Universidad Nacional Autónoma de México
Facultad de Ciencias

Asignatura: Redes de computadoras
Semestre: 2024-1

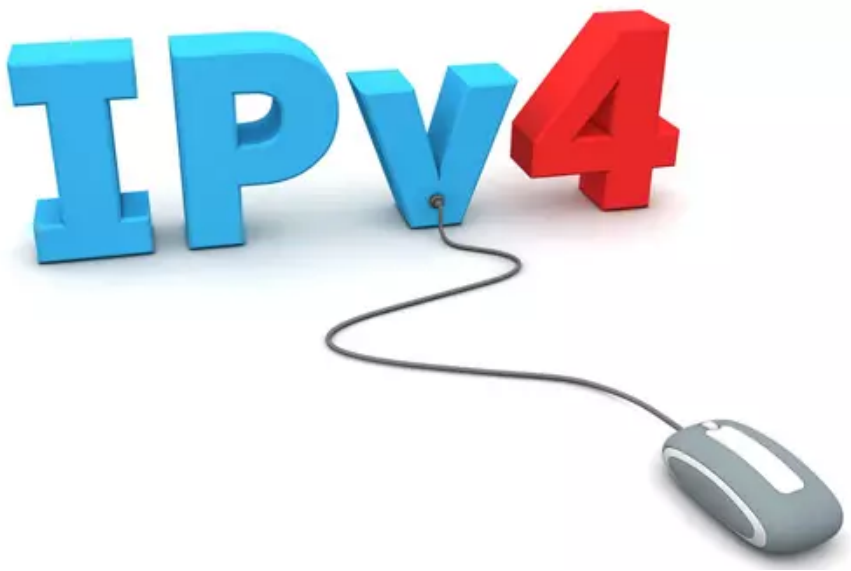
Profesor: Javier León Cotonieto

Ayudantes: Magdalena Reyes Granados
Itzel Gómez Muñoz
Sandra Plata Velázquez

Práctica 7. "Dirección IPv4"

Equipo 5
Integrantes:

- **Almanza Torres José Luis**
- **Jimenez Reyes Abraham**
- **Martínez Pardo Esaú**



Ejercicio1.

Programar una calculadora VLSM.

✓ Lenguaje de programación Python.
✓ Pedir IP (segmento de red)
✓ Máscara o con prefijo
✓ Cantidad de subredes con nombre y host

Resultado: Mostrar una tabla con los siguientes datos.

→ Subred (A,B,C,D...1,2,3,4)
→ Id de Red (X.X.X.X)
→ Máscara o prefijo
→ Rango de direcciones útiles
→ Broadcast

Opcional:

✓ Que cuenta con una interfaz
✓ Menú con otro método (subnetting o CIDR)

Ejemplo 1 como el que hicimos en clase :
Dirección IP: 192.168.1.0/24
Red D 80
Red C 50
Red A 25
Red B 10

```
atjl_jra_mpe@equipo5:~/Descargas$ python3 P7_Equipo5.py
*******************************************************
**************Calculadora VLSM  EQUIPO 5***************
*******************************************************

Escribe la dirección IP: 192.168.1.0

Escribe el tamaño del prefijo para la dirección IP: 24

Escribe el número de redes que necesitas: 4

Escribe el nombre de la red: D

Escribe el número de host de la red D: 80

Escribe el nombre de la red: C

Escribe el número de host de la red C: 50

Escribe el nombre de la red: A

Escribe el número de host de la red A: 25

Escribe el nombre de la red: B

Escribe el número de host de la red B: 10

Subred  Id de red           Prefijo            Rango útil Primera IP   Rango útil Última IP    Broadcast
D       192.168.1.0         25                 192.168.1.1             192.168.1.126           192.168.1.127
C       192.168.1.128       26                 192.168.1.129           192.168.1.190           192.168.1.191
A       192.168.1.192       27                 192.168.1.193           192.168.1.222           192.168.1.223
B       192.168.1.224       28                 192.168.1.225           192.168.1.238           192.168.1.239
```

Ejemplo 2:
Dirección IP: 192.168.1.0/24
Red D 90
Red C 60
Red A 15
Red B 10

```
atjl_jra_mpe@equipo5:~/Descargas$ python3 P7_Equipo5.py
**********************************************************
**************Calculadora VLSM  EQUIPO 5****************
**********************************************************

Escribe la dirección IP: 192.168.1.0

Escribe el tamaño del prefijo para la dirección IP: 24

Escribe el número de redes que necesitas: 4

Escribe el nombre de la red: D

Escribe el número de host de la red D: 90

Escribe el nombre de la red: C

Escribe el número de host de la red C: 60

Escribe el nombre de la red: A

Escribe el número de host de la red A : 15

Escribe el nombre de la red: B

Escribe el número de host de la red B: 10

Subred  Id de red           Prefijo            Rango útil Primera IP   Rango útil Última IP    Broadcast
D       192.168.1.0         25                 192.168.1.1             192.168.1.126           192.168.1.127
C       192.168.1.128       26                 192.168.1.129           192.168.1.190           192.168.1.191
A       192.168.1.192       28                 192.168.1.193           192.168.1.206           192.168.1.207
B       192.168.1.208       28                 192.168.1.209           192.168.1.222           192.168.1.223
atjl_jra_mpe@equipo5:~/Descargas$
```

Ejemplo 3:
num host mayor a 64
Dirección IP: 192.168.1.0/24
Red D 100
Red C 80
Red A 70
Red B 64
FALLA EL ALGORITMO PORQUE ES MAYOR A 64 LOS NÚMEROS DE HOST

```
atjl_jra_mpe@equipo5:~/Descargas$ python3 P7_Equipo5.py
**********************************************************
**************Calculadora VLSM  EQUIPO 5****************
**********************************************************

Escribe la dirección IP: 192.168.1.0

Escribe el tamaño del prefijo para la dirección IP: 24

Escribe el número de redes que necesitas: 4

Escribe el nombre de la red: D

Escribe el número de host de la red D: 100

Escribe el nombre de la red: C

Escribe el número de host de la red C: 80

Escribe el nombre de la red: A

Escribe el número de host de la red A: 70

Escribe el nombre de la red: B

Escribe el número de host de la red B: 64

El algoritmo falló debido porque el número de host no coincide con la dirección IP
```

Código:

```python
# Ejecutar con python3 P7_Equipo5.py
# Equipo 5: Almanza Torres José Luis
#           Jimenez Reyes Abraham
#           Martínez Pardo Esaú

# Función para obtener la dirección IP del usuario
def obtener_direccion_ip():
    ip = input("\nEscribe la dirección IP: ")
    octetos = ip.split(".")

    # Validar cada octeto de la dirección IP
    for i, octeto in enumerate(octetos):
        try:
            octetos[i] = int(octeto)
            if not (0 <= octetos[i] <= 255):
                raise ValueError
        except ValueError:
            print("\nLa dirección IP está mal escrita")
            exit()

    return octetos

# Función para obtener el prefijo de la dirección IP
def obtener_prefijo():
    try:
        prefijo = int(input("\nEscribe el tamaño del prefijo para la dirección IP: "))
        if not (8 <= prefijo <= 32):
            raise ValueError
    except ValueError:
        print("\nEl prefijo escrito no es válido.")
        exit()

    return prefijo

# Función para obtener el número de redes que el usuario necesita
def obtener_numero_de_redes():
    try:
        num_redes = int(input("\nEscribe el número de redes que necesitas: "))
    except ValueError:
        print("\nEl número de redes no es un número")
        exit()

    return num_redes

# Función para obtener información sobre las redes del usuario
def obtener_informacion_redes(num_redes):
    info_redes = []

```

```python
def obtener_informacion_redes(num_redes):
    info_redes = []

    for i in range(num_redes):
        nombre_red = input("\nEscribe el nombre de la red: ")
        try:
            num_host = int(input(f"\nEscribe el número de host de la red {nombre_red}: "))
        except ValueError:
            print("\nEl número de host no es un número")
            exit()

        info_redes.append((nombre_red, num_host))

    return info_redes

# Función principal para calcular VLSM
def calcular_vlsm(octetos, info_redes):
    # Ordenar la información de las redes de acuerdo al número de hosts en orden descendente
    info_redes_ordenadas = sorted(info_redes, key=lambda x: x[1], reverse=True)
    resultado = []
    contador = 0

    # Calcular parámetros para cada red
    for nombre_red, num_host in info_redes_ordenadas:
        ip_red, prefijo_red, host_util, primera_ip, segunda_ip, broadcast, fail = calcular_parametros_red(octetos
        resultado.append((nombre_red, ip_red, prefijo_red, host_util, primera_ip, segunda_ip, broadcast))

        # Verificar si hay algún fallo y mostrar mensaje en caso de ser el primero
        if fail and contador == 0:
            print("\nEl algoritmo falló debido porque el número de host no coincide con la dirección IP")
            contador += 1

    return resultado

# Función para calcular los parámetros de una red específica
def calcular_parametros_red(octetos, num_host):
    n = calcular_valor_n(num_host)
    host_util = pow(2, n) - 2
    prefijo_red = 32 - n
    numero_magico = pow(2, n)
    fail = False

    ip_red = ".".join(map(str, octetos))
    primera_ip = ".".join(map(str, octetos[:3] + [octetos[3] + 1]))
    segunda_ip = ".".join(map(str, octetos[:3] + [octetos[3] + numero_magico - 2]))
    broadcast = ".".join(map(str, octetos[:3] + [octetos[3] + numero_magico - 1]))

    octetos[3] += numero_magico
    if octetos[3] >= 255:
        octetos[2] += 1
        octetos[3] = 0
        if not fail:
            fail = True
```

```
        octetos[3] = 0
        if not fail:
            fail = True

    return ip_red, prefijo_red, host_util, primera_ip, segunda_ip, broadcast, fail

# Función para calcular el valor de n
def calcular_valor_n(num_host):
    n = 1
    while pow(2, n) < num_host:
        n += 1
    return n

# Función para verificar si el prefijo coincide con la clase de la dirección IP
def verificar_prefijo(prefijo, octetos):
    oct_1 = octetos[0]
    if oct_1 >= 0 and oct_1 < 128:
        if not (8 <= prefijo <= 32):
            print("\nEl prefijo no coincide con la clase de la dirección IP.")
            exit()
    elif oct_1 >= 128 and oct_1 < 192:
        if not (16 <= prefijo <= 32):
            print("\nEl prefijo no coincide con la clase de la dirección IP.")
            exit()
    elif oct_1 >= 192 and oct_1 < 224:
        if not (24 <= prefijo <= 32):
            print("\nEl prefijo no coincide con la clase de la dirección IP.")
            exit()
    else:
        print("\nLa IP ingresada no es válida.")

# Función para imprimir la tabla resultante
def imprimir_tabla_resultante(resultado):
    print("\n{:<7} {:<19} {:<19} {:<23} {:<23} {:<15}".format(
        "Subred", "Id de red", "Prefijo",
        "Rango útil Primera IP", "Rango útil Última IP", "Broadcast"
    ))

    for i in range(len(resultado)):
        print("{:<7} {:<19} {:<19} {:<23} {:<23} {:<15}".format(
            str(resultado[i][0]),
            str(resultado[i][1]),
            str(resultado[i][2]),
            str(resultado[i][4]),
            str(resultado[i][5]),
            str(resultado[i][6])
        ))
```

```
    elif oct_1 >= 128 and oct_1 < 192:
        if not (16 <= prefijo <= 32):
            print("\nEl prefijo no coincide con la clase de la dirección IP.")
            exit()
    elif oct_1 >= 192 and oct_1 < 224:
        if not (24 <= prefijo <= 32):
            print("\nEl prefijo no coincide con la clase de la dirección IP.")
            exit()
    else:
        print("\nLa IP ingresada no es válida.")

# Función para imprimir la tabla resultante
def imprimir_tabla_resultante(resultado):
    print("\n{:<7} {:<19} {:<19} {:<23} {:<23} {:<15}".format(
        "Subred", "Id de red", "Prefijo",
        "Rango útil Primera IP", "Rango útil Última IP", "Broadcast"
    ))

    for i in range(len(resultado)):
        print("{:<7} {:<19} {:<19} {:<23} {:<23} {:<15}".format(
            str(resultado[i][0]),
            str(resultado[i][1]),
            str(resultado[i][2]),
            str(resultado[i][4]),
            str(resultado[i][5]),
            str(resultado[i][6])
        ))

# Función principal
def main():
    print("************************************************************")
    print("***************Calculadora VLSM  EQUIPO 5*****************")
    print("************************************************************")

    octetos = obtener_direccion_ip()
    prefijo = obtener_prefijo()
    verificar_prefijo(prefijo, octetos)
    num_redes = obtener_numero_de_redes()
    info_redes = obtener_informacion_redes(num_redes)
    resultado = calcular_vlsm(octetos, info_redes)
    imprimir_tabla_resultante(resultado)

if __name__ == "__main__":
    main()
```

CONCLUSIONES

José Luis: En esta práctica hablamos sobre las direcciones IP permiten identificar de manera única cada dispositivo en una red, facilitando la comunicación y el intercambio de datos. Sin embargo, con la limitada cantidad de direcciones IPv4 disponibles, se implementan estrategias eficientes como el direccionamiento VLSM, además cabe mencionar que se creó un nuevo sistema llamado IPv6 para la gran demanda que existe.
Además la calculadora VLSM se presenta como una herramienta muy importante en el desarrollo de redes y la capacidad de asignar direcciones IP de manera eficiente considerando las necesidades de cada subred en términos de hosts.

Abraham: VLSM permite un mejor aprovechamiento y optimización del uso de direcciones. El proceso de VLSM toma una dirección de red o subred y la divide en subredes más

pequeñas adaptando las máscaras según las necesidades de hosts de cada subred, generando una máscara diferente para las distintas subredes de una red. Ya pues mi compañero realizó la calculadora y con algunos ejemplos podemos ver su funcionamiento.

Esaú: En esta práctica de direccionamiento IPV4 checamos el método de VLSM (Variable Length Subnet Masking) que vimos en clase e hicimos todos esos cálculos en papel. El método consiste en dividir a una subred en otras subredes y así sucesivamente. Consta de diversos pasos: ordenar las redes respecto a su número de host, calcular una n que me permite descifrar la máscara de subred y saber cuantos bits son host y cuántos para la subred y obtener el número mágico, y así con todas las subredes hasta obtener una tabla resultante de Id de red, Máscara, primer y último rango útil de dirección IP y Broadcast, todo esto trasladado ahora a una calculadora en Python para facilitar todos estos cálculos.

REFERENCIAS

- Coto, A. (--). Introducción a redes | Ministerio de Agricultura y Desarrollo Rural. Recuperado el 12 de octubre, de https://www.minagricultura.gov.co/ministerio/recursos-humanos/Actos_Administrativos/Informe_2.pdf
- Universidad América Latina. (2013). VLSM | ual. Recuperado el 12 de octubre, de http://ual.dyndns.org/biblioteca/Redes/Pdf/Unidad%2007.pdf