PRÁCTICA 8 – PROTOCOLO ARP

FECHA: 25/11/2021

NOMBRE DEL EQUIPO: EL SIUUU TEAM

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

UNIDAD ACADÉMICA: REDES DE COMPUTADORAS

INTRODUCCION

Como se ha venido realizando desde prácticas anteriores, se ha estado manejando el empleo de los sockets crudos en Linux, así como contar con la ayuda del manual que está incluido en este sistema operativo, con el propósito de poder codificar programas que nos permitan enviar y recibir tramas situados en la red o en nuestro propio dispositivo.
En esta ocasión, estaremos visualizando con la herramienta de Wireshark el envío y llegada de tramas que incluyen el protocolo ARP, el cual se ha mencionado con anterioridad que funciona cuando se asignan direcciones IP usando este protocolo y distingue si la IP del host de destino se encuentra en la misma subred o en otra, esto para que pueda llevar a cabo una revisión de la máscara de la subred. Así, con los recursos que ya se han generado a través de las prácticas anteriores, y los ajustes solicitados para mostrar el cambio de color en las tramas, intentaremos observar a detalle cada una de las tramas que llegan al ordenador y poder darle la interpretación correcta a dichos elementos capturados.

OBJETIVOS

OBJETIVO PRINCIPAL: ENVIAR UNA TRAMA ARP Y CAPTURARLA CON WIRE SHARK LINUX

OBJETIVO SECUNDARIO. IMPRIMIR LA TRAMA ARP (SOLICITUD Y RESPUESTA) EN CONSOLA E INTERPRETAR LAS TRAMAS QUE RECIBIMOS CON EL PROGRAMA

Manuales man socket , man 7 ip, man packet
Compilador de c

RECURSOS NECESARIOS PARA REALIZAR LA PRÁCTICA

Terminal de Linux
Navegador de internet

PARTE 1: DIAGRAMA DE FLUJO

- **INCLUYE DIAGRAMA DE FUJO DEL PROGRAMA**

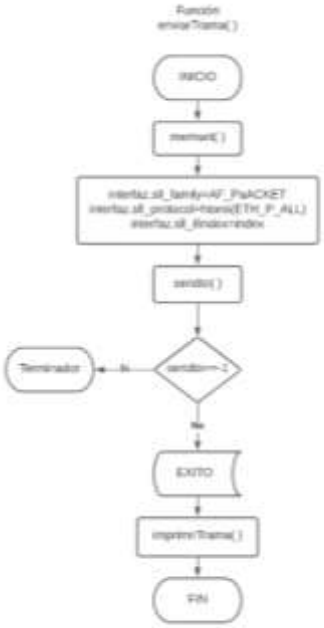

Imagen 1. Diagrama de flujo de función enviarTrama( )

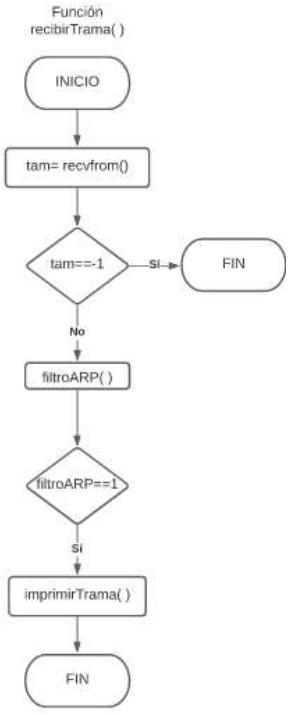

Imagen 2. Diagrama de flujo de función recibirTrama( )

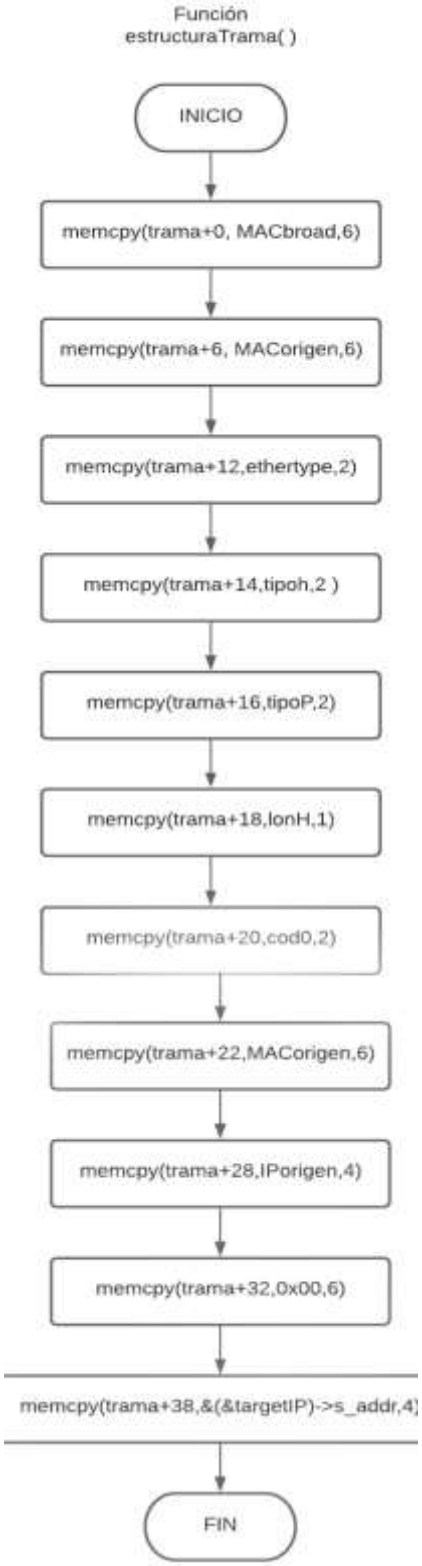

Imagen 3.Diagrama de flujo de función estructuraTrama( )

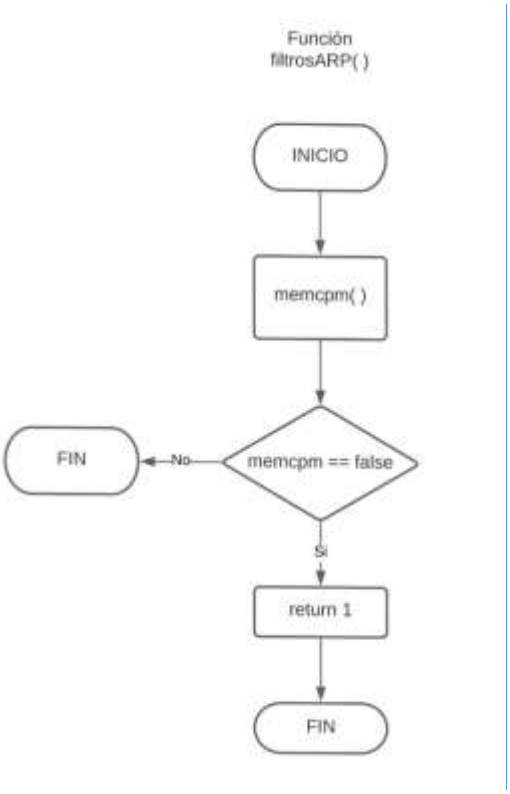

Imagen 4.Diagrama de flujo de función filtros ARP( )

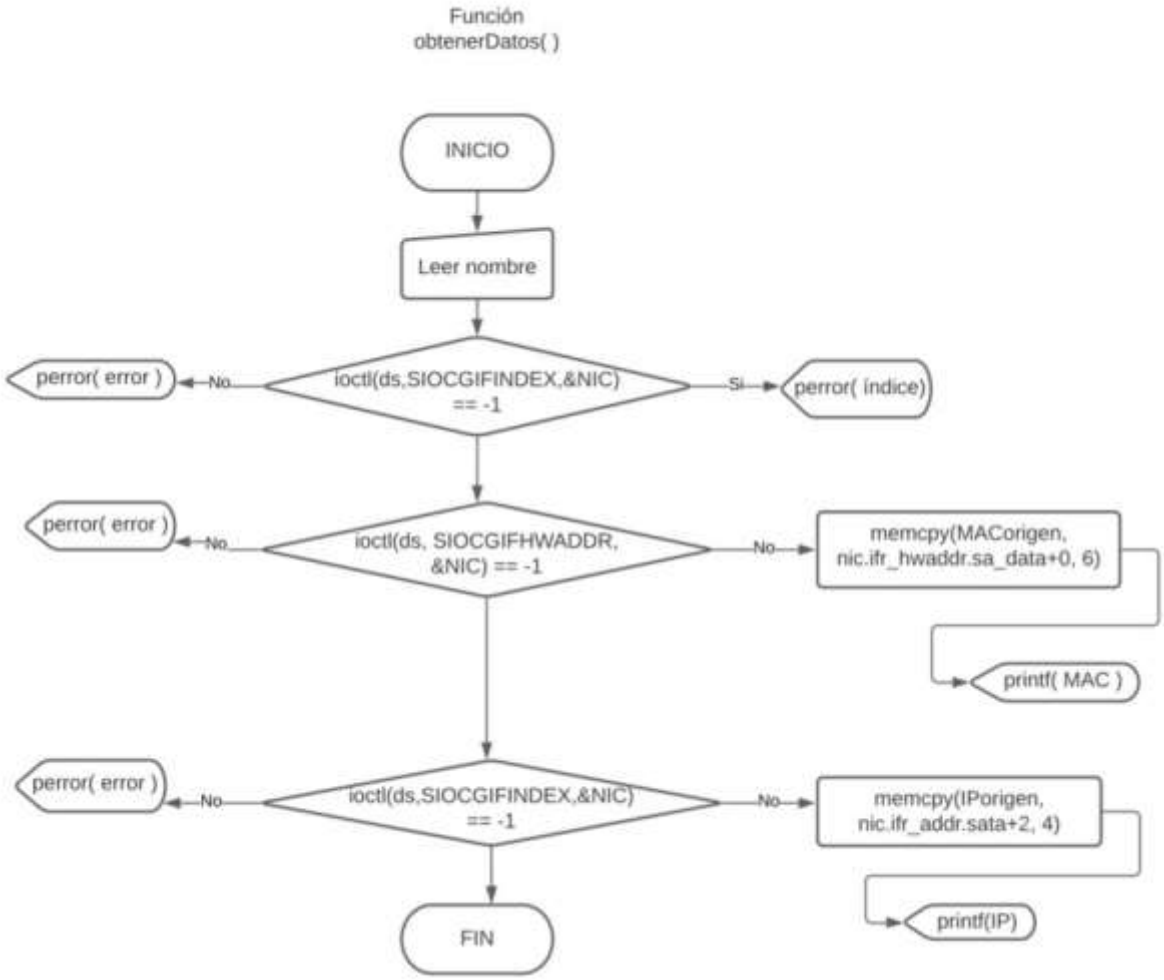

Imagen 5. Diagrama de flujo de función obtenerDatos()

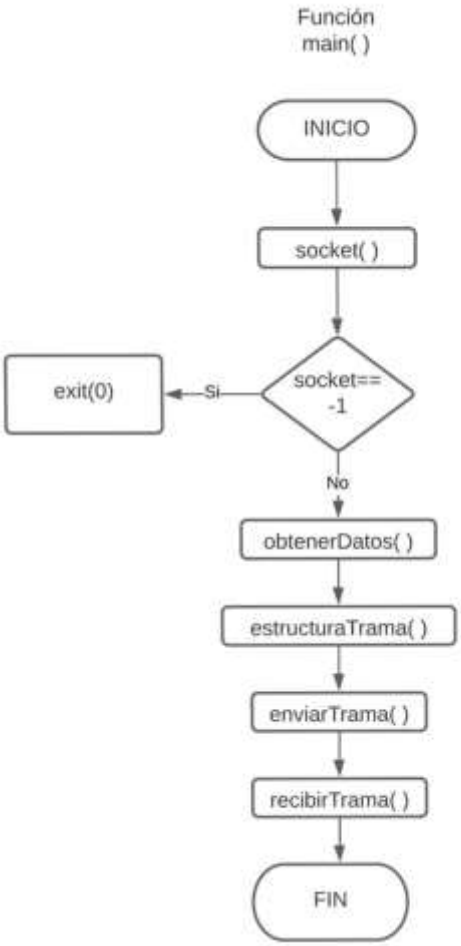

Imagen 6.Diagrama de flujo de función estructuraTrama( )

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. (LINEA ALINEA)

A continuación, vemos las capturas del código utilizado comentado y explicado línea por línea.

```c
#include <sys/socket.h>
#include <linux/if_packet.h>
#include <net/ethernet.h> /* the L2 protocols */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <string.h>
#include <sys/ioctl.h>
#include <net/if.h>
#include <netinet/in.h>

/*
    *FUnciones para colores
*/
void red(){
    printf("\033[1;31m");
}
void reset(){
    printf("\033[0m");
}
void yellow(){
    printf("\033[1;33m");
}
void blue(){
    printf("\033[1;34m");
}
void green(){
    printf("\033[1;32m");
}
void white(){
    printf("\033[1;37m");
}
void black(){
    printf("\033[1;30m");
}
void purple(){
    printf("\033[1;35m");
}
```

```c
void cyan(){
    printf("\033[1;36m");
}

unsigned char MACorigen[6],MACdestino[6];
unsigned char MACbroad[6]={0xff,0xff,0xff,0xff,0xff,0xff};//*longitud de 6 bytes e inicializamos para broadcast
//solicitud de ARP
unsigned char ethertype[2]={0x08, 0x06};        //*2 bytes para ethertype y se inicializa en ARP
unsigned char tramaEnv[1514], tramaRec[1514];   //*Para la trama enviada y recibida
unsigned char IPorigen[4];                      //*Ip origen
unsigned char opcodeRes[2]={0x00,0x02};         //*Código de operacion de respuesta ARP

struct in_addr targetIP;                        //*Estructura necesaria para inet_aton()

/*
    *Esta funcion es para imprimir la trama recibida.
    *Recibe el paquete (la trama) recibida y el tamaño de esta
*/
void imprimirTrama(unsigned char *paq, int len){
    int i;
    /*
        *El siguiente for es para imprimir la trama, vemos que va desde i hasta el tamaño
        *de la trama recibida
    */
    for(i=0; i<len; i++){
        if( i < 6 )                     //*coloreamos de rojo la direccion mac origen
            red();
        if( i >= 6 && i <= 11 )         //*coloreamos de amarillo la direccion mac origen
            yellow();
        if( i >= 12 && i <= 13 )        //*coloreamos de azul el ethertype
            blue();
        if( i >= 14 && i <= 15 )        //*coloreamos de verde el tipo ethernet
            green();
        if( i >= 16 && i <= 17 )        //*coloreamos de negro el tipo de protocolo
            black();
        if( i == 18 )                   //*coloreamos de cyan la longitud de direcciones de hardware
            cyan();
        if( i == 19 )                   //*Coloreamos de morado la longitud de protocolo
            purple();
        if( i >= 20 && i <= 21 )        //*coloreamos de rojo el código de operación
            red();
        if( i >= 22 && i <= 27 )        //*coloreamos de amarillo la MAC origen
            yellow();
        if( i >= 28 && i <= 31 )        //*coloreamos de verde la IP origen
            green();
        if( i >= 32 && i <= 37 )        //*coloreamos de morado la MAC destino
            purple();
        if( i >= 38 && i <= 41 )        //*coloreamos de morado la IP destino
            cyan();
        if( i >= 42)                    //*coloreamos de blanco los bits de relleno de la trama
            white();

        /*
            *El siguiente if es para darle formato parecido al de wireshark, una vez se hayan impreso 16 caracteres
            *salta a la siguiente linea y continua con la impresión de los digitos
        */
        if(i%16==0)
            printf("\n");       //*salto de lineal para hacer el formato de 16 en 16
        printf("%.2x ", paq[i]);//*imprimimos los caracteres de dos en dos en formato hexadecimal
```

```
        reset(); //*Reseteamos los colores
    }
    red();
    printf("\n\nDireccion MAC origen");
    yellow();
    printf("\nDireccion MAC fuente");
    blue();
    printf("\nEthertype");
    green();
    printf("\nTipo de ethernet");
    black();
    printf("\nTipo de protocolo");
    cyan();
    printf("\nLongitud de direcciones de hardware");
    purple();
    printf("\nLongitud de protocolo");
    red();
    printf("\nCodigo de operación");
    green();
    printf("\nIP origen");
    purple();
    printf("\nMAC destino");
    cyan();
    printf("\nIP destino");
    white();
    printf("\nBits de relleno");


    reset();//*Reseteamos los colores
    //saltos de linea para mejorar legibilidad de la salida
    printf("\n");
    printf("\n");
}
```

```c
/*
    *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){
    char IP[50];
    printf("\nInserta la direccion IP:");       //  Pide una IP
    scanf("%s", IP);
    // fgets(IP);


    /*
        *Con inet_aton, convertimos la dirección IP de la notación estandar de números y puntos
        *a la representación binaria, la guardamos en la estructura IPdestino
    */
    inet_aton(IP,&targetIP);

    unsigned char tipoh[2]={0x00,0x01};     //*Tipo de hardware Ethernet
    unsigned char tipoP[2]={0x08,0x00};     //*Tipo de protocolo
    unsigned char lonH[1]={6};              //*longitud de las direcciones de hardware
    unsigned char longP[1]={4};             //*longitud de protocolo
    unsigned char codO[2]={0x00,0x01};      //*codigo de operacion de peticion ARP

    //*se estructura la trama

    //encabezado MAC
    memcpy(trama+0, MACbroad, 6);   //*copiamos en las primeras 6 posiciones la mac de broadcast
    memcpy(trama+6, MACorigen, 6);  //*copiamos a partir de la posicion 6 la mac origen
    memcpy(trama+12, ethertype,2);  //*posteriormente copiamos los bytes ppara ethertype
    //Mesaje de ARP

    memcpy(trama+14,tipoh,2);       //*Copiamos los valores del ethernet
    memcpy(trama+16,tipoP,2);       //*COpiamos el protocolo que usa los servicios ARP
    memcpy(trama+18,lonH,1);        //*Copiamos la longitud de direcciones de hardware
    memcpy(trama+19,longP,1);       //*Copiamos la longitud de direcciones
    memcpy(trama+20,codO,2);        //*Copiamos el codigo de operacion
    memcpy(trama+22, MACorigen, 6); //*Copiamos a partir de la posicion 22 los valores para la MACOrigen
    memcpy(trama+28, IPorigen, 4);  //*Copiamos a partir de la posicion 28 los valores para la IPOrigen
    memset(trama+32, 0x00, 6);
    memcpy(trama+38,&(&targetIP)->s_addr, 4);//*Copiamos a partir de la posicion 28 los valores para la IPDestino
}

/*
    *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 ds, int index, unsigned char *trama){

    int tam;
    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=index;             //*el indice será el obtenido previamente

    /*
        *Aqui mandamos la trama haciendo uso de la función sendto, con una tamaño de 60
    */
    tam=sendto(ds, trama, 60, 0, (struct sockaddr *)&interfaz, sizeof(interfaz));
```

```
    if(tam==-1){
        perror("\nError al enviar");
        exit(0);
    }
    else{
        perror("\nexito al enviar");
        printf("\nTrama enviada (Solicitud ARP)");
        printf("\n");
        imprimirTrama(trama,60);            //*Imprimimos la trama enviada
    }
}


/*
    *Función para filtros al recibir la trama de la respuesta ARP
*/
int filtrosARP(unsigned char *paq)
{
 unsigned char etherARP[2]={0x08,0x06};   //*ethertype de tipo ARP

 /*
    *EL siguiente if nos sirve para determinar que tramas vamos a recibir, recibiremos tramas que contengan mi dirección
    *que su ethertype sea ARP, ademas que el codigo de operación sea el 0002 y que tengan la ip destino que ingreso el us
    *al estructurar la trama
 */
 if(!memcmp(paq+0,MACorigen,6)&&!memcmp(paq+12,etherARP,2)&&!memcmp(paq+20,opcodeRes,2)&&!memcmp(paq+28,&(&targetIP)->s_
      return 1;    //  *REtornamos 1 si se cumplen los filtros
 else
     return 0;
}
```

```
void recibirTrama(int ds, unsigned char *trama){
    /*
        *En esta caso la estructura será la tarjeta de red desde la cual recibiremos
        *colocamos NULL ya que no nos importará desde cual tarjeta vamos a recibir, igual
        *debido a esto colocamos un 0 al final.
        *Usamos un ciclo infinito para recibir tramas en loop.
    */
    while(1){
        int tam;
        tam=recvfrom(ds, trama, 60, 0, NULL, 0);
        if(tam==-1){
            perror("\nError al recibir");
            exit(0);
        }
        else{
            /*
                *En el siguiente if hacemos una comparación para solo capturar tramas
                *que cumplan con los filtros establecidos
            */

            if(filtrosARP(trama))
            {
                printf("\nTrama recibida (Respuesta ARP)");
                printf("\n");
                imprimirTrama(trama, tam);  //  *Imprimimos la trama recibida
                break;
            }
        }
    }

}
```

```c
/*
    *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 ds){
    struct ifreq nic;           //*estructura ifreq para consultar datos de nuestra interfaz, man netdevice
    unsigned char nombre[20];   //*variable que alamcenara el nombre de nustra interfaz
    int i, index;               //*variable apra loop y para el indice

    printf("Inserta el nombre de la interfaz: ");
    scanf("%s", nombre);        //*leemos el nombre de la interfaz de red

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

     /*
        *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(ds, SIOCGIFINDEX, &nic)==-1){
        perror("\nerror al obtener el index");
    }
    else{
        index=nic.ifr_ifindex;              //*almacenamos el indice en nuestra variable
        printf("El indice es %d", index);   //*imprimimos nuestro indice de red}

        //*obtener la MAC usando SIOCGIFHWADDR
        if(ioctl(ds, SIOCGIFHWADDR, &nic)==-1){
            perror("\nErro al obtener la MAC");
            exit(0);
        }
        else{
            //*si obtenemos la mac la almacenamos en nuestra variable global MACorigen,
            memcpy(MACorigen, nic.ifr_hwaddr.sa_data+0, 6);
            printf("\nLa 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:", MACorigen[i]);


            //*obtener la IP usando SIOCGIFADDR
            if(ioctl(ds, SIOCGIFADDR, &nic)==-1){
            perror("\nErro al obtener la IP");
            exit(0);
              }
           else{
            //*Guardamos la IP oprigen en  nuesyta variable IPorigen
            memcpy(IPorigen, nic.ifr_addr.sa_data+2, 4);
            printf("\nLa IP es: ");

            //  *Imprimimos nuestra IP
            for(i=0; i<4; i++)
                printf("%d.", IPorigen[i]);

        }
        }
    }
    return index;
}
```

```c
int main(){
    int packet_socket, indice;
    packet_socket = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL)); //*abrimos nuestro socket

    if(packet_socket==-1){
        perror("Error al abrir el socket");
        exit(0);
    }
    else{
        perror("\nExito al abrir el socket\n");
        indice=obtenerDatos(packet_socket); //*se le manda el descriptor de socket para obtener datos


        printf("\n\n");

        estructuraTrama(tramaEnv);                      //*Damos estructura a la trama a enviar
        enviarTrama(packet_socket, indice, tramaEnv);   //*Enviamos la trama
        recibirTrama(packet_socket, tramaRec);          //*Recibimos la trama
    }

    close(packet_socket); //*cerramos nuestro socket

    return 0;
}
```

2.2 EJECUTAR TOMAR CAPTURA DE PANTALLA DE CADA ETAPA DEL PROGRAMA.

Primeramente, se puede ver la parte donde el programa obtiene los datos de nuestra interfaz de red a partir del nombre de nuestra interfaz y espera a que el usuario ingrese una IP destino.

```
Exito al abrir el socket
: Success
Inserta el nombre de la interfaz: enp0s3
El indice es 2
La MAC es: 08:00:27:69:7a:5f:
La IP es: 10.0.2.15.


Inserta la direccion IP:
```

Al ingresar una IP, se imprime la trama enviada identificando cada parte de la trama por colores especificando cada parte en texto.

```
Inserta la direccion IP:10.0.2.2

exito al enviar: Success

Trama enviada (Solicitud ARP)

ff ff ff ff ff ff 08 00 27 69 7a 5f 08 06 00 01
08 00 06 04 00 01 08 00 27 69 7a 5f 0a 00 02 0f
00 00 00 00 00 00 0a 00 02 02 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00

Direccion MAC origen
Direccion MAC fuente
Ethertype
Tipo de ethernet
Tipo de protocolo
Longitud de direcciones de hardware
Longitud de protocolo
Codigo de operación
IP origen
MAC destino
IP destino
Bits de relleno
```

Posteriormente se imprime la trama recibida que será la respuesta de ARP obtenida con el mismo formato que la trama enviada

```
Trama recibida (Respuesta ARP)

08 00 27 69 7a 5f 52 54 00 12 35 02 08 06 00 01
08 00 06 04 00 02 52 54 00 12 35 02 0a 00 02 02
08 00 27 69 7a 5f 0a 00 02 0f 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00

Direccion MAC origen
Direccion MAC fuente
Ethertype
Tipo de ethernet
Tipo de protocolo
Longitud de direcciones de hardware
Longitud de protocolo
Codigo de operación
IP origen
MAC destino
IP destino
Bits de relleno
```

2.3 INCLUYE LA CAPTURA DE PANTALLA DE LAS TRAMAS Y EXPLIQUE LOS DATOS RECIBIDOS EN CADA PARTE DEL PROGRAMA

Trama enviada

```
Trama enviada (Solicitud ARP)

ff ff ff ff ff ff 08 00 27 69 7a 5f 08 06 00 01
08 00 06 04 00 01 08 00 27 69 7a 5f 0a 00 02 0f
00 00 00 00 00 00 0a 00 02 02 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00
```

Primeramente, se imprime la trama enviada, esta es una petición de broadcast, esto lo podemos saber por la dirección de broadcast al inicio, posteriormente de la dirección de broadcast se imprime la MAC origen, en este caso la de nuestra computadora, posteriormente sigue el código de operación, en este caso es 0806, el cual indica que Estanis haciendo uso de ARP, después va el tipo de ethernet que es IP, seguido del tipo de protocolo, después siguen las longitudes de direcciones y de protocolo. después vuelve a aparecer nuestra dirección MAC origen seguida de la IP origen, finalmente aparece la MAC destino en este caso no hay y la IP destino que es la que ingreso el usuario.

Trama recibida:

```
Trama recibida (Respuesta ARP)

08 00 27 69 7a 5f 52 54 00 12 35 02 08 06 00 01
08 00 06 04 00 02 52 54 00 12 35 02 0a 00 02 02
08 00 27 69 7a 5f 0a 00 02 0f 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00
```

La trama recibida tiene el mismo formato anteriormente explicado, el detalle aquí es que al inicio aparece la MAC origen de la respuesta, siendo esta nuestra MAC. de igual forma cambia nuestra MAC origen, además de que ahora la IP origen se convierte en nuestra IP destino y viceversa.

2.4 MODIFIQUE EL PROGRAMA PARA CAMBIAR DE COLOR CADA PARTE DE LA TRAMA (SOLICITUD Y RESPUESTA) QUE RECIBIMOS AGREGUE CAPTURAS DE PANTALLA DE LA TRAMA CAPTURADA Y EXPLIQUE LOS DATOS.

Estos cambios ya se realizaron anteriormente, las capturas mostradas anteriormente muestran cómo se realizó y los resultados de las tramas y los datos.

3. CONCLUSIONES INDIVIDUALES DE CADA PARTICIPANTE DEL EQUIPO

FISCHER SALAZAR CÉSAR EDUARDO

Gracias a la implementación del código de la presente practica pude comprender claramente el funcionamiento del protocolo ARP, ya que pudimos realizar una consulta mediante la estructuración de una trama la cual nos permitiera conocer la MAC perteneciente a la dirección IP indicada dentro de nuestra red de área local, dicha trama fue identificada con un código de color que nos permitiera distinguir los elementos importantes de los cuales está conformada, tales como la IP de origen y de destino , la dirección MAC de destino y el tipo de protocolo que fue utilizado. Una vez terminada la ejecución del programa pudimos verificar su funcionamiento correcto mediante la herramienta wireshark la cual nos mostró que los datos de la consulta y de la respuesta eran los mismo a los que nos arrojaba el programa.

Puedo concluir que esta práctica fue algo interesante ya que nos permite conocer más sobre el cómo funciona la comunicación entre nuestras computadoras.

LÓPEZ GARCÍA JOSÉ EDUARDO

Por medio de la implementación de este código, pudimos darnos cuenta de que esta práctica del protocolo ARP tiene una similitud/unión considerable de lo que hemos visto en las anteriores prácticas de envío y recibo de tramas de red, ahí se han empleado los mismos recursos del manual, el wireshark y los sockets para poder ver la trama enviada o recibida, con el detalle que esta corresponde a una respuesta del protocolo ARP. Aquí la implementación de los colores en la identificación de elementos de la trama ayudó a que pudieran distinguirse unos de otros, si era la MAC, la IP de origen o destino, del hardware, o si se trataban de bits de relleno.
Por tanto, puede decir que he comprendido de forma más clara cómo es que el ARP interviene en la resolución de las direcciones (tal como su nombre en inglés dice), para que así ayude a la vinculación de la IP con la MAC y exista un reconocimiento entre los sistemas.

MEZA VARGAS BRANDON DAVID

Esta práctica fue la combinación de dos prácticas anteriores, la de envió de una trama y la de recibir una trama, en esta hubo una particularidad, pues la trama enviada fue estructurada como una petición ARP, por lo tanto, la trama recibida fue una respuesta de ARP.
Gracias a esta práctica logré comprender de mejor forma el total funcionamiento del protocolo ARP, pues al estructurar una petición de ARP los conceptos que hemos visto a lo largo de las clases quedaron entendidos y aterrizados en la práctica con la realización del programa de la práctica actual.
Concluyendo así que la importancia del protocolo ARP radica en asociar dirección IP a una dirección MAC que se reconozca en la red local, en este caso de nuestra computadora y nuestra red de internet.