INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

REDES DE COMPUTADORAS

"MI TRAMA-ESTRUCTURA"

Abstact

In this report we will see an estructure of the frame, where you can find out the different parts on the basic frame. By the way, here you will check the languages, routers, ID and line code.

By:

MARCOS OSWALDO VÁZQUEZ MORENO

Professor:
MSc. NIDIA ASUNCIÓN CORTEZ DUARTE

DICIEMBRE 2018

1	Mensaje	n bytes					
	Idioma	3 bits	010	Español			
			011	Inglés			
			100	Portugués			
			101	Francés			
			110	Alemán			
2	Enrutamiento	1 bit	0	Estático			
			1	Dinámico			
	Id origen/destino	1 byte	01	2CM7			
			10	3CM5			
			11	5CM3			
3	Control de Flujo	2-10 bits	0	Parar y esperar			
			1	Ventana deslizante	0	Go Back N	{
					1	Rechazo Selectivo	
	Control de Error	4-35 bits	00	Bit de paridad	0	Par	FCS
					1	Impar	
			01	CRC	0	8 bits	
					1	32 bits	
			11	Checksum 32 bits			
4	Medio	3 bits	0	Alámbrico	01	Fibra Óptica	
					10	Cable Coaxial	
					11	UTP	
			1	Inalámbrico	00	Infrarrojo	
					01	WiFi	
					10	NFC	
	Código de línea	6 bits	0	Unipolar			
			0	Polar			
			0	RZ			
			0	NRZ			
			0	AMI			
			0	Manchester			

```c
/*
VAZQUEZ MORENO MARCOS OSWALDO
REDES DE COMPUTADORAS
EJERCICIO 1
CLASIFICACION DE CLASES EN UNA DIRECCION IP, MASCARA DE RED, TIPO, RANGO DE HOST, BROADCAST, REPETIR
*/

#include<stdio.h>
#include<stdlib.h>

//int validacion(int ip);

int main(){


//arreglo de cuatro octetos
unsigned char IP[4]= {0,0,0,0}, MR[4]= {0,0,0,0};
char a= 'y';
do{

    printf("ingrese direccion ip valida: \n");
    scanf("%u.%u.%u.%u",&IP[0], &IP[1], &IP[2], &IP[3]);
    fflush(stdin);
//validacion para entrar al distinto tipo de casos
if(IP[0] & 128) //clase A
    {
        if(IP[0] & 64) //clase B
        {
            if(IP[0] & 32) //clase C
            {
                if(IP[0] & 16) //clase D
                {
                    if(IP[0] & 8) //clase E
                    {
                        printf("Clase E\n");
                    }
                    else
                    {
                        printf("Clase E\n");
                    }
                }
                else
                {
                    printf("Clase D\n");
                }
            }
            else
            {
                printf("---------------------------------\n");
                printf("CLASE C\n");
                MR[0]= 255; //VALORES DE LS OCTETOS DE CADA MASCARA DE RED
                MR[1]= 255;
                MR[2]= 255;
                MR[3]= 0;
                if(IP[3]==255){
                    printf("TIPO BROADCAST\n");
                }
                else if(IP[3]==0){
```

```
                    printf("TIPO RED\n");
                }
                else
                    printf("TIPO HOST\n");
                printf("MASCARA DE RED: %u.%u.%u.%u\n", MR[0],MR[1], MR[2], MR[3])
;
                printf("RED: %u.%u.%u.%u\n", IP[0]&MR[0], IP[1]&MR[1], IP[2]&MR[2]
, IP[3]&MR[3]);
                printf("BROADCAST: %u.%u.%u.%u\n", IP[0] | (unsigned char)~MR[0],
IP[1] | (unsigned char)~MR[1], IP[2] | (unsigned char)~MR[2], IP[3] | (unsigned ch
ar)~MR[3]); //CASTEO DE ENTERO A UNSIGNED CHAR
                printf("RANGO DE HOST: %u.%u.%u.%u A %u.%u.%u.%u\n", IP[0] & MR[0]
, IP[1] & MR[1], IP[2] & MR[2], (IP[3] &MR[3])+1, IP[0] | (unsigned char)~MR[0], I
P[1] | (unsigned char)~MR[1], IP[2] | (unsigned char)~MR[2], (IP[3] | (unsigned ch
ar)~MR[3])-1);
            }
        }
        else
        {
            printf("---------------------------------\n");
            printf("CLASE B\n");
            MR[0]= 255; //VALORES DE LS OCTETOS DE CADA MASCARA DE RED
            MR[1]= 255;
            MR[2]= 0;
            MR[3]= 0;
            if(IP[3]==255 && IP[2]==255){
                    printf("TIPO BROADCAST\n");
                }
                else if(IP[3]==0 && IP[2]==0){
                    printf("TIPO RED\n");
                }
                else
                    printf("TIPO HOST\n");
            printf("MASCARA DE RED: %u.%u.%u.%u\n", MR[0],MR[1], MR[2], MR[3]);
            printf("RED: %u.%u.%u.%u\n", IP[0]&MR[0], IP[1]&MR[1], IP[2]&MR[2], IP
[3]&MR[3]);
            printf("BROADCAST: %u.%u.%u.%u\n", IP[0] | (unsigned char)~MR[0], IP[1
] | (unsigned char)~MR[1], IP[2] | (unsigned char)~MR[2], IP[3] | (unsigned char)~
MR[3]);
            printf("RANGO DE HOST: %u.%u.%u.%u A %u.%u.%u.%u\n", IP[0] & MR[0], IP
[1] & MR[1], IP[2] & MR[2], (IP[3] &MR[3])+1, IP[0] | (unsigned char)~MR[0], IP[1]
 | (unsigned char)~MR[1], IP[2] | (unsigned char)~MR[2], (IP[3] | (unsigned char)~
MR[3])-1);
        }
    }
    else
    {
        printf("---------------------------------\n");
        printf("CLASE A\n");
        MR[0]= 255; //VALORES DE LS OCTETOS DE CADA MASCARA DE RED
        MR[1]= 0;
        MR[2]= 0;
        MR[3]= 0;
        if(IP[3]==255 && IP[2]==255 && IP[1]==255){
                        printf("TIPO BROADCAST\n");
                    }
                    else if(IP[3]==0 && IP[2]==0 && IP[1]==0){
                        printf("TIPO RED\n");
                    }
                    else
                        printf("TIPO HOST\n");
```

```
107.
108.               printf("MASCARA DE RED: %u.%u.%u.%u\n", MR[0], MR[1], MR[2], MR[3]
    );
109.               printf("RED: %u.%u.%u.%u\n", IP[0] & MR[0], IP[1] & MR[1], IP[2] &
     MR[2], IP[3] &MR[3]);
110.               printf("BROADCAST: %u.%u.%u.%u\n", IP[0] | (unsigned char)~MR[0],
    IP[1] | (unsigned char)~MR[1], IP[2] | (unsigned char)~MR[2], IP[3] | (unsigned ch
    ar)~MR[3], IP[4] | (unsigned char)~MR[4]);
111.               printf("RANGO DE HOST: %u.%u.%u.%u A %u.%u.%u.%u\n", IP[0] & MR[0]
    , IP[1] & MR[1], IP[2] & MR[2], (IP[3] &MR[3])+1, IP[0] | (unsigned char)~MR[0], I
    P[1] | (unsigned char)~MR[1], IP[2] | (unsigned char)~MR[2], (IP[3] | (unsigned ch
    ar)~MR[3])-1);
112.           };
113.
114.       printf("desea repetir el proceso (Y/N)\n");
115.       scanf("%c", &a);
116.       fflush(stdin);
117.
118.       }while(a == 'Y' || a == 'y');
119.       return 0;
120.
121.       }
```

A continuación, en la imagen 1 se muestra la salida del código y su ejecución.

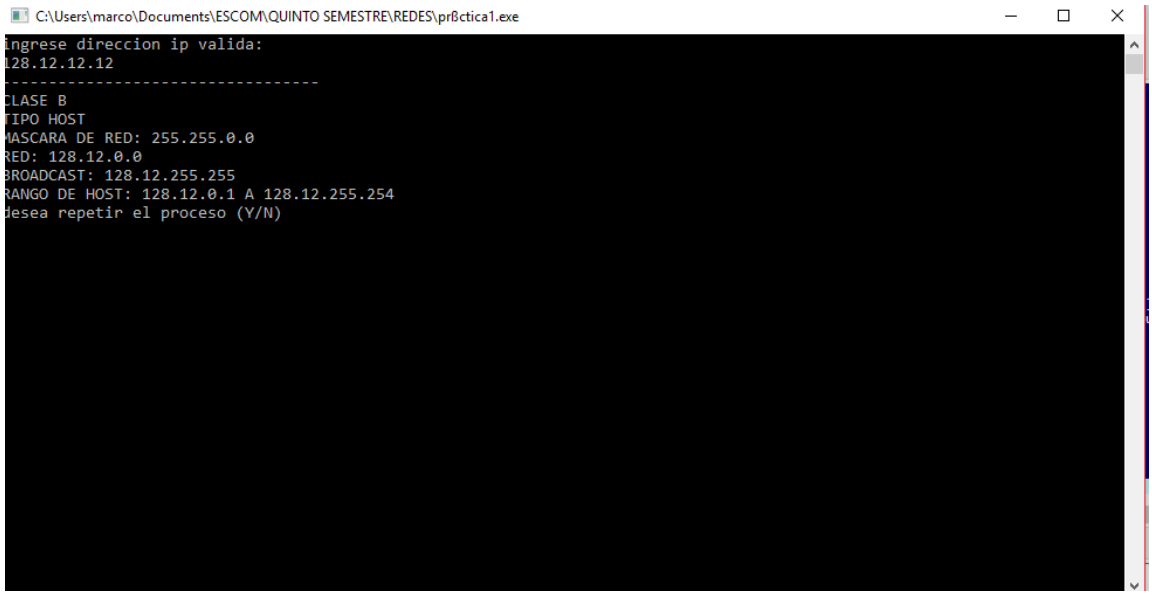

Imagen 1. Captura mi Trama.