# MIT World Peace University

Computer Networks
Second Year B. Tech, Semester 3

---

# Subnetting

---

Practical Report
Assignment 5

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

November 20, 2022

# Contents

**1 Aim and Objectives** **1**

**2 Platform** **1**

**3 Code** **1**

**4 Output** **4**

**5 Packet Tracer Implementation** **6**

**6 Conclusion** **6**

# 1 Aim and Objectives

## Aim

Write a program to implement subnetting to find subnet mask

## Objectives

To understand and learn the concept of IP address, subnet mask and subnetting

# 2 Platform

**Operating System**: Arch Linux x86-64
**IDEs or Text Editors Used**: Visual Studio Code
**Programs Used**: Cisco Packet Tracer v8.2 **Interpreter Used**: Python 3.9

# 3 Code

```
from audioop import reverse
from codecs import getdecoder
from mailbox import NotEmptyError
from os import getpid
from unittest import result
import pandas as pd
import numpy as np

ip_classes = {
    'A' : [8, 24, 126],
    'B' : [16, 16, 191],
    'C' : [24, 8, 223]
}

def get_dec_in_bin_bits(decimal_val, how_many_bits):
    '''
    Returns the given value as <how_many_bits> binary bits as a list of 1s or 0s
    '''
    octet = [0 for i in range(how_many_bits)]
    i = how_many_bits - 1
    while(decimal_val != 0):
        rem = decimal_val % 2
        decimal_val = decimal_val // 2
        octet[i] = rem
        i-=1
    return octet

def get_ip_class(ip):
    '''
    Gets the Class of the IP Address
    '''
    for i in ip:
        if i <= ip_classes['A'][2]:
            return 'A'
        elif i <= ip_classes['B'][2]:
            return 'B'
        elif i <= ip_classes['C'][2]:
```

```
38              return 'C'
39
40  def find_subnet_bits(subnet_number):
41      '''
42      Returns the Required Number of Subnet bits to Borrow from the host.
43      '''
44      for i in range(10):
45          if pow(2, i) >= subnet_number:
46              return i
47
48  def calc_subnet_mask(cidr):
49      '''
50      Returns the Subnet Mask as a string
51      '''
52      temp = [[], [], [], []]
53      subnet_masks = []
54      divs = cidr // 8
55      net_bits = cidr % 8
56      val = 0
57
58      # Makes everything 1 till divs
59      for i in range(4):
60          for j in range(8):
61              if i <= divs:
62                  temp[i].append(1)
63              else: temp[i].append(0)
64
65      # Makes the important things 1
66      for i in range(8):
67          if i < net_bits:
68              temp[divs][i] = 1
69          else: temp[divs][i] = 0
70
71      return get_ip(temp)
72
73  def get_ip(ip_list):
74      '''
75      Input : A list containing 4 lists, each having 8 int values, 0 or 1
76      Output: The Familiar IP Addr string
77      '''
78
79      ip = []
80
81      for i in range(4):
82          val = 0
83          # Calc decimal value from binary
84          for j, _ in zip(ip_list[i], range(8)):
85              if j:
86                  val += pow(2, (7 - _))
87          ip.append(str(val))
88
89      return '.'.join(ip)
90
91  def main():
92
93      given_ip = []
94      # s = '205.16.37.24'
95      # s = '172.20.15.1'
96      s = '192.168.1.0'
```

```
    subnet_number = 32


    # Input from user.
    print('Enter the Given IP Address: ')
    # s = input()

    print('Enter the Number of Subnets you Want to make: ')
    # subnet_number = int(input())


    given_ip = [int(i) for i in s.split('.')]
    given_ip_in_bits = [get_dec_in_bin_bits(i, 8) for i in given_ip]

    ip_class = get_ip_class(given_ip)
    const_ip_part = given_ip_in_bits[:int(ip_classes[ip_class][0] / 8)]

    subnet_bits = find_subnet_bits(subnet_number);
    host_bits = ip_classes[ip_class][1] - subnet_bits

    if subnet_bits > ip_classes[ip_class][1]:
        print("Cant Borrow ", subnet_bits, "Number of bits from the host, not enough host bits remaining in class ", ip_class)

    cidr = ip_classes[ip_class][0] + subnet_bits
    possible_subnets = pow(2, subnet_bits)
    possible_hosts = pow(2, host_bits)
    subnet_mask = calc_subnet_mask(cidr)

    # Add things to the Dataframe
    df = pd.DataFrame(columns=['Subnet IP', 'Starting Host IP', 'Last Host IP', 'Broadcast IP', 'Subnet Mask'])

    data = {
        'Subnet IP' : 0,
        'Starting Host IP' : 0,
        'Last Host IP': 0,
        'Broadcast IP' : 0,
        'Subnet Mask' : subnet_mask
    }



    for i in range(possible_subnets):
        # to get the respective ips,we first need the subnet ip
        # there are only possible_subnets possible subnets, so we could just iterate through i

        # Get the subnet part and the host part in bits which will be different each iteratiion of this loop.
        subnet_part = get_dec_in_bin_bits(i, subnet_bits)
        host_part = get_dec_in_bin_bits(1, host_bits)

        # Creating Copies of the Constant part of the IP Address to append to.
        subnet_ip = const_ip_part[:]
        starting_host_ip = const_ip_part[:]
        last_host_ip = const_ip_part[:]
        broadcast_ip = const_ip_part[:]

```

```
        # Appending the Varying part to the constant part for each required arguement.
        subnet_ip.append(subnet_part + get_dec_in_bin_bits(0, host_bits))
        starting_host_ip.append(subnet_part + get_dec_in_bin_bits(1, host_bits))
        last_host_ip.append(subnet_part + get_dec_in_bin_bits(possible_hosts - 2, host_bits))
        broadcast_ip.append(subnet_part + get_dec_in_bin_bits(possible_hosts - 1, host_bits))

        # Adding the values to the dictionary keys.
        data['Subnet IP'] = get_ip(subnet_ip)
        data['Starting Host IP'] = get_ip(starting_host_ip)
        data['Last Host IP'] = get_ip(last_host_ip)
        data['Broadcast IP'] = get_ip(broadcast_ip)

        # Appending Values to the DataFrame
        df_dictionary = pd.DataFrame([data])
        df = pd.concat([df, df_dictionary], ignore_index=True)

    df.to_csv("output.csv")

    print()
    print("The IP Address you have entered is: ", given_ip)
    print("Subnet Number: ", subnet_number)
    print("Subnet Bits: ", subnet_bits)
    print("CIDR Value: ", cidr)
    print("Maximum Possible Subnets: ", possible_subnets)
    print("Maxixmum Possible Hosts: ", possible_hosts)
    print("Subnet Mask is: ", subnet_mask)

main()
```

Listing 1: Hamming Code.cpp

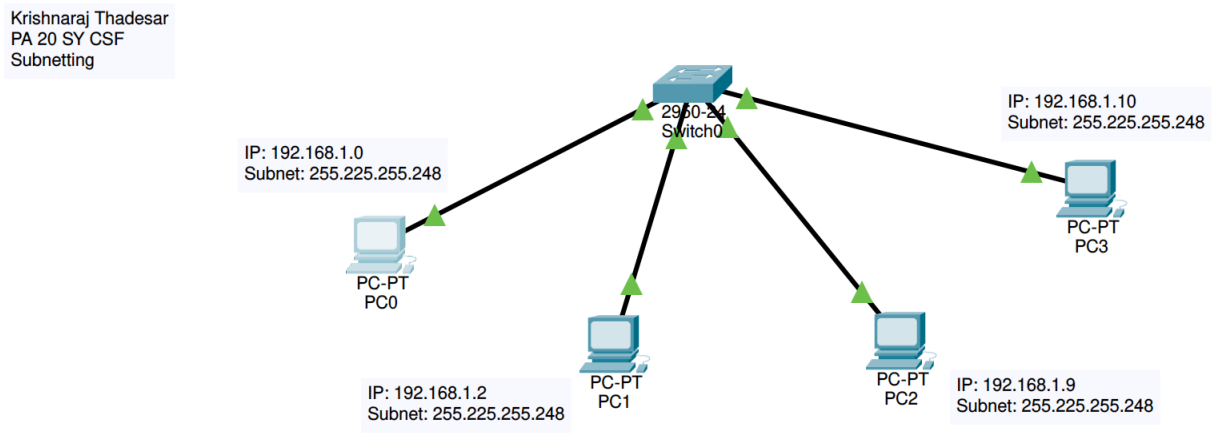
## 4 Output

```
Enter the Given IP Address:
Enter the Number of Subnets you Want to make:

The IP Address you have entered is:  [192, 168, 1, 0]
Subnet Number:  32
Subnet Bits:  5
CIDR Value:  29
Maximum Possible Subnets:  32
Maxixmum Possible Hosts:  8
Subnet Mask is:  255.255.255.248
```

| | Subnet IP | Starting Host IP | Last Host IP | Broadcast IP | Subnet Mask |
|---|---|---|---|---|---|
| 0 | 192.168.1.0 | 192.168.1.1 | 192.168.1.6 | 192.168.1.7 | 255.255.255.248 |
| 1 | 192.168.1.8 | 192.168.1.9 | 192.168.1.14 | 192.168.1.15 | 255.255.255.248 |
| 2 | 192.168.1.16 | 192.168.1.17 | 192.168.1.22 | 192.168.1.23 | 255.255.255.248 |
| 3 | 192.168.1.24 | 192.168.1.25 | 192.168.1.30 | 192.168.1.31 | 255.255.255.248 |
| 4 | 192.168.1.32 | 192.168.1.33 | 192.168.1.38 | 192.168.1.39 | 255.255.255.248 |
| 5 | 192.168.1.40 | 192.168.1.41 | 192.168.1.46 | 192.168.1.47 | 255.255.255.248 |
| 6 | 192.168.1.48 | 192.168.1.49 | 192.168.1.54 | 192.168.1.55 | 255.255.255.248 |
| 7 | 192.168.1.56 | 192.168.1.57 | 192.168.1.62 | 192.168.1.63 | 255.255.255.248 |
| 8 | 192.168.1.64 | 192.168.1.65 | 192.168.1.70 | 192.168.1.71 | 255.255.255.248 |
| 9 | 192.168.1.72 | 192.168.1.73 | 192.168.1.78 | 192.168.1.79 | 255.255.255.248 |
| 10 | 192.168.1.80 | 192.168.1.81 | 192.168.1.86 | 192.168.1.87 | 255.255.255.248 |
| 11 | 192.168.1.88 | 192.168.1.89 | 192.168.1.94 | 192.168.1.95 | 255.255.255.248 |
| 12 | 192.168.1.96 | 192.168.1.97 | 192.168.1.102 | 192.168.1.103 | 255.255.255.248 |
| 13 | 192.168.1.104 | 192.168.1.105 | 192.168.1.110 | 192.168.1.111 | 255.255.255.248 |
| 14 | 192.168.1.112 | 192.168.1.113 | 192.168.1.118 | 192.168.1.119 | 255.255.255.248 |
| 15 | 192.168.1.120 | 192.168.1.121 | 192.168.1.126 | 192.168.1.127 | 255.255.255.248 |
| 16 | 192.168.1.128 | 192.168.1.129 | 192.168.1.134 | 192.168.1.135 | 255.255.255.248 |
| 17 | 192.168.1.136 | 192.168.1.137 | 192.168.1.142 | 192.168.1.143 | 255.255.255.248 |
| 18 | 192.168.1.144 | 192.168.1.145 | 192.168.1.150 | 192.168.1.151 | 255.255.255.248 |
| 19 | 192.168.1.152 | 192.168.1.153 | 192.168.1.158 | 192.168.1.159 | 255.255.255.248 |
| 20 | 192.168.1.160 | 192.168.1.161 | 192.168.1.166 | 192.168.1.167 | 255.255.255.248 |
| 21 | 192.168.1.168 | 192.168.1.169 | 192.168.1.174 | 192.168.1.175 | 255.255.255.248 |
| 22 | 192.168.1.176 | 192.168.1.177 | 192.168.1.182 | 192.168.1.183 | 255.255.255.248 |
| 23 | 192.168.1.184 | 192.168.1.185 | 192.168.1.190 | 192.168.1.191 | 255.255.255.248 |
| 24 | 192.168.1.192 | 192.168.1.193 | 192.168.1.198 | 192.168.1.199 | 255.255.255.248 |
| 25 | 192.168.1.200 | 192.168.1.201 | 192.168.1.206 | 192.168.1.207 | 255.255.255.248 |
| 26 | 192.168.1.208 | 192.168.1.209 | 192.168.1.214 | 192.168.1.215 | 255.255.255.248 |
| 27 | 192.168.1.216 | 192.168.1.217 | 192.168.1.222 | 192.168.1.223 | 255.255.255.248 |
| 28 | 192.168.1.224 | 192.168.1.225 | 192.168.1.230 | 192.168.1.231 | 255.255.255.248 |
| 29 | 192.168.1.232 | 192.168.1.233 | 192.168.1.238 | 192.168.1.239 | 255.255.255.248 |
| 30 | 192.168.1.240 | 192.168.1.241 | 192.168.1.246 | 192.168.1.247 | 255.255.255.248 |
| 31 | 192.168.1.248 | 192.168.1.249 | 192.168.1.254 | 192.168.1.255 | 255.255.255.248 |

# 5 Packet Tracer Implementation




# 6 Conclusion

Thus learnt how Subnetting works, and implemented a simple program using python to calculate the IP Addresses, and subnet masks of each Subnetwork.