PRÁCTICA 5 – ENVIO DATOS

FECHA: 25/10/21

NOMBRE DEL EQUIPO: EL SIUU TEAM

PARTICIPANTES: -FISCHER SALAZAR CÉSAR EDUARDO
-LÓPEZ GARCÍA JOSÉ EDUARDO
-MEZA VARGAS BRANDON DAVID

UNIDAD ACADÉMICA: REDES DE COMPUTADORAS

PANORÁMICA

INTRODUCCION

La presente práctica pretende que, empleando como recursos indispensables en la creación de códigos como son los manuales de Linux, su terminal, los sockets crudos que se han estado programando desde anteriores prácticas y el software de Wireshark, se permita realizar con éxito el envío de una trama, así como observar y dar una explicación concisa de lo que se ha capturado dentro del Wireshark para considerar que la ejecución se realizó satisfactoriamente.

De igual forma, en la presente práctica se hará la estructuración correcta de una trama, para de esta forma, como se mencionó enviarla a través de la red.

Antes de seguir recordemos algo sobre las tramas.

La trama se encuentra en la capa de enlace responsable de la transmisión y separa el flujo de datos de bits en bloques o tramas. Recordemos que una trama debe tener al menos 64 bytes para que funciones la detección de colisiones y puede tener un máximo de 1518 bytes. La estructura de la trama es la siguiente:

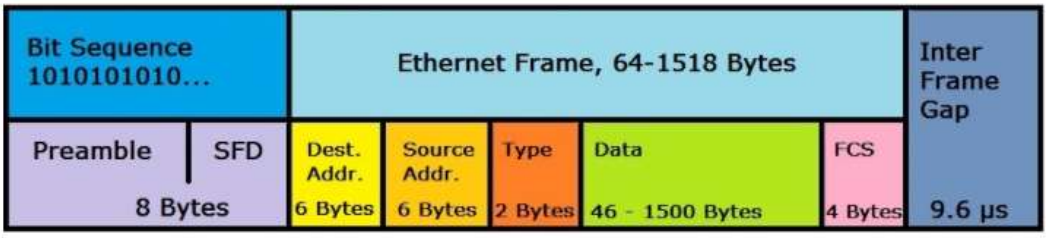

Imagen 1. Estructura de la trama.

OBJETIVOS

OBJETIVO PRINCIPAL: PROGRAMAR SOCKETS CRUDOS Y ESTRUCTURAR Y ENVIA UNA TRAMA ATRAVEZ DE LA RED

OBJETIVO SECUNDARIO. ENVIAR Y CAPTURAR TRAMA BIEN ESTRUCTURADA CON LA AYUDA DEL PROGRAMA WIRESHARK

RECURSOS NECESARIOS PARA REALIZAR LA PRÁCTICA

Manuales man socket , man 7 ip, man packet
Compilador de c
Terminal de Linux
Navegador de internet

PARTE 1: DIAGRAMA DE FLUJO

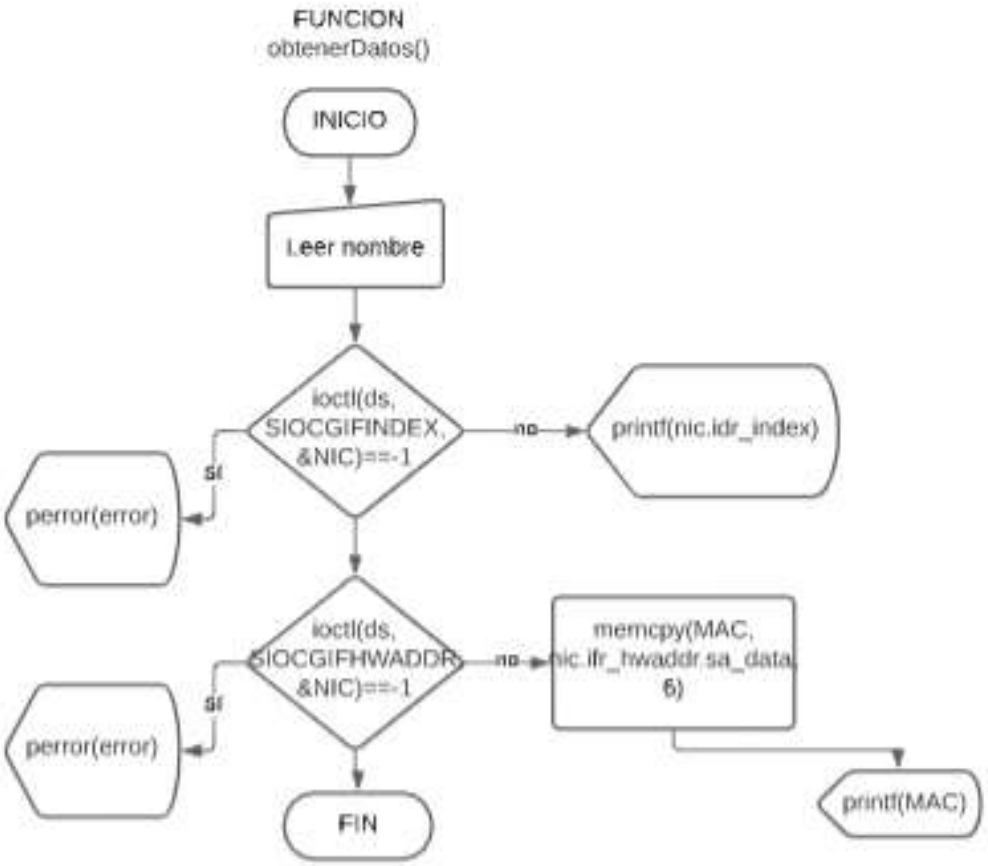

Imagen 2. Diagrama de flujo de función obtenerDatos()

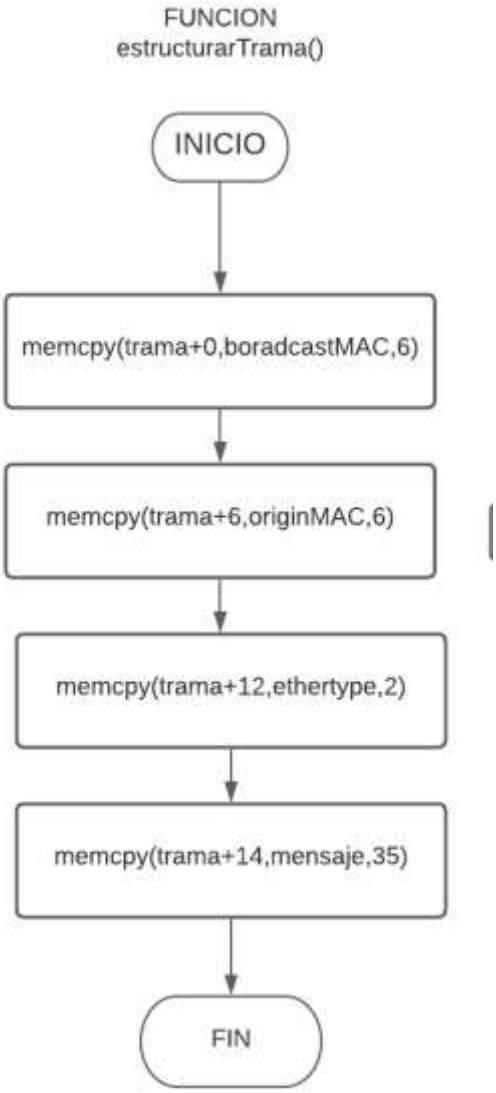

Imagen 3. Diagrama de flujo de función estructurarTrama()

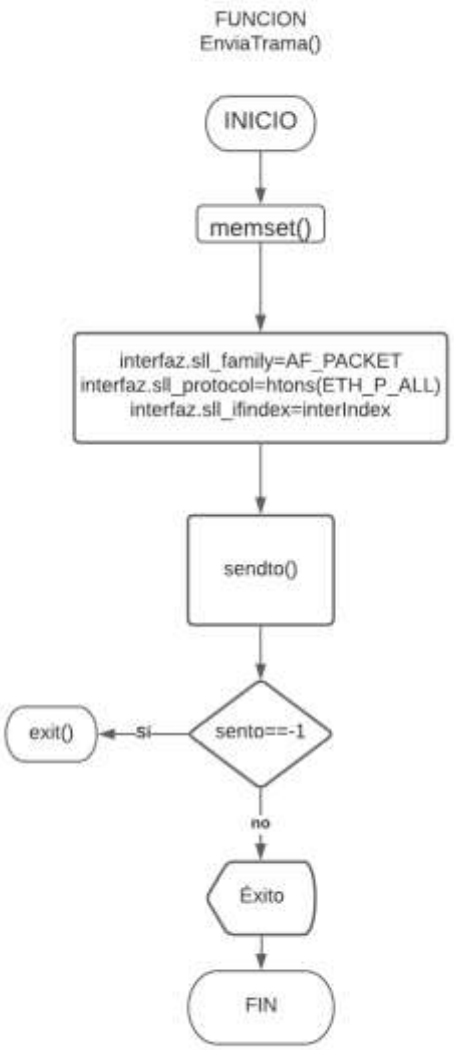

Imagen 4. Diagrama de flujo función enviarTrama()

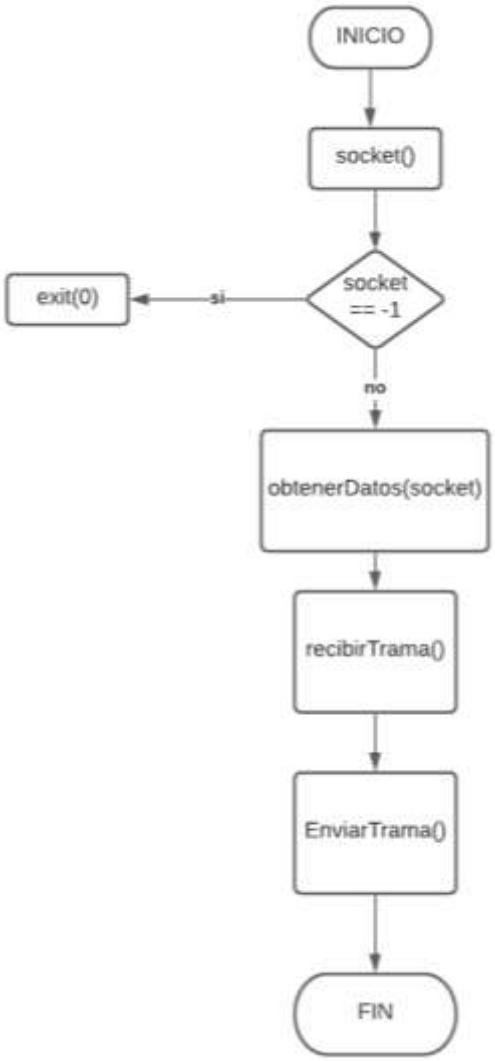

Imagen 5. Diagrama de flujo main()

PARTE 2: CÓDIGOS, COMANDOS Y EJECUCIÓN Y EXPLICACIÓN.

2.1 INCLUIR CODIGO EXPLICANDO LAS ESTRUCTURAS DEL PROGRAMA, Y FUNCIONES USADAS Y MENCIONAR DE QUE MANUAL DE LINUX CONSULTARON, CAMBIAR NOMBRE DE SUS VARIABLES Y ESTRUCTURAS DE FORMA PERSONAL. RECUERDEN QUE LAS MEJORAS QUE LE HAGAN AL PROGRAMA VISTO EN CLASE AUMENTA SU CALIFICACIÓN.

```c
#include <sys/socket.h>
#include <linux/if_packet.h>
#include <linux/if_ether.h>
#include <net/ethernet.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <net/if.h>
#include <string.h>
#include <stdio_ext.h> //para __fpurge(stdin)

//varibales para la estructura de la trama

unsigned char originMAC[6]; //para la MAC origen
unsigned char broadcastMAC[6]={0xff,0xff,0xff,0xff,0xff,0xff}; //longitud de 6 bytes e inicializamos para broadcast
unsigned char ethertype[2]={0x0c,0x0c}; //2 bytes para ethertype y se inicializa
//para la trama a enviar usamos un tamaño de  1514
unsigned char sentTrama[1514];

/*
    Esta función de obtener datos recibe el descriptor del socket y nos devuelve
    el indice de nuestra interfaz de red.
    ESta funcion se encarga de obtener la MAC de nuestra maquina e imprimirla
    para identificarla en las tramas que recibiremos.
    De igual forma se encarga de regresar el indice de nuestra interfaz de red a partir del nombre de
    nuestra interfaz.
*/

int obtenerDatos (int socketDesc){
    struct ifreq nic; //estructura ifreq para consultar datos de nuestra interfaz, man netdevice
    char interName[20]; //variable que alamcenara el nombre de nustra interfaz
    int i,interIndex; //variable apra loop y para el indice

    printf("\nInserta el nombre de la interfaz de red: ");
    scanf("%s", interName); //leemos el nombre de la interfaz de red
```

```c
//almacenamos el nombre de la interfaz en nic.ifr_name
strcpy(nic.ifr_name,interName);

 /*
    La función ioctl la usamos para manipular los valores de los paramatros de un socket.
    Proporciona una iterfaz para controlar el comportamiento de dispositivos y de sus descriptores,
    en este caso de sockets
*/
// obtener el indice usando SIOCGIFINDEX
 if(ioctl(socketDesc,SIOCGIFINDEX,&nic)==-1){
    perror("\n Error al obtener el indice de red"); //error si no obtenemos el indice
    exit(0);    // salimos del programa
}
else{

    interIndex=nic.ifr_ifindex; //almacenamos el indice en nuestra variable
    printf("\n El indice es: %d", interIndex); //imprimimos nuestro indice de red


        //obtener la MAC usando SIOCGIFHWADDR
        if(ioctl(socketDesc,SIOCGIFHWADDR,&nic)==-1){
            perror("\nError al obtener la MAC"); //error si no obtiene la MAC
            exit(0); //salimos del programa
        }
        else{
            //si obtenemos la mac la almacenamos en nuestra variable global originMAC,
           memcpy(originMAC, nic.ifr_hwaddr.sa_data,6);

           printf("\n La MAC es: ");
           //realizamos un for hasta 6 para imprimir nuestra MAC, usamos el formato hexadecimal usando %.2x
            for(i=0;i<6;i++)
            {
                printf("%.2x: ", originMAC[i]);
            }
            printf("\n"); //salto de linea para leer mejor la salida del programa

        }
```

```
    }

    return interIndex; //Retornamos el indice
}

/*
    Esta función se encarga de estructurar la trama, recibe la trama, como sabemos debe ser
    un unsigned char con el tamaño de 1514.
    lo que se hace es copiar en la trama los valores de la mac origen, de la mac de broadcast, el ethertye
    y los datos, en este caso un mensaje
*/
void estructuraTrama(unsigned char *trama){
    unsigned char mensaje[50];

    __fpurge(stdin); //Limpíamops buffer del teclado
    printf("Ingrese el mensaje de la trama: ");
    fgets(mensaje, sizeof(mensaje),stdin); //esperamos a que se escriba un mensaje a enviar


    /*
        memcpy nos sirve para copiar n caracteres de un espacio de memoria
        a un espacio de memoria destino, en esta caso el destino es la trama  y se copiaran
        nuestros datos.
    */
    memcpy(trama+0,broadcastMAC,6); //copiamos en las primeras 6 posiciones la mac de broadcast
    memcpy(trama+6,originMAC,6); //copiamos a partir de la posicion 6 la mac origen
    memcpy(trama+12,ethertype,2); //posteriormente copiamos los bytes ppara ethertype
    memcpy(trama+14,mensaje, strlen(mensaje)+1);//copiamos nuestro mensaje a la trama
}
/*
    Esta función envia la trama, recibe el descriptor del socket, el indice de nuestra interfaz de red
    y la trama a enviar
*/
void EnviarTrama (int socketDesc, int interIndex, unsigned char *trama)
{

    int sendTam;
    struct sockaddr_ll interfaz; //usamos la estructura sockaddr_ll, esta es una direccion de la capa fisica
                                 // man packet, de esta manera podemos mandar paquetes de nuestra interfaz


    /*
        con memset copiamos el caracter 0x00, a los primeros n caracteres de nuestra interfaz
        en este caso los n caracterez son el tamaño de la interfaz
    */
    memset(&interfaz, 0x00, sizeof(interfaz));

    interfaz.sll_family = AF_PACKET; //le asignamos la familia AF_PACKET
    interfaz.sll_protocol = htons(ETH_P_ALL); //Le asignamos todos los protocolos
    interfaz.sll_ifindex = interIndex; //el indice será el obtenido previamente

    /*
        Aquí mandamos la trama haciendo uso de la función sendto, con una tamaño de 60
    */

    sendTam = sendto(socketDesc,trama,60,0,(struct sockaddr *)&interfaz, sizeof(interfaz));

        if(sendTam == -1){
            perror("\nError"); //si es -1 manda error de envio de trama
            exit(0);//salimos
        } else perror("\nExito al enviar"); //se envia correctamente
}
```

```c
int main(){

    int rawSocket, index; //variable para el socket crudo y el indice de la interfaz
    char opc = 's';

    printf("\n\t\t\nEnviando una trama\n\n");

    /*
        primer parametro: familia a trabajar
        segundo parametro: SOCK_RAW: para trabajar en la capa de enlace de datos
        tercer parametro: colocamos todos los protocolos
        socket Devuelve un valor entero
    */
    rawSocket = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL));

    if(rawSocket == -1){
        perror("\nError al abrir el socket"); //si no se puede abrir un socket manda error
        exit(0);    //salimos del programa
    } else{
        perror("\nExito al abrir el socket");
        index = obtenerDatos(rawSocket); //Aqui obtenemos el indice de nuestra interfaz, ademas de la MAC

        //usamos un while para mandar tramas si el usuario desea
        while( opc == 's'){
            estructuraTrama(sentTrama);   //Le damos la estructura a la trama
            EnviarTrama(rawSocket,index,sentTrama);  //enviamos nuestra trama
            printf("\nInserte una s para seguir mandando tramas, de lo contrario inserte una n: ");
            scanf("%c", &opc);
        }
    }

    close(rawSocket); //cerramos nuestro socket
    printf("\n"); //salto de linea para mejorar legibilidad
    return 0; //se termina el programa
}
```

Imagen 6. Código del programa para enviar tramas.

En este código se hizo una mejora, esta fue implementar un ciclo while que permita al usuario enviar tramas hasta que este ya no quiera mandar otra trama, de esta forma se evita el estar ejecutando a cada rato el programa para enviar una trama, además con esto el usuario escribe el mensaje que quiere incluir en la trama. esto nos ayudó para analizar de mejor manera las tramas enviadas usando wireshark.

2.2 CON LA AYUDA DEL WIRESHARK CAPTURAR LA TRAMA Y EXPLICARLA CON CAPTURAS DE PANTALLA,

LA TRAMA CAPTURADA FUE:

```
0000  ff ff ff ff ff ff 08 00  27 69 7a 5f 0c 0c 53 6f   ........ 'iz_..So
0010  6d 6f 73 20 65 6c 20 53  69 75 75 54 65 61 6d 20   mos el S iuuTeam
0020  79 20 65 73 74 61 20 65  73 20 6c 61 20 70 72 61   y esta e s la pra
0030  63 74 69 63 61 20 35 0a  00 00 00 00              ctica 5· ····
```

Imagen 7. Trama capturada

Vemos que el primer dato es la mac de broadcast indicada en el programa.

```
ff ff ff ff ff ff
```

Imagen 8. MAC broadcast

Después le sigue la dirección origen.

```
08 00  27 69 7a 5f
```

Imagen 9. Dirección origen

Le sigue el ethertype, en este caso fue el definido en el programa:

```
0c 0c
```

Imagen 10. Ethertype

Por último tenemos los datos de la trama:

```
ff ff ff ff ff ff 08 00  27 69 7a 5f 0c 0c 53 6f
6d 6f 73 20 65 6c 20 53  69 75 75 54 65 61 6d 20
79 20 65 73 74 61 20 65  73 20 6c 61 20 70 72 61
63 74 69 63 61 20 35 0a  00 00 00 00
```

Imagen 11. Datos trama

2.3 TODA LA INFORMACIÓN QUE REPORTE EL WIRESHARK.

A continuación se presenta la trama enviada y capturada por wireshark:

```
448 658.105935199 PcsCompu_69:7a:5f    Broadcast           0x0c0c      60 Ethernet II

▸ Frame 448: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▾ Ethernet II, Src: PcsCompu_69:7a:5f (08:00:27:69:7a:5f), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ▸ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▸ Source: PcsCompu_69:7a:5f (08:00:27:69:7a:5f)
    Type: Unknown (0x0c0c)
▸ Data (46 bytes)

0000  ff ff ff ff ff ff 08 00  27 69 7a 5f 0c 0c 53 6f   ........ 'iz_··So
0010  6d 6f 73 20 65 6c 20 53  69 75 75 54 65 61 6d 20   mos el S iuuTeam
0020  79 20 65 73 74 61 20 65  73 20 6c 61 20 70 72 61   y esta e s la pra
0030  63 74 69 63 61 20 35 0a  00 00 00 00               ctica 5· ····
```

Imagen 12. Trama enviada y capturada en wireshark

Como vemos se despliega toda la información de la trama, analicemosla primeramente por el apartado de frame:

```
Frame 448: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▾ Interface id: 0 (enp0s3)
    Interface name: enp0s3
  Encapsulation type: Ethernet (1)
  Arrival Time: Oct 15, 2021 14:51:17.858887486 CDT
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1634327477.858887486 seconds
  [Time delta from previous captured frame: 1.258949287 seconds]
  [Time delta from previous displayed frame: 1.258949287 seconds]
  [Time since reference or first frame: 658.105935199 seconds]
  Frame Number: 448
  Frame Length: 60 bytes (480 bits)
  Capture Length: 60 bytes (480 bits)
  [Frame is marked: False]
  [Frame is ignored: False]
  [Protocols in frame: eth:ethertype:data]
  [Coloring Rule Name: Broadcast]
  [Coloring Rule String: eth[0] & 1]
```

Imagen 13. Apartado Frame de la trama

En esta parte primeramente nos muestra el id de nuestra interfaz, asi como el nombre, en este caso es la enp0s3 y datos generales sobre el frame como lo son:

- Tipo de encapsulamiento
- Tiempo de llegada
- La longitud del frame, en este caso es de 60, ya que así fue establecido en el código del programa

Ahora veamos la parte de ethernet ii:

```
Ethernet II, Src: PcsCompu_69:7a:5f (08:00:27:69:7a:5f), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▾ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
    Address: Broadcast (ff:ff:ff:ff:ff:ff)
    .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory default)
    .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)
▾ Source: PcsCompu_69:7a:5f (08:00:27:69:7a:5f)
    Address: PcsCompu_69:7a:5f (08:00:27:69:7a:5f)
    .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Type: Unknown (0x0c0c)
```

Imagen 14. Parte de la trama Ethernet II

En esta parte se captura la dirección destino, en este caso fue la de broadcast que colocamos en el programa, de igual forma se captura la fuente, la cual como se ve, es la mac de nuestra computadora y nos muestra el ethertype.

Por último tenemos la parte de data:

```
▾ Data (46 bytes)
    Data: 536f6d6f7320656c20536975755465616d20792065737461...
    [Length: 46]
```

Imagen 15. Parte de la trama Data.

En esta parte se muestran los datos y la longitud de los datos, en este caso 46.

2.4 INCLUYE LA CAPTURA DE PANTALLA Y EXPLIQUE EL RESULTADO DE EJECUTAR EL PROGRAMA.

```
root@BDMV:/media/root/Mas espacio/Redes 1/Practicas/Practica 5# gcc enviotram.c -o env
root@BDMV:/media/root/Mas espacio/Redes 1/Practicas/Practica 5# ./env


Enviando una trama


Exito al abrir el socket: Success

Inserta el nombre de la interfaz de red: enp0s3

 El indice es: 2
 La MAC es: 08: 00: 27: 69: 7a: 5f:

Ingrese el mensaje de la trama: hola soy una trama

Exito al enviar: Success

Inserte una s para seguir mandando tramas, de lo contrario inserte una n: s

Ingrese el mensaje de la trama: soy otra trama

Exito al enviar: Success

Inserte una s para seguir mandando tramas, de lo contrario inserte una n: n
```

Imagen 16. Ejecución del programa

Al ejecutar el programa tenemos lo de la imagen anterior, primero nos solicita el nombre de la interfaz de red para obtener el índice y la MAC, posteriormente nos indica el mensaje que contendrá la trama y enviarlo.

Cuando se envía nos muestra un mensaje para volver a enviar una trama o terminar el programa.

3. CONCLUSIONES INDIVIDUALES DE CADA PARTICIPANTE DEL EQUIPO

FISCHER SALAZAR CÉSAR EDUARDO

La realización de esta práctica nos presenta un envió diferente de trama al que anteriormente habíamos utilizado, ya que en esta nosotros definimos la trama además de poder ingresar el mensaje que nosotros queremos que se envié en conjunto. Pudimos corroborar que lo hicimos él envió de información de manera correcta mediante la captura de trama para un posterior análisis de esta que efectuamos mediante el Wireshark.

Con respecto al programa anterior se me hizo un más fácil el entender cómo se hacia la consulta de datos, pero en este me costó entender un poco le definir la trama para este mensaje, después de un poco de juego con el programa se me hizo más entendible e interesante.

LÓPEZ GARCÍA JOSÉ EDUARDO

Con la ayuda de la práctica realizada, rodeando al tópico del envío de tramas, pude comprender más entendiblemente cómo es que se realizan estos envíos por medio de la red, de tal manera que se pueda definir una estructura y añadirle un pequeño mensaje. Con la ventaja que ofrece el software Wireshark, se hizo un análisis de lo que había capturado, de esta manera comprobando que se haya realizado el envío de forma correcta.

De igual manera, siento que se entendió bastante claro la manera en que se estructuran las tramas de una red, para poder checar distintos parámetros para asegurarnos de que se encuentren en forma, y que no haya ocurrido algún siniestro durante su envío.

MEZA VARGAS BRANDON DAVID

Gracias a esta práctica comprendí de mejor manera como se hace el envió de tramas por la red, pues estructuramos desde cero una trama, además de adjuntarle nuestros propios datos que fue en este caso un mensaje. Dicha trama la analizamos en wireshark para ver que la enviamos de forma correcta.

Personalmente me gustó mucho esta práctica, pues me parece muy interesante como con la función sendto, podemos enviar esta información y no solo un simple mensaje como en prácticas anteriores, sino que podemos estructurar nuestra propia trama y mandarla a través de la red. Con esto me puedo dar una idea de como el gran tráfico que pasa por la red está estructurado, en clases anteriores lo habíamos visto de manera algo teórica y con esta práctica quedo mucho más reforzado al ver nuestra propia trama.

Finalmente, creo que es importante saber cómo una trama está estructurada, pues a partir de esto podemos jugar modificando valores y ver como es enviada la trama y verificar si existió un error o algo raro.