**Instituto Politécnico Nacional**

Escuela Superior de Cómputo

**Redes De Computadoras**

**Reporte De Practica**

**“Calculadora IP”**

**Profesor:**

Juan Jesús Alcaraz Torres

**Alumnos:**

Torres Abonce Luis Miguel

Olguin Castillo Brayan Mauricio

**Grupo:**

5SCM4

**Fecha de entrega**

22/12/22

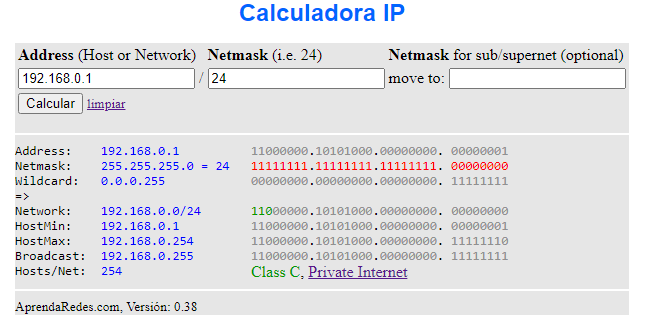
Introducción.

Una calculadora IP es una herramienta útil para trabajar con direcciones IP y netmask en redes de computadoras. La dirección IP es un número único asignado a cada dispositivo conectado a una red de computadoras, y la netmask es un conjunto de números que se utiliza para dividir la red en subredes.

La calculadora IP realiza diversas operaciones con estos valores, como calcular el host mínimo y máximo de una subred, el broadcast de una red o el número de subredes que se pueden crear al agregar otra netmask.

Para utilizar la calculadora IP, se debe ingresar la dirección IP y la netmask de la red y opcionalmente otra netmask para calcular subredes. Entonces la calculadora realizara las operaciones necesarias para mostrar el resultado.

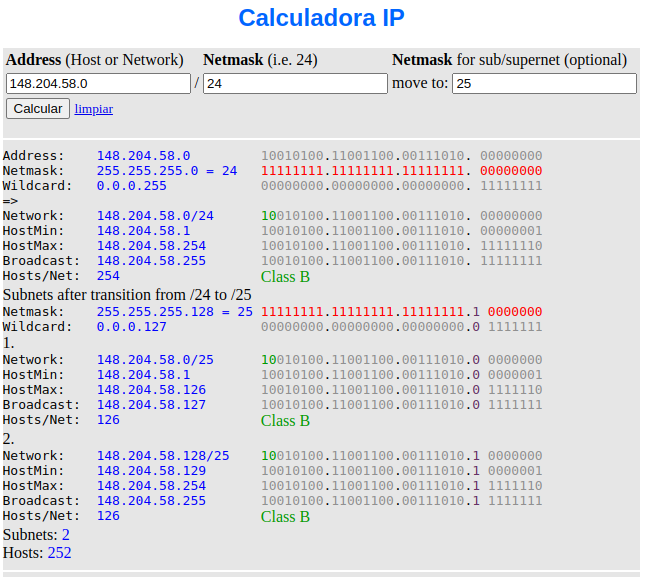
Se utilizo el lenguaje de programación de Python para el desarrollo del programa.



Desarrollo.

Para el desarrollo del programa se utilizó el lenguaje de programación de Python debido a que este es bastante sencillo de comprender y programar, además contamos con la suficiente experiencia, esto gracias a que en otras materias lo utilizamos de manera habitual.

Utilizamos la librería tkinter para realizar una interfaz gráfica y a través de esta pedir al usuario los datos (dirección IP, netmask1, netmask2(opcional)), así como proporcionar los resultados en una ventana aparte.

Para la realización nos basamos en la página web proporcionada por el profesor y fue una guía bastante útil, esto para darnos una idea de cómo mostrar los datos calculados y los dados por el usuario. Nuestra calculadora para mostrar los datos lo hacemos en cuadro de texto donde si las subredes son demasiadas se podrá usar la barra de desplazamiento para observar las demás subredes que no se alcanzan a observar en la pantalla.

Para el cálculo de la primera parte (Network, HostMin, HostMax, Broadcast, Host/Net) primeramente colocamos los datos como la address, calculamos la netmask y convertimos tanto decimal como a binario, de igual forma lo hacemos con la wildcard que será de mucha ayuda, después para dar valor a Network este depende del valor de la netmask y la address por ejemplo si la netmask es 10 se tomarán los primeros 10 valores de la address y los demás se convertirán en 0’s

Para calcular el HostMin se toma el valor de network y en su ultimo octeto se le suma 1. Para el HostMax se le resta 1 en su último octeto.

Para el Broadcast se toma el valor del Network y se le suma el valor de la wildcard.

Después de calcular todos estos valores se convierten en binario o decimal según sea el caso. Por ultima para esta primera parte se calcula el No de hosts y el tipo de clase de la red para esto se utiliza la formula donde n representa el número de bits que no se utilizan en la red.

Para determinar la clase de la red es a partir del número del primer octeto.

Posteriormente se calculan las subredes si es que el usuario introdujo la segunda netmask y esta es mayor a la primera.

Para esto necesitamos 2 datos la netmask2 y la wildcard la cual básicamente es la netmask invertida (donde hay un 1 se pone y 0 de igual forma con el 0) se convierten estos datos en binarios y decimales para mostrarlos en pantalla.

Debemos de saber cuántas subredes se pueden calcular así que esto se puede saber restando la netmask2 – netmask1 y a 2 elevamos el resultado (); teniendo el numero de subredes podemos empezar a calcular cada una de ellas, la diferencia de bits entre cada subred será igual a la wildcard por ejemplo si tenemos que address = 148.204.58.0 y netmask1=24 y netmask2=27, la wildcard=0.0.0.31 ya que subredes.

Esto es un ciclo que no terminara hasta que tengamos todas las subredes, para la primera subred se toma el network subred = network de la red, se le suma y resta 1 en el ultimo octeto para el host mínimo y host máximo, para el broadcast a network de la subred se le suma la wildcard. Para la siguiente subred se toma el broadcast de la anterior y se le suma 1 en su ultimo octeto para obtener el network de la subred y se repite el anterior proceso para calcular los demás datos, y así sucesivamente con todas las subredes hasta terminar con cada una de ellas.

De esta forma fue como realizamos nuestra calculadora IP.

Código de la calculadora IP.

from tkinter import \*

from tkinter import messagebox

import re

# Crear ventana

calculadora = Tk()

calculadora.title("CalculadoraIP")

calculadora.geometry("300x200")

calculadora.configure(background="white")

def limpiar*()*:#limpiar los campos

    E1.delete(0,END)

    E2.delete(0,END)

    E3.delete(0,END)

def calcular*()*:# Obtener valores de los Entry

    netmask = E2.get()

    address = E1.get()

    move\_to = E3.get()

    # Validar que los valores no esten vacios

    try:

        if netmask == "" or address == "":#verificar que los campos no esten vacios

            messagebox.showerror("Error", "Debe llenar todos los campos")

        regex = "^((25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9]?[0-9])\.){3}(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9]?[0-9])$"

        if not re.match(regex, address):#verificar si la direccion ip es valida

            messagebox.showerror("Error", "La direccion IP no es valida")

            return

        if not int(netmask) in range(1, 33):#verificar si la mascara es valida

            messagebox.showerror("Error", "La mascara de red no es valida")

            return

        if move\_to == "":#verificar que el campo no este vacio

            move\_to= "no"

        elif not int(move\_to) in range(1, 33):#verificar si move\_to es valido

            messagebox.showerror("Error", "El valor de move\_to no es valido")

            return

    except:

        return

    #convertir a enteros los valores

    netmask = int(netmask)

    if move\_to != "no":

        move\_to = int(move\_to)

    #Verificar la clase de la direccion ip

    if int(address.split(".")[0]) in range(1, 128):

        clase = "A"

    elif int(address.split(".")[0]) in range(128, 192):

        clase = "B"

    elif int(address.split(".")[0]) in range(192, 224):

        clase = "C"

    elif int(address.split(".")[0]) in range(224, 240):

        clase = "D"

    elif int(address.split(".")[0]) in range(240, 256):

        clase = "E"

    #obtener la direccion de red binaria

    address\_bin = ""

    j=0

    for i in address.split("."):

        address\_bin += bin(int(i))[2:].zfill(8)

        if (j!=3):

            address\_bin += "."

        j+=1

    #obtener la mascara de red binaria

    netmask\_bin = ""#declarar mascara binaria vacia

    for i in range(0, netmask):

        netmask\_bin = netmask\_bin + "1"

        if i == 7 or i == 15 or i == 23:

            if i == netmask - 1:

                break

            netmask\_bin = netmask\_bin + "."#agregar un punto cada 8 bits

    while(i!=31):

        if i == 7 or i == 15 or i == 23:

            if i != 31:

                netmask\_bin = netmask\_bin + "."

            else:

                break

        netmask\_bin = netmask\_bin + "0"

        i = i + 1

    #obtener la wildcard

    wildcard = ""

    i=0

    for i in netmask\_bin.split("."):

        wildcard += str(255 - int(i, 2))

        wildcard += "."

    wildcard = wildcard[:-1]

    #obtener la wildcard binaria

    wildcard\_binaria = ""

    for i in wildcard.split("."):

        wildcard\_binaria += bin(int(i))[2:].zfill(8)

        wildcard\_binaria += "."

    wildcard\_binaria = wildcard\_binaria[:-1]

    #obtener network\_bin

    i=netmask

    network\_bin=address\_bin[:i+2]

    i+=2

    while i!=len(address\_bin):

        if address\_bin[i]!='.':

            network\_bin+='0'

        else:

            network\_bin=network\_bin+'.'

        i+=1

    #obtener network

    network = ""

    i=0

    j=0

    for i in network\_bin.split("."):

        network += str(int(i, 2))

        if(j!=3):

            network += "."

        j+=1

    NoHosts = 2\*\*(32 - netmask)-2

    HostMin\_bin=network\_bin[:-1]

    HostMin\_bin+='1'

    #obtener la host minima

    HostMin = ""

    i=0

    for i in HostMin\_bin.split("."):

        HostMin += str(int(i, 2))

        HostMin += "."

    HostMin = HostMin[:-1]

    #obtener HostMax\_bin

    j=0

    if netmask > 8:  j+=1

    if netmask > 16: j+=1

    if netmask > 24: j+=1

    HostMax\_bin=network\_bin[:j+netmask]

    i=j+netmask

    while i!=35:

        if network\_bin[i]!='.':

            HostMax\_bin+='1'

        else:

            HostMax\_bin=HostMax\_bin+'.'

        i+=1

    HostMax\_bin=HostMax\_bin[:-1]

    HostMax\_bin+='0'

    #obtener la host maxima

    HostMax = ""

    i=0

    for i in HostMax\_bin.split("."):

        HostMax += str(int(i, 2))

        HostMax += "."

    HostMax = HostMax[:-1]

    #obtener broadcast\_bin

    broadcast\_bin= HostMax\_bin[:-1]

    broadcast\_bin+='1'

    #obtener broadcast

    broadcast = ""

    i=0

    for i in broadcast\_bin.split("."):

        broadcast += str(int(i, 2))

        broadcast += "."

    broadcast = broadcast[:-1]

    resultado = Tk()

    resultado.title("Resultado")

    resultado.geometry("700x500")

    #crear la scrollbar

    label=Label(resultado)

    texto=Text(label)

    texto.grid(row=0,column=0,sticky=W)

    scrollbar=Scrollbar(label, orient=VERTICAL, command=texto.yview)

    texto.config(yscrollcommand=scrollbar.set)

    scrollbar.grid(row=0,column=1,sticky=W)

    label.grid(row=0,column=0,sticky=W)

    #resultado.configure(background="white")

    texto.insert(END,"Address:      " + address +"          "+ address\_bin+"\n")

    texto.insert(END,"Netmask:      " + str(netmask) +"                       "+ netmask\_bin+"\n")

    texto.insert(END,"Wildcard:     " + wildcard +"             "+ wildcard\_binaria+"\n")

    texto.insert(END,"--------------------------------------------------------------------------"+"\n")

    texto.insert(END,"Network:      " + network +"               "+ network\_bin+"\n")

    texto.insert(END,"HostMin:      " + HostMin +"               "+ HostMin\_bin+"\n")

    texto.insert(END,"HostMax:      " + HostMax +"          "+ HostMax\_bin+"\n")

    texto.insert(END,"Broadcast:    " + broadcast +"          "+ broadcast\_bin+"\n")

    texto.insert(END,"Hosts/net:    " + str(NoHosts)+"\n")

    texto.insert(END,"Clase:        " + clase+"\n")

    texto.insert(END,"--------------------------------------------------------------------------"+"\n")

    if move\_to != "no":

        Netmask2 = int(move\_to)

        #obtener la mascara de red binaria

        netmask2\_bin = ""#declarar mascara binaria vacia

        for i in range(0, Netmask2):

            netmask2\_bin = netmask2\_bin + "1"

            if i == 7 or i == 15 or i == 23:

                if i == Netmask2 - 1:

                    break

                netmask2\_bin = netmask2\_bin + "."#agregar un punto cada 8 bits

        while(i!=31):

            if i == 7 or i == 15 or i == 23:

                if i != 31:

                    netmask2\_bin = netmask2\_bin + "."

                else:

                    break

            netmask2\_bin = netmask2\_bin + "0"

            i = i + 1

        #obtener la wildcard2

        wildcard2 = ""

        i=0

        for i in netmask2\_bin.split("."):

            wildcard2 += str(255 - int(i, 2))

            wildcard2 += "."

        wildcard2 = wildcard2[:-1]

        #obtener la wildcard binaria

        wildcard\_binaria2 = ""

        for i in wildcard2.split("."):

            wildcard\_binaria2 += bin(int(i))[2:].zfill(8)

            wildcard\_binaria2 += "."

        wildcard\_binaria2 = wildcard\_binaria2[:-1]

        #obtener las subnets

        subnet = Netmask2 - netmask

        subnet\_bin = ""

        for i in range(0, subnet):

            subnet\_bin += "1"

        subnet\_dec = int(subnet\_bin,2)

        texto.insert(END,"Netmask2:     " + str(Netmask2) +"                    "+ netmask2\_bin+"\n")

        texto.insert(END,"Wildcard2:    " + wildcard2 +"           "+ wildcard\_binaria2+"\n"+"\n")

        i=1

        suma=[]

        string\_suma=""

        acarreo=0

        binario=network.split(".")

        wildcard2=wildcard2.split(".")

        print(binario)

        print(wildcard2)

        string\_suma=network

        string\_hostmin2=HostMin

        for i in range(0, subnet\_dec+1):

            texto.insert(END,"Subnet "+str(i+1)+":"+"\n")

            texto.insert(END,"Network:      " + string\_suma +"          "+"\n")

            texto.insert(END,"HostMin:      " + string\_hostmin2 +"          "+"\n")

            #calcular Host Min

            suma.clear()

            string\_suma=""

            for i in range(3,-1,-1):

                if wildcard2[i]=='255':

                    suma.append(255)

                elif int(binario[i])+int(wildcard2[i])-256>=0:

                    suma.append(int(binario[i])+int(wildcard2[i])-256)

                    acarreo=1

                else:

                    if acarreo==1:

                        suma.append(int(binario[i])+int(wildcard2[i])+1)

                        acarreo=0

                    else:

                        suma.append(int(binario[i])+int(wildcard2[i]))

            #calcular Host Max

            suma.reverse()

            for i in suma:

                string\_suma+=str(i)+"."

            string\_suma=string\_suma[:-1]

            #host maximo

            i=0

            binario=string\_suma.split(".")

            string\_hostmax2=""

            for i in range(3,-1,-1):

                if binario[i]!='255':

                    binario[i]=(int(binario[i])-1)

                    break

                else:

                    binario[i]=254

            for i in binario:

                string\_hostmax2+=str(i)+"."

            string\_hostmax2=string\_hostmax2[:-1]

            texto.insert(END,"HostMax:      " + string\_hostmax2 +"          "+"\n")

            texto.insert(END,"Broadcast:    " + string\_suma+"\n")

            #calcular Host min

            i=0

            binario=string\_suma.split(".")

            suma.clear()

            string\_suma=""

            for i in range(3,-1,-1):

                if binario[i]!='255':

                    binario[i]=(int(binario[i])+1)

                    break

                else:

                    binario[i]=0

            for i in binario:

                string\_suma+=str(i)+"."

            string\_suma=string\_suma[:-1]

            #host minimo

            i=0

            binario=string\_suma.split(".")

            string\_hostmin2=""

            for i in range(3,-1,-1):

                if binario[i]!='255':

                    binario[i]=(int(binario[i])+1)

                    break

                else:

                    binario[i]=0

            for i in binario:

                string\_hostmin2+=str(i)+"."

            string\_hostmin2=string\_hostmin2[:-1]

            binario=string\_suma.split(".")

    texto.configure(state="disabled")

#crear label y entry de address

L1=Label(calculadora, text="Address(Host or Network)",bg="white").grid(row=0,column=0,sticky=W)

E1=Entry(calculadora, bd =1, bg="white")

E1.grid(row=0,column=1,sticky=W)

#crear label y entry de netmask

L2=Label(calculadora, text="Netmask(i.e. 24)",bg="white").grid(row=1,column=0,sticky=W)

E2=Entry(calculadora, bd =1,bg="white")

E2.grid(row=1,column=1,sticky=W)

#crear label y entry de move\_to

L3=Label(calculadora, text="Move to:",bg="white").grid(row=2,column=0,sticky=W)

E3=Entry(calculadora, bd =1, bg="white")

E3.grid(row=2,column=1,sticky=W)

# Crear botones

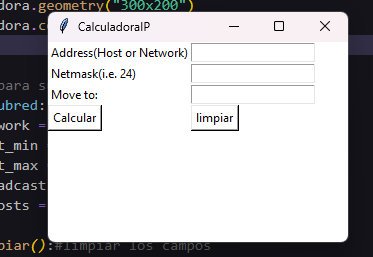
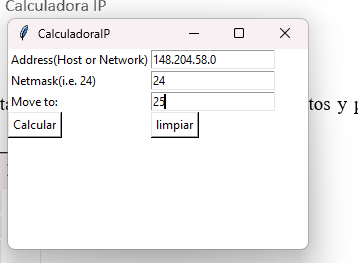
B1=Button(calculadora, text="Calcular",command=calcular,bg="white").grid(row=3,column=0,sticky=W)

B2=Button(calculadora, text="limpiar",command=limpiar,bg="white").grid(row=3,column=1,sticky=W)

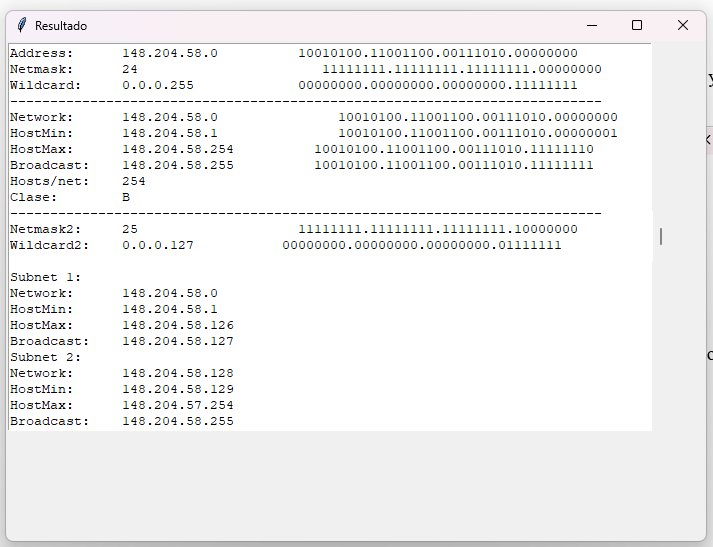
calculadora.mainloop()

Funcionamiento de la calculadora.

Al ejecutar el programa se nos abrirá esta pestaña donde introduciremos los datos y podremos calcular o limpiar los campos.



Una vez introducimos todos lo datos y presionamos el botón de calcular se nos abrirá otra ventana con todos los datos que se calcularon.



Como se puede observar salen todos los datos que se piden, en caso de que las subredes sean demasiadas solo se bajara con el mouse en el cuadro de texto y de esta manera se podrá observar todas las subredes.

Finalmente, solo se cierran las ventas, o si se desea se puede limpiar e introducir nuevos datos.

Conclusiones.

Al desarrollar la practica no tuvimos muchos problemas ya que es un poco sencilla algunos de los pequeños problemas que tuvimos fue con el cálculo de las subredes ya que de primera instancia no recordábamos como se calculaban pero basto con revisar nuestros apuntes de clases para recordarlo, otro problema fue como mostraríamos los datos en la pantalla ya que la librería de Python tkinter tiene diferentes formas de mostrar texto como la función grid que divide la pantalla en filas y columnas, pero esto no nos ayudaba para mostrar los datos de la manera que queríamos por lo que investigando se puede introducir información en un cuadro de texto y simplemente bloqueamos el cuadro de texto para que el usuario no pueda introducir información en el resultado. Esta calculadora es una herramienta valiosa para entender y administrar las redes de computadoras. Ayuda a los profesionales de TI a determinar la configuración de una red y asegurar que se utilicen de manera eficiente las direcciones IP disponibles.

Bibliografías

*Calculadora IP*. (2020, 4 julio). AprendaRedes. https://www.aprendaredes.com/blog/calculadora-ip/