBits);
}




    return 0;
    }
```

OUTPUT

```
-----------------------------------------
------SUBNET CALCULATOR WITH IP VALIDATOR---
-----------------------------------------



IP address CIDR format is:192.168.100.0/26
Valid IP address

IP Address : 192.168.100.0

Value of n : 26

IP address in binary:
11000000  10101000  01100100  00000000

First address in binary:
11000000  10101000  01100100  00000000

Last address in binary:
11000000  10101000  01100100  00111111

Class : C

First Address : 192.168.100.0

Last Address : 192.168.100.63

IP Range:192.168.100.1 to 192.168.100.62

Total Number of Addresses :64
```

OUTPUT

```
Total Number of Addresses :64

Total Number of usable hosts per network :62
Enter number of required networks, or q to quit: 4

Total subnets to be created: 4
------------------------------
Network address: 192.168.100.0/28
Total hosts: 14
First host address: 192.168.100.1
Last host address: 192.168.100.14
Broadcast address: 192.168.100.15

Network address: 192.168.100.16/28
Total hosts: 14
First host address: 192.168.100.17
Last host address: 192.168.100.30
Broadcast address: 192.168.100.31

Network address: 192.168.100.32/28
Total hosts: 14
First host address: 192.168.100.33
Last host address: 192.168.100.46
Broadcast address: 192.168.100.47

Network address: 192.168.100.48/28
Total hosts: 14
First host address: 192.168.100.49
Last host address: 192.168.100.62
Broadcast address: 192.168.100.63


...Program finished with exit code 0
Press ENTER to exit console.
```

```
-------------------------------------------
------SUBNET CALCULATOR WITH IP VALIDATOR---
-------------------------------------------



IP address CIDR format is:167.512.170.82/27
Not valid IP address
```

REFERENCES

a) https://docs.oracle.com/cd/E19683-01/806-4075/ipref-1/index.html#:~:text=The%20IPv4%20address%20is%20a,byte%20of%20the%20IPv4%20address.

b) https://www.geeksforgeeks.org/structure-and-types-of-ip-address/

c) https://www.geeksforgeeks.org/introduction-to-subnetting/