**PSG COLLEGE OF TECHNOLOGY**

**DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCES**

**M. Sc Software Systems**

**COMPUTER NETWORKS LAB – IP Subnet Calculator**

**Subnetting**

There comes a time when the network becomes too large to manage and performance numbers hit an all-time low as a result of too much traffic. One of the most effective techniques to solve this network congestion problem is to break the TCP/IP network into smaller, more manageable pieces. The practice of dividing the network is called subnetting, and a tool that can identify these different divisions is called a subnetting calculator.

**Need for a IPv4 subnet calculator**

Every subnet has an address to represent it and these subnets are interconnected via router. The router needs network address and subnet mask to find out whether the incoming IP packet has to be routed to any of its subnetwork. Performing network calculations can be hard at times when you have to consider different parameters as to how many subnets should be there and how many hosts should be in a subnetwork.

The subnet calculator is a handy tool for finding the number of possible subnets for any given network address block. You can choose the combination of subnets and number of hosts per subnet that suits your network and get the host address range and broadcast address for any given subnet mask. Partitioning a large network and allocating IP address ranges to different teams is a task that can be calculated mentally, but it's better to have an option like an IP range calculator or subnet mask calculator to double check your subnet calculations before configuring them in the router.

**Objectives**

**Part 1: Determine IPv4 Address Subnetting**

**Part 2: Calculate IPv4 Address Subnetting**

**Background / Scenario**

The ability to work with IPv4 subnets and determine network and host information based on a given IP address and subnet mask is critical to understanding how IPv4 networks operate. The first part is designed to reinforce how to compute network IP address information from a given IP address and subnet mask. When given an IP address and subnet mask, you will be able to determine other information about the subnet. **You can work for classful addressing with equal length subnetting or for classless addressing with variable length subnetting. The language to be used for implementation is left for your choice. It can be developed as an HTTP web server also as given in the links at the end of the document.**

### Part 1: Determine IPv4 Address Subnetting

In Part 1, you will determine the network and broadcast addresses, as well as the number of hosts, given an IPv4 address and subnet mask.

Determine the network and broadcast addresses and number of host bits and hosts **for the given IPv4 addresses and prefixes** as shown in the following table.

**Sample Output**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **IPv4 Address/Prefix** | **Network Address** | **Broadcast Address** | **Total Number of Host Bits** | **Total Number of Hosts** |
| 192.168.100.25/28 | 192.168.100.16 | 192.168.100.31 | 4 | 14 |
| 172.30.10.130/30 | 172.30.10.128 | 172.30.10.131 | 2 | 2 |
| 10.1.113.75/19 | 10.1.96.0 | 10.1.127.255 | 13 | 8190 |
| 198.133.219.250/24 | 198.133.219.0 | 198.133.219.255 | 8 | 254 |
| 128.107.14.191/22 | 128.107.12.0 | 128.107.15.255 | 10 | 1022 |
| 172.16.104.99/27 | 172.16.104.96 | 172.16.104.127 | 5 | 30 |

**Part 2: Calculate IPv4 Address Subnetting**

When given an IPv4 address, the network mask and the subnet mask, you will be able to determine:

* Network address of this subnet
* Broadcast address of this subnet
* Range of host addresses of this subnet
* Number of subnets created
* Number of hosts per subnet

**Sample Output**

|  |  |
| --- | --- |
| **Given:** | |
| **Host IP Address:** | 172.16.77.120 |
| **Network Mask** | 255.255.0.0 |
| **New Subnet Mask:** | 255.255.240.0 |
| **Find:** | |
| **Number of Subnet Bits** | 4 |
| **Number of Subnets Created** | 16 |
| **Number of Host Bits per Subnet** | 12 |
| **Number of Hosts per Subnet** | 4,094 |
| **Network Address of this Subnet** | 172.16.64.0 |
| **IPv4 Address of First Host on this Subnet** | 172.16.64.1 |
| **IPv4 Address of Last Host on this Subnet** | 172.16.79.254 |
| **IPv4 Broadcast Address on this Subnet** | 172.16.79.255 |

**Reference Links:**

<https://www.calculator.net/ip-subnet-calculator.html>

<https://www.site24x7.com/tools/ipv4-subnetcalculator.html>

<https://www.subnet-calculator.com/cidr.php>

<https://www.subnet-calculator.com/>

<https://jodies.de/ipcalc>

import random

import sys

def subnet\_calc():

try:

while True:

# Take IP as input

input\_ip = input("\nEnter the IP address: ")

# Validate the IP

octet\_ip = input\_ip.split(".")

#print octet\_ip

int\_octet\_ip = [int(i) for i in octet\_ip]

if (len(int\_octet\_ip) == 4) and \

(int\_octet\_ip[0] != 127) and \

(int\_octet\_ip[0] != 169) and \

(0 <= int\_octet\_ip[1] <= 255) and \

(0 <= int\_octet\_ip[2] <=255) and \

(0 <= int\_octet\_ip[3] <= 255):

break

else:

print ("Invalid IP, retry \n")

continue

# Predefine possible subnet masks

masks = [0, 128, 192, 224, 240, 248, 252, 254, 255]

while True:

# Take subnet mask as input

input\_subnet = input("\nEnter the Subnet Mask: ")

# Validate the subnet mask

octet\_subnet = [int(j) for j in input\_subnet.split(".")]

# print octet\_subnet

if (len(octet\_subnet) == 4) and \

(octet\_subnet[0] == 255) and \

(octet\_subnet[1] in masks) and \

(octet\_subnet[2] in masks) and \

(octet\_subnet[3] in masks) and \

(octet\_subnet[0] >= octet\_subnet[1] >= octet\_subnet[2] >= octet\_subnet[3]):

break

else:

print ("Invalid subnet mask, retry\n")

continue

# Converting IP and subnet to binary

ip\_in\_binary = []

# Convert each IP octet to binary

ip\_in\_bin\_octets = [bin(i).split("b")[1] for i in int\_octet\_ip]

# make each binary octet of 8 bit length by padding zeros

for i in range(0,len(ip\_in\_bin\_octets)):

if len(ip\_in\_bin\_octets[i]) < 8:

padded\_bin = ip\_in\_bin\_octets[i].zfill(8)

ip\_in\_binary.append(padded\_bin)

else:

ip\_in\_binary.append(ip\_in\_bin\_octets[i])

# join the binary octets

ip\_bin\_mask = "".join(ip\_in\_binary)

# print ip\_bin\_mask

sub\_in\_bin = []

# convert each subnet octet to binary

sub\_bin\_octet = [bin(i).split("b")[1] for i in octet\_subnet]

# make each binary octet of 8 bit length by padding zeros

for i in sub\_bin\_octet:

if len(i) < 8:

sub\_padded = i.zfill(8)

sub\_in\_bin.append(sub\_padded)

else:

sub\_in\_bin.append(i)

# print sub\_in\_bin

sub\_bin\_mask = "".join(sub\_in\_bin)

# calculating number of hosts

no\_zeros = sub\_bin\_mask.count("0")

no\_ones = 32 - no\_zeros

no\_hosts = abs(2 \*\* no\_zeros - 2)

# Calculating wildcard mask

wild\_mask = []

for i in octet\_subnet:

wild\_bit = 255 - i

wild\_mask.append(wild\_bit)

wildcard = ".".join([str(i) for i in wild\_mask])

# Calculating the network and broadcast address

network\_add\_bin = ip\_bin\_mask[:no\_ones] + "0" \* no\_zeros

broadcast\_add\_bin = ip\_bin\_mask[:no\_ones] + "1" \* no\_zeros

network\_add\_bin\_octet = []

broadcast\_binoct = []

[network\_add\_bin\_octet.append(i) for i in [network\_add\_bin[j:j+8]

for j in range(0, len(network\_add\_bin), 8)]]

[broadcast\_binoct.append(i) for i in [broadcast\_add\_bin[j:j+8]

for j in range(0,len(broadcast\_add\_bin),8)]]

network\_add\_dec\_final = ".".join([str(int(i,2)) for i in network\_add\_bin\_octet])

broadcast\_add\_dec\_final = ".".join([str(int(i,2)) for i in broadcast\_binoct])

# Calculate the host IP range

first\_ip\_host = network\_add\_bin\_octet[0:3] + [(bin(int(network\_add\_bin\_octet[3],2)+1).split("b")[1].zfill(8))]

first\_ip = ".".join([str(int(i,2)) for i in first\_ip\_host])

last\_ip\_host = broadcast\_binoct[0:3] + [bin(int(broadcast\_binoct[3],2) - 1).split("b")[1].zfill(8)]

last\_ip = ".".join([str(int(i,2)) for i in last\_ip\_host])

# print all the computed results

print ("\nIP address : " + input\_ip)

print( "subnet mask: " + input\_subnet)

print ("number of hosts per subnet: {0}".format(str(no\_hosts)))

print( "number of mask bits: {0}".format(str(no\_ones)))

print( "The Network address: {0}".format(network\_add\_dec\_final))

print( "The Broadcast address: {0}".format(broadcast\_add\_dec\_final))

print ("IP address range: {0} - {1}".format(first\_ip, last\_ip))

print ("Maximum number of subnets: " + str(2\*\*abs(24 - no\_ones)))

list\_ip = []

print ("")

# ask to generate a random ip in the range

if input("Do you want to generate a random ip? [y/n]") == 'y':

while True:

randip = []

# Check if the octet bit is same in first and last host address.

# If same, append it. else generate random IP

for i in range(0,len(first\_ip\_host)):

for j in range(0,len(last\_ip\_host)):

if i == j:

if first\_ip\_host[i] == last\_ip\_host[j]:

randip.append(int(first\_ip\_host[i],2))

else:

randip.append(random.randint(int(first\_ip\_host[i],2),int(last\_ip\_host[j],2)))

random\_ip\_final = ".".join(str(i) for i in randip)

# check if generated IP has already been printed. If so, compute again till unique IP is obtained

if random\_ip\_final in list\_ip:

# if all IPs in the host range are used, exit

if len(list\_ip) == no\_hosts:

print ("All IPs in the range used up, exiting\n")

break

continue

else:

print (random\_ip\_final + '\n')

list\_ip.append(random\_ip\_final)

print ("List of generated IPs:" , sorted(list\_ip) ,'\n')

if input("\nGenerate another random IP? [y/n]") == 'y':

continue

else:

break

except KeyboardInterrupt:

print ("Interrupted by the User, exiting\n")

except ValueError:

print ("Seem to have entered an incorrect value, exiting\n")

# Calling the above defined function

if \_\_name\_\_ == '\_\_main\_\_':

subnet\_calc()