INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

PROYECTO

CALCULADORA IP

NOMBRE DE LOS ALUMNOS:

GARCÍA QUIROZ GUSTAVO IVAN

GRUPO: 5CV3

NOMBRE DEL PROFESOR: ALCARAZ TORRES JUAN JESUS

REDES DE COMPUTADORAS

27/11/2023

Índice

Introducción

La Calculadora IP facilita la obtención de información esencial a partir de una dirección IP y su máscara de red. Entre los datos que puedes obtener se encuentran la dirección de broadcast, la red, la Cisco wildcard mask y el número de IPs en la red. Además, al introducir un segundo valor de máscara de red, tendrás la capacidad de diseñar subredes, permitiéndote experimentar con diferentes configuraciones y observar los resultados.

Para comprender el proceso, es crucial entender el concepto de subneteo.

Subnetear implica tomar un conjunto de direcciones IP donde todas sean locales entre sí y dividirlas en distintos rangos o subredes. Cada rango posee direcciones IP que son independientes de las demás.

Si deseas determinar cuántos hosts hay en un rango IP específico, el primer paso es identificar cuántos bits están asignados para los hosts. Por ejemplo, consideremos la dirección IP 192.107.2.4 con una máscara de red de 255.255.255.0. Una vez establecido que el ID de red es 192.107.2 y el ID de host es 4, puedes utilizar la fórmula $2^N - 2$, donde N es el número de bits para los hosts. En este caso, obtendríamos $2^8 - 2 = 254$ hosts. Esto significa que en nuestro ejemplo, la red 192.107.2.0 tiene 254 direcciones IP posibles, todas locales entre sí.

Para subdividir una red en subredes, es necesario quitar bits de la sección de hosts. Para planificar la segmentación, dos consideraciones importantes son el número de usuarios por red y la cantidad de redes necesarias. Cuantos más usuarios por red, menos redes necesitarás, y viceversa. Cabe destacar que por cada red creada, se pierden 2 IPs, una asociada al broadcast y otra a la identificación de la misma red.

Objetivos

- Crear una herramienta de red que sea accesible y fácil de usar para calcular información clave a partir de direcciones IP y máscaras de red.
- Permitir a los usuarios usar la herramienta que deberá ser capaz de calcular y mostrar de manera clara los detalles de las subredes resultantes.
- Contribuir a mejorar la comprensión del subneteo y sus implicaciones.
- Proporcionar información completa y la calculadora IP deberá generar información detallada, incluyendo direcciones de broadcast, redes, Cisco wildcard mask y el número de IPs en cada red. Esto ayudará a los usuarios a obtener una visión completa de la topología de la red.

Material

1 computadora personal.

1 software para compilar código.

Desarrollo

Crearemos una calculadora de IP para hacer subredes usando el IDE Visual Studio Code para realizar y visualizar nuestro programa y lenguaje C para programarlo.

Este programa en C se utiliza para obtener información sobre subredes a partir de una dirección IP y una máscara de subred en notación CIDR. Se explica una guía paso a paso de cómo usarlo:

Instrucciones

1. Ejecuta el programa. Se te pedirá que ingreses una dirección IPv4. Se debe ingresarla en formato de punto decimal (por ejemplo, 192.168.1.1).

2. El programa verificará si la dirección IP que ingresaste es válida. Si no es válida, se te pedirá que ingreses una nueva dirección IP. Si es válida, el programa continuará.

3. A continuación, se pedirá que ingreses una máscara de subred en notación CIDR. Debes ingresar un número entre 0 y 32.

4. El programa verificará si el número CIDR que ingresaste es válido. Si no es válido, se te pedirá que ingreses un nuevo número. Si es válido, el programa continuará.

5. El programa imprimirá la dirección IP que ingresaste, junto con su clase correspondiente (A, B, C, D o E).

6. Luego, el programa llamará a la función ‘printSubnetInfo’ para imprimir información sobre la subred.

7. A continuación, se te pedirá que ingreses el número de redes requeridas. Si ingresas 'q', el programa terminará. Si ingresas un número, el programa calculará la cantidad de bits necesarios para contener las redes requeridas.

8. Si la cantidad de redes es demasiado grande para ser acomodada, se te informará y se te pedirá que ingreses un nuevo número de redes.

9. Finalmente, el programa imprimirá información sobre cada subred que se creará.

Este programa debe ser compilado y ejecutado en un entorno que soporte el lenguaje de programación C, como GCC en Linux o MinGW en Windows.

Código fuente

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>
#include <stdint.h>

int ipVerify(char* ipAddress, unsigned char* octetArray) {
    // La función verifica que se ha ingresado una IP válida, y luego
actualiza el array de octetos con los octetos validados.

    char* token;
    int i = 0;
    int j = 0;
    int periodCount = 0;

    // Continuaremos obteniendo tokens mientras no sea nulo
    token = strtok(ipAddress, ".");
    while (token != NULL) {

        // Recorre cada carácter y verifica si es un dígito
        // Si no lo es, interrumpe el ciclo. Usamos j para ver si se iteró
la cantidad correcta de veces
        for (j=0; j<strlen(token); j++) {
            if (isdigit(token[j]) == 0) {
                break;
            }
        }

        // Si se ha ingresado la cantidad correcta de dígitos, confirma el
octeto como validado y lo agrega al array
        if (strlen(token) > 0 && strlen(token) < 4 && j == strlen(token)
                && atoi(token) < 256 && atoi(token) >= 0) {
            periodCount++;
            octetArray[i] = atoi(token);
        } else {
            // No tiene sentido continuar si incluso un octeto falla la
prueba
            break;
        }

        i++;
```

```
        token = strtok(NULL, ".");
    }

    if (periodCount != 4) {
        return 0;
    } else {

        return 1;
    }
}

void printSubnetInfo(uint32_t* addressOctets, int* CIDR, int* subnetBits) {
// Imprime información sobre la subred dada.
// Toma punteros a los datos requeridos, sin embargo, no cambia nada dentro
de ellos.
// Cualquier manipulación requerida se hace con variables locales

    uint32_t netAddress;
    uint32_t netMask;

    netMask = (0xFFFFFFFF << (32 - (*CIDR + *subnetBits)) & 0xFFFFFFFF);
    netAddress = *addressOctets & netMask;

    // Desempaqueta y muestra la dirección de red
    printf("\nDireccion de red: %d.%d.%d.%d/%d\n", (netAddress >> 24) &
0xFF, (netAddress >> 16) & 0xFF,
                            (netAddress >> 8) & 0xFF, (netAddress) & 0xFF,
*CIDR + *subnetBits);

    // Resta la dirección de red de la dirección de broadcast y resta uno
del resultado para el total de hosts
    printf("Total hosts: %d\n", ((netAddress | ~netMask) - netAddress) - 1);

    // Muestra la primera dirección de host sumando a cada uno de nuestros
octetos desempaquetados
    printf("Primera direccion de host: %d.%d.%d.%d\n", ((netAddress + 1) >>
24) & 0xFF, ((netAddress + 1) >> 16) & 0xFF,
                            ((netAddress + 1) >> 8) & 0xFF, (netAddress + 1)
& 0xFF);

    //Realiza un OR bit a bit con la dirección int y la máscara negada para
obtener la dirección de broadcast en la variable
    netAddress = netAddress | ~netMask;
```

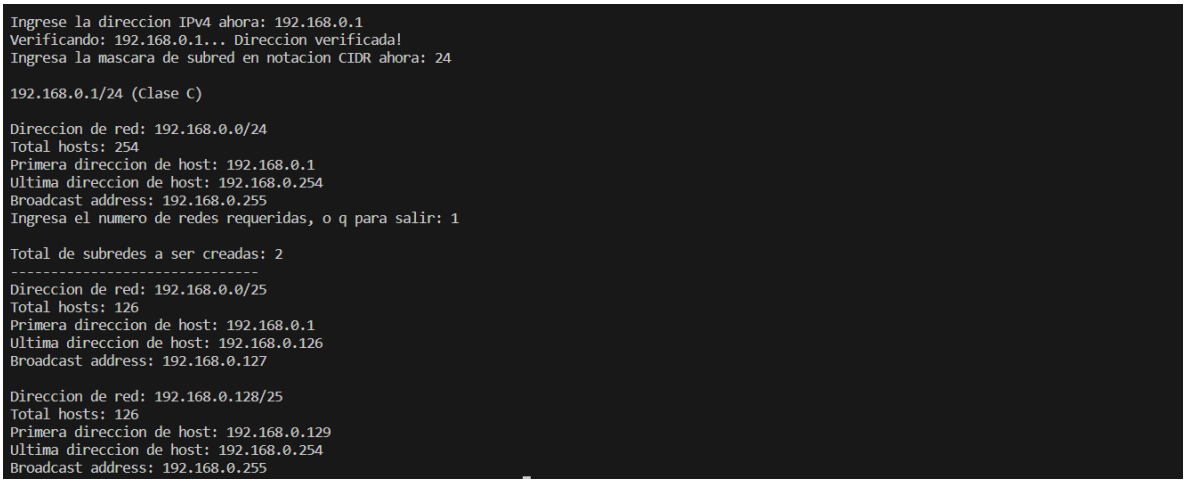
```
    // Reste de la dirección de broadcast para obtener la dirección del host
final
    printf("Ultima direccion de host: %d.%d.%d.%d\n", ((netAddress - 1) >>
24) & 0xFF, ((netAddress - 1) >> 16) & 0xFF,
                           ((netAddress - 1) >> 8) & 0xFF, (netAddress - 1)
& 0xFF);

    // Desempaqueta y muestra la dirección de broadcast
    printf("Broadcast address: %d.%d.%d.%d\n", (netAddress >> 24) & 0xFF,
(netAddress >> 16) & 0xFF,
                           (netAddress >> 8) & 0xFF, (netAddress) & 0xFF);
}

int main() {

    char ipAddress[18];
    char buffer[4];
    int CIDR;
    unsigned char* octetArray;
    octetArray = calloc(4, sizeof(char));
    uint32_t addressOctets;

    int subnetNumber;
    int subnetBits = 0;
    int totalSubnets = 0;
    uint32_t currentSubnet;
    int i;

    // conseguir la dirección IP
    while (1) {
        printf("Ingrese la direccion IPv4 ahora: ");
        fgets(ipAddress, 17, stdin);
        ipAddress[strlen(ipAddress)-1] = '\0';

        printf("Verificando: %s... ", ipAddress);

        // Verifícador de IP
        if (ipVerify(ipAddress, octetArray) == 0) {
            printf("IP no valida ingresada.\n");
        } else {
            printf("Direccion verificada!\n");
            break;
        }
    }
```

```
    //Obtén el número CIDR
    while (1) {
        printf("Ingresa la mascara de subred en notacion CIDR ahora: ");
        fgets(buffer, 4, stdin);

        CIDR = atoi(buffer);

        if (CIDR > 0 && CIDR < 32) {
            break;
        } else {
            printf("Se ha introducido un CIDR invalido. Intenta otra
vez.\n");
        }
    }

    printf("\n%d.%d.%d.%d/%d ", octetArray[0], octetArray[1], octetArray[2],
octetArray[3], CIDR);

    if (octetArray[0] > 239) {
        printf("(Clase E)\n");
    } else if (octetArray[0] > 223) {
        printf("(Clase D)\n");
    } else  if (octetArray[0] > 191) {
        printf("(Clase C)\n");
    } else if (octetArray[0] > 127) {
        printf("(Clase B)\n");
    } else {
        printf("(Clase A)\n");
    }

    // Empaqueta los bits de la dirección IP en un entero
    addressOctets = (octetArray[0] << 24) | (octetArray[1] << 16) |
(octetArray[2] << 8) | (octetArray[3]);

    // Llama a la función subnetinfo para la red
    printSubnetInfo(&addressOctets, &CIDR, &subnetBits);

    do {
        printf("Ingresa el numero de redes requeridas, o q para salir: ");
        fgets(buffer, 4, stdin);
        subnetNumber = atoi(buffer);

        if (subnetNumber == 0) {
            printf("Saliendo...\n");
            exit(0);
```

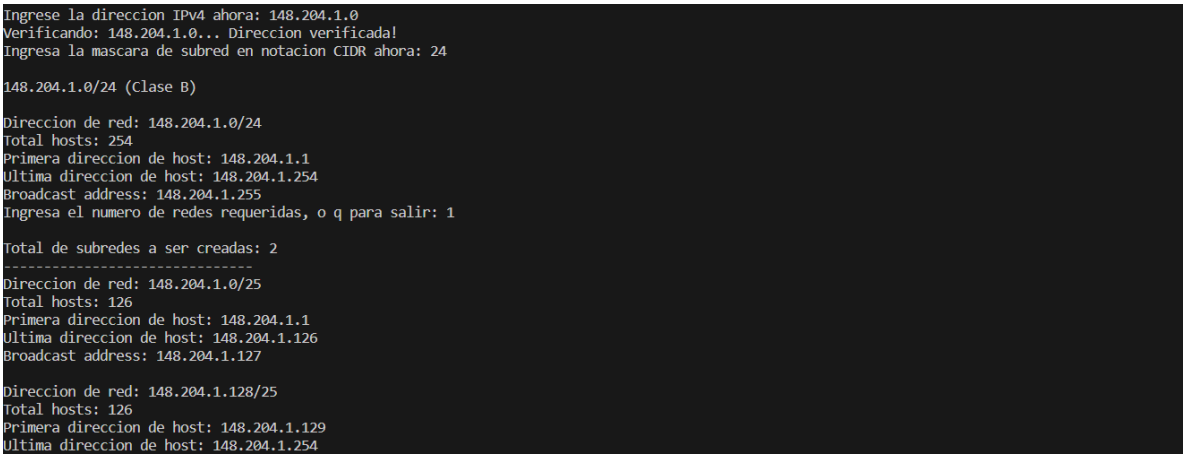
```
        }

        // Determina la cantidad de bits necesarios para contener las redes
requeridas
        while (subnetNumber > totalSubnets) {
            subnetBits++;
            totalSubnets = pow(2, subnetBits);
        }

        // Verifica que tenemos la cantidad necesaria de bits para subnetear
exitosamente
        if ((CIDR + subnetBits) > 31) {
            printf("La cantidad de redes es demasiado grande para ser
acomodada.\n");
        }
    } while ((CIDR + subnetBits) > 31);

    printf("\nTotal de subredes a ser creadas: %d\n-------------------------
------", totalSubnets);
    // Construye los bits de red de la subred, luego imprime la información
    for (i=0; i<totalSubnets; i++) {
        currentSubnet = (addressOctets & ((0xFFFFFFFF << (32 - CIDR)) &
0xFFFFFFFF))
                | i << (32 - (CIDR + subnetBits));
        printSubnetInfo(&currentSubnet, &CIDR, &subnetBits);
    }

    free(octetArray);

    return 0;
}
```

Capturas de pantalla

Dirección de red: 192.168.0.1 /24



Dirección de red: 148.204.1.0 /24



Conclusiones

- GARCÍA QUIROZ GUSTAVO IVAN

El subneteo es esencial en esta práctica porque permite una gestión eficiente de direcciones IP al dividir una red en subredes más pequeñas. Esto facilita la optimización de recursos, la segmentación lógica de la red, la reducción del tráfico de broadcast, la mejora de la seguridad, la adaptabilidad a diferentes requisitos y la optimización del uso del ancho de banda. Además, el subneteo facilita el escalado de la red y contribuye a la reducción de colisiones, mejorando así el rendimiento general de la red. En resumen, el subneteo es una herramienta fundamental para diseñar y administrar redes de manera efectiva y eficiente.

- a

Bibliografía

- Oviedo, B. O. B., Samaniego, E. S. M. & Murillo, J. M. O. (2018). Fundamentos de redes (1.a ed.). Grupo Compás.
- Yedlapalli, S. & Md, Y. R. (2022). Fundamentos de las redes y protocolos informáticos (Spanish Edition). Ediciones Nuestro Conocimiento.