Implementación del Protocolo VLSM para la División Eficiente de Subredes

Brayan Aldahir Torres Yandun

Yan carlos Pinchao guerra

Institución universitaria del putumayo

Electiva profesional V

Daniel ceron

22 de septiembre de 2025

Introducción

Implementación del Protocolo VLSM para la División Eficiente de Subredes

Este documento presenta el desarrollo de un programa en Python diseñado para implementar la técnica de máscara de subred de longitud variable (VLSM). El sistema permite calcular divisiones de red optimizadas a partir de una dirección IP base y los requisitos específicos de hosts por subred. La implementación incluye funciones para validación de direcciones IP, cálculo automático de máscaras variables y generación de rangos de direcciones útiles. Los resultados demuestran la eficacia del algoritmo para optimizar el espacio de direccionamiento IPv4 en escenarios de redes heterogéneas.

Palabras clave: VLSM, subredes, IPv4, Python, administración de redes

Primero que todo importamos las librerías para hacer las operaciones necesarias y declaramos la primera función para validar si la ip es valida.

```
import ipaddress
import math

def validar_ip(ip_str):
    try:
        ipaddress.IPv4Address(ip_str)
        return True
    except ValueError:
        return False
```

La siguiente función lo que realiza es que dentro de un ciclo while pide la IP base y con la función y manda a llamar la anterior función llamada validar IP esta nos retorna la IPBASE y si no es correcta la ip nos devuelve un mensaje.

```
def solicitar_ip_base():
    while True:
        ip_base = input("Ingrese la IP base (ejemplo 192.168.1.0): ").strip()
        if validar_ip(ip_base):
            return ip_base
        print("IP inválida. Intente nuevamente.")
```

Creamos otra función para el numero de hosts primero que todo creamos un arreglo con el nombre hosts después creamos el ciclo para recorrer las subredes que se necesitan y que se ingrese por consola y dentro de un ciclo while recorremos el número de subredes para que asi mismo se ingrese el número de host lo siguientes es validar que no se ingrese un numero menor a 0 cuando se agregue los números de hosts estos se agregan al arreglo que inicialmente creamos llamado hosts.

```
def solicitar_hosts_por_subred(num_subredes):
    hosts = []
    for i in range(num_subredes):
        while True:
            try:
                h = int(input(f"Ingrese el número de hosts para la subred {i+1}: "))
                if h > 0:
                    hosts.append(h)
                    break
                else:
                    print("El número de hosts debe ser mayor a 0.")
            except ValueError:
                print("Ingrese un número válido.")
    return hosts
```

Declaramos la última función llamamos los parámetros necesarios que nos retornas las otras funciones. primero que todo organizamos de mayor a menor los hosts de las subredes con la función sorted y cuando ya se los organiza se los retorna a hosts_por_subred después declaramos un arreglo llamado subred y mandamos a llamar la ipbase por medio de la variable ipactual

Recorremos el arreglo numero de host por las subredes que se hiso la petición en la primera operación en la variable bits_hosts lo que se realiza es la operación del algoritmo para calcular el numero de bits donde lo que se calcule se lo redondea con la función math.ceil al haber realizado el calculo del numero de bits se procede a realizar el calculo de la mascara restando el numero de bits que son 32 de IPV4 restando a los bits calculados anteriormente.

Ahora para imprimir la red declaramos una función que nos ayuda a que sea declarada como ip llamamos a ipactual y se digita la primer mascara listamos el numero de hosts por cada IP en el rango lo que se hace es donde comienza la ip le restamos -1 para saber la ultima ip disponible dentro de esa subred por ultimo agregamos todo lo calculado anteriormente y con la funcion str lo que hace es reiniciar los campos para ingresar los nuevos fuera del ciclo a la ipactual le agregamos lo de la red broadcast +1

```python
def calcular_subredes_vlsm(ip_base, hosts_por_subred):
    hosts_por_subred = sorted(hosts_por_subred, reverse=True)
    subredes = []
    ip_actual = ipaddress.IPv4Address(ip_base)

    for num_hosts in hosts_por_subred:
        bits_host = math.ceil(math.log2(num_hosts + 2))
        mascara = 32 - bits_host
        red = ipaddress.ip_network(f"{ip_actual}/{mascara}", strict=False)
        hosts = list(red.hosts())
        rango = f"{hosts[0]} - {hosts[-1]}" if hosts else "Sin hosts disponibles"
        subredes.append({
            'network': str(red.network_address),
            'mask': str(red.netmask),
            'broadcast': str(red.broadcast_address),
            'rango': rango,
            'num_hosts': num_hosts
        })
        ip_actual = red.broadcast_address + 1
    return subredes
```

Declaramos esta ultima funcion para imprimir la tabla vlsm

```
def mostrar_resultados(subredes):
    print("\nSubredes generadas:")
    for idx, subred in enumerate(subredes, 1):
        print(f"Subred {idx}:")
        print(f"  Network:   {subred['network']}")
        print(f"  Máscara:   {subred['mask']}")
        print(f"  Broadcast: {subred['broadcast']}")
        print(f"  Rango:     {subred['rango']}")
        print(f"  Hosts:     {subred['num_hosts']}")
        print()
```

Esto es lo que se pide al comienzo las entrada de los datos como el numero de subredes valida que el numero de subredes sea mayor que 0 y aplica las funciones explicadas anteriormente los host por subred las subredes o las ip creadas por cada subred y por ultimo llama a la función mostrar resultados para imprimir lo que se calculo

```
def main():
    ip_base = solicitar_ip_base()
    while True:
        try:
            num_subredes = int(input("Ingrese el número de subredes: "))
            if num_subredes > 0:
                break
            else:
                print("El número de subredes debe ser mayor a 0.")
        except ValueError:
            print("Ingrese un número válido.")
    hosts_por_subred = solicitar_hosts_por_subred(num_subredes)
    subredes = calcular_subredes_vlsm(ip_base, hosts_por_subred)
    mostrar_resultados(subredes)

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