INSTITUTO POLITÉCNICO NACIONAL

ESCOM

Redes

PRÁCTICA 3

"Analizador de protocolo LLC"

INTEGRANTES:

González Mora Erika Giselle

Olivares Ménez Gloria Oliva

GRUPO: 2CV16

Índice

Índice de figuras

1. Introducción

El protocolo LLC (control lógico de enlace) es un protocolo de capa de enlace de datos derivado de HDLC, del cual hereda su campo de control, y fue estandarizado por la IEEE bajo la denominación 802.2. Igual que en HDLC se tienen tramas de información, supervisión y no numeradas distinguiéndose entre ellas por los bits menos significativos de su campo de control.

Es la más alta de las dos subcapas de enlace de datos definidas por el IEEE y la responsable del control de enlace lógico. La subcapa LLC maneja el control de errores, control del flujo, entramado, control de diálogo y direccionamiento de la subcapa MAC. El protocolo LLC más generalizado es IEEE 802.2, que incluye variantes no orientadas a conexión y orientadas a conexión.

1.1. Subcapa LLC

En la subcapa LLC se contemplan dos aspectos bien diferenciados:

1.1.1. Los protocolos

Los protocolos LLC: Para la comunicación entre entidades de la propia subcapa LLC, definen los procedimientos para el intercambio de tramas de información y de control entre cualquier par de puntos de acceso al servicio del nivel de enlace LSAP.

1.1.2. Las interfaces

Las interfaces: con la subcapa inferior MAC y con la capa superior (de Red).

- **Interfaz LLC – MAC:** Especifica los servicios que la subcapa de LLC requiere de la subcapa MAC, independientemente de la topología de la subred y del tipo de acceso al medio.
- **Interfaz LLC – Capa de Red Modelo OSI:** Especifica los servicios que la Capa de Red Modelo OSI obtiene de la Capa de Enlace Modelo OSI, independientemente de su configuración.

La subcapa IEEE 802.2 agrega información de control al mensaje creado por la capa superior y pasado a LLC para su transmisión a otro nodo en el mismo enlace de datos. El paquete resultante se denomina generalmente unidad de datos de protocolo LLC (PDU) y la información adicional agregada por la subcapa LLC es el encabezado LLC . El encabezado LLC consta de DSAP (punto de acceso al servicio de destino), SSAP (punto de acceso al servicio de origen) y el campo de control . Los dos campos de 8 bits DSAP y SSAP permiten la multiplexación de varios protocolos de capa superior por encima de LLC. Sin embargo, muchos protocolos usan la extensión Subnetwork Access Protocol (SNAP) que permite usar valores EtherType para especificar el protocolo que se transporta sobre IEEE 802.2. También permite a los proveedores definir sus propios espacios de valores de protocolo. El campo de control de estilo HDLC de 8 o 16 bits sirve

para distinguir el modo de comunicación, para especificar una operación específica y para facilitar el control de la conexión y el control del flujo (en el modo de conexión) o los reconocimientos (en el modo sin conexión reconocido).

2. Desarrollo

2.1. Explicación Código

En esta práctica se requería la elaboración de un Analizador de Protocolo LLC. Para esto se necesitaba tener varias tramas como ejemplo para poder realizar el análisis correspondiente.

Dichas tramas se desarrollaron con la ayuda del programa llamado LLC.c agregado en la sección de Anexos.

Una vez que se comprobó el funcionamiento del programa de apoyo, se procedió a verificar las tramas.

Lo primero que se necesitaba saber era su longitud en bytes, y esto, daría a conocer si se trataba de una trama IEEE802.3 o de una trama Ethernet. Una trama IEEE802.3, debe tener un valor menor a (1500)10, contrario a la trama Ethernet.

La comprobación de la longitud se debe realizar en el campo Tipo/Longitud de la trama. En la siguiente figura se muestra la estructura de una trama IEEE802.3 para que la localización de los campos sea más sencilla.

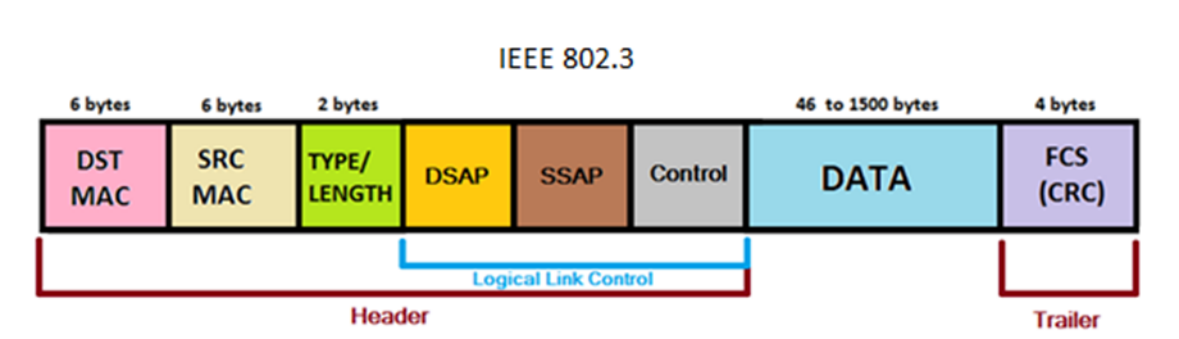


Figura 1. Formato Trama IEEE802.3

La validación de la longitud en código se muestra en a continuación:

```c
unsigned char longitud = (pkt_data[12]*256)+pkt_data[13];
printf("Longitud en hexadecimal: %.2x \n",longitud);
printf("Longitud en decimal: %d \n",longitud);


if(longitud<=1500){
    printf("Es una trama IEEE802.3\n");
}
else if(longitud>1500){
    printf("Es una trama Ethernet\n");
}else{
    printf("Inválido\n");
}
```

Figura 2. Longitud de la trama

Se declara la variable longitud donde se guardará la operación que ayudará a saber el valor del campo Tipo/Longitud. Se utilizan los bits 12 y 13 de la trama que, de acuerdo con la Figura 1, son los que corresponden a dicho campo. Se muestra al usuario su valor en decimal y hexadecimal.

Posteriormente se realiza un condicional if para decidir si es una trama IEEE802.3 o Ethernet.

Los resultados de algunas de las tramas se muestran en la siguiente figura.

```
1395685473:0 (64)
00 02 b3 9c ae ba 00 02 b3 9c df 1b 00 03 f0 f0
7f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 43 05 90 6d


Analisis:
Longitud en hexadecimal: 03
Longitud en decimal: 3
Es una trama IEEE802.3
```

(a)

```
1395685473:1 (64)
00 02 b3 9c df 1b 00 02 b3 9c ae ba 00 03 f0 f1
73 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 54 90 6d


Analisis:
Longitud en hexadecimal: 03
Longitud en decimal: 3
Es una trama IEEE802.3
```

(b)

```
1395685473:2 (64)
00 02 b3 9c ae ba 00 02 b3 9c df 1b 00 04 f0 f0
01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 41 a3 90 6d


Analisis:
Longitud en hexadecimal: 04
Longitud en decimal: 4
Es una trama IEEE802.3
```

(c)

Figura 3. Demostración longitudes. (a) Ejemplo 1, (b) Ejemplo 2, (c) Ejemplo 3.

Ahora se recurre a los campos DSAP (Punto de Acceso al Servicio Destino) y SSAP (Punto de Acceso al Servicio Origen) para determinar si es individual o grupal y si es comando o respuesta. Al observar la Figura 1, podemos notar que se utilizan los bits 14 y 15 respectivamente.

A continuación, se muestra la estructura de estos campos.

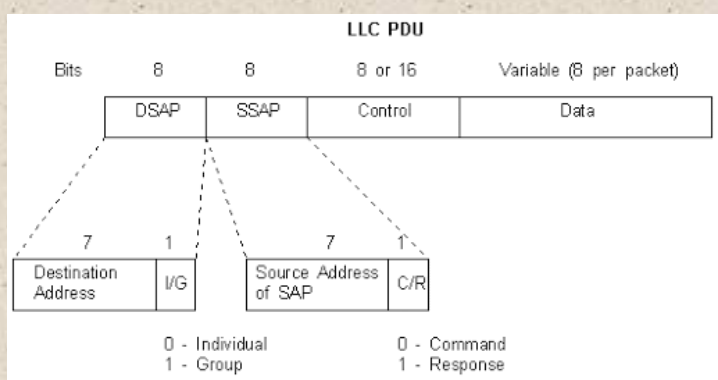


Figura 4. Estructura campos DSAP y SSAP.

Estos campos están asociados a protocolos de nivel superior o aplicaciones que reciben o generaron los datos contenidos en la trama.

En el código se plasma de la siguiente manera:

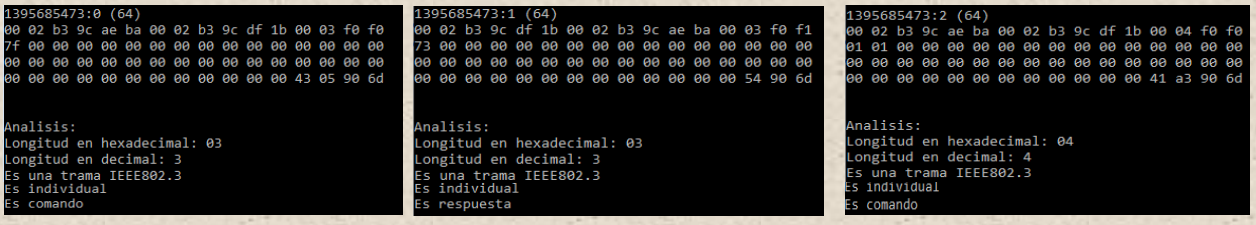
```
unsigned char dsap=pkt_data[14]&0x01;

if(dsap==0){
    printf("Es individual\n");
}else if(dsap==1){
    printf("Es grupal\n");
}

unsigned char ssap=pkt_data[15]&0x01;
if(ssap==0){
    printf("Es comando\n\n");
}else{
    printf("Es respuesta\n\n");
}
```

Figura 5. Código DSAP y SSAP.

Se declaran las variables dsap y ssap para guardar las operaciones que se realizan con sus bits correspondientes (14 y 15). A éstos solamente se le aplica la máscara 0x01 para que sólo obtengamos valores de 0 y 1.

Se realiza de igual forma un condicional if para sacar los datos requeridos.

Para demostrar los resultados, se tomarán las mismas tramas de ejemplo de la Figura 3.



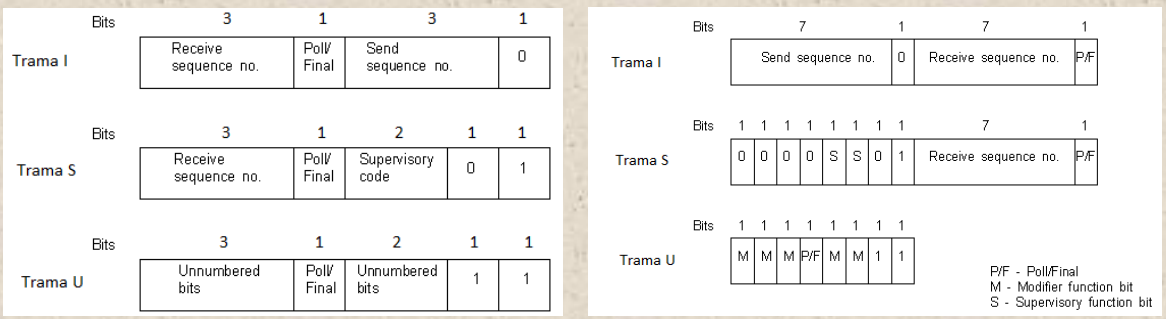
(a) (b) (c)

Figura 6. Demostración DSAP y SSAP. (a) Ejemplo 1, (b) Ejemplo 2, (c) Ejemplo 3.

Lo siguiente a revisar es el campo de control. Puede a ver 2 modos: normal o extendido. En el primer modo se usa 1 bit de campo de control, mientras que en el segundo se usan 2 bits.

Este campo es importante porque también determina qué tipo de mensaje trae la trama (Tipo I, Tipo S o Tipo U).

En la Figura 7 se muestran ambas configuraciones (normal y extendida) de los tipos de mensajes.



(a) (b)

Figura 7. Estructura campo de control. (a) Modo normal, (b) Modo extendido.

Para saber si es modo normal o modo extendido la longitud que se había guardado, se debe averiguar si es mayor o menor o igual a 3.

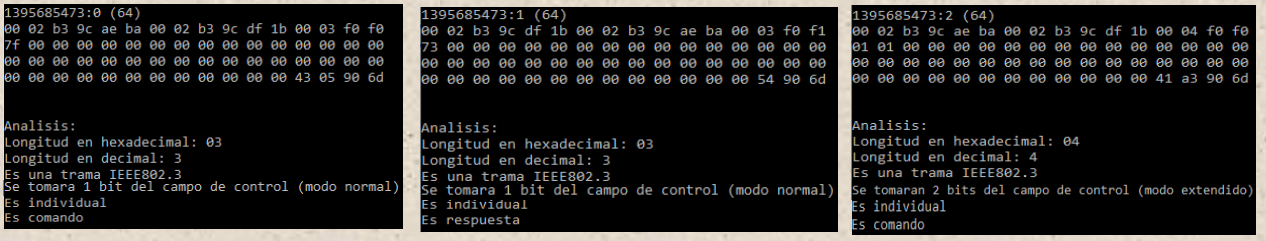
El código se hace de la siguiente manera:

```
if(longitud>3){
    printf("Se tomaran 2 bits del campo de control (modo extendido)\n");
}

else{
    printf("Se tomara 1 bit del campo de control (modo normal)\n");
}
```

Figura 8. Modo normal o extendido.

Se realiza un condicional if-else para determinar cuál modo es el correcto.

Se tomarán los ejemplos anteriores (Figura 3 y Figura 6) para demostrar el funcionamiento del código.



(a) (b) (c)

Figura 9. Demostración Modos. (a) Ejemplo 1, (b) Ejemplo 2, (c) Ejemplo 3.

Ahora se debe comprobar qué tipo de mensaje lleva la trama. Para eso se recurre a los diagramas de la Figura 7.

Como se puede observar, lo que determina qué tipo de mensaje se lleva son: el último bit (trama I) o los últimos 2 bits (tramas S y U).

A continuación, se muestra y se explica el código utilizado.

```
unsigned char b1 = pkt_data[16]&0x01;
unsigned char b2 = (pkt_data[16]>>1)&0x01;

    //printf("%x\n\n",b1);
    //printf("%x\n\n",b2);

if(b1==0){
    printf("Su mensaje es de tipo I\n");
    }
else if(b1==1 && b2==0){
    printf("Su mensaje es de tipo S\n");
}else if(b1==1 && b2==1){
    printf("Su mensaje es de tipo U\n");

}
```

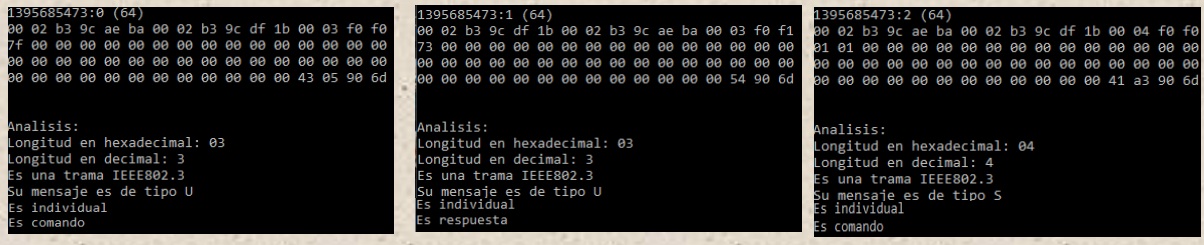
Figura 10. Código tipos de mensaje.

Primero se declaran los 2 bits que vamos a usar, el 16 y el 17, en el caso de este último se toma el bit 16 y se hace un corrimiento a la derecha. A ambos se les aplica la máscara 0x01 para garantizar que sólo salgan valores de 0 y 1.

Posteriormente se aplican los condicionales if-else if para mostrar cada uno de los casos:

- Si el último bit es 0, entonces su mensaje es tipo I.
- Si el último bit es 1, puede que el mensaje sea S o U.
    - Si el penúltimo es 0, entonces es S.
    - Si el penúltimo es 1, entonces es U.

Se utilizarán los 3 ejemplos anteriores para demostrar el funcionamiento del código.



(a) (b) (c)

Figura 11. Demostración Tipos de Mensaje. (a) Ejemplo 1, (b) Ejemplo 2, (c) Ejemplo 3.

Posteriormente se mostrarán datos específicos sobre cada uno de los tipos de mensajes. Por ejemplo, número de secuencia, número de acuse, etc.

En el caso del tipo I, se requiere de un número de secuencia y un número de acuse. De igual forma se hará con base en la Figura 7b.

Se usarán los bits 16 y 17 con la máscara 0x7f, cada uno con sus corrimientos a la derecha.

El código para eso se muestra en la siguiente figura.

```
int nsec = (pkt_data[16]>>1)&0x7f;
int nack = (pkt_data[17]>>1)&0x7f;

if(b1==0){
    printf("Su mensaje es de tipo I\n");
    printf("Este es el numero de acuse: %d\n",nack);
    printf("Este es el numero de secuencia: %d\n",nsec);


    }
```

Figura 12. Acuse y secuencia trama I.

Se declaran las variables para el número de secuencia y el número de acuse, donde se van a guardar las operaciones para sacar esos datos.

Dentro del condicional if que se hizo previamente para sacar el tipo de mensaje I, se imprimen dichos números.

Se usará un ejemplo de trama I para hacer la demostración de este código.

```
1395685473:12 (160)
00 02 b3 9c ae ba 00 02 b3 9c df 1b 00 91 f0 f0
04 04 0e 00 ff ef 16 0c 00 00 28 00 28 00 7f 23
ff 53 4d 42 73 00 00 00 00 10 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 82 09
0d 75 00 5d 00 68 0b 02 00 00 00 7f 07 00 80 00
00 00 00 00 00 00 00 01 00 00 00 20 00 00 00 45
53 43 4f 4d 00 57 69 6e 64 6f 77 73 20 34 2e 30
00 57 69 6e 64 6f 77 73 20 34 2e 30 00 04 ff 00
00 00 02 00 02 00 17 00 20 00 5c 5c 50 52 4f 47
59 44 45 53 41 5c 49 50 43 24 00 49 50 43 00 00


Analisis:
Longitud en hexadecimal: 91
Longitud en decimal: 145
Es una trama IEEE802.3
Su mensaje es de tipo I
Este es el numero de acuse: 2
Este es el numero de secuencia: 2
0404
Se tomaran 2 bits del campo de control (modo extendido)
Es individual
Es comando
```

Figura 13. Demostración Trama I.

Por otro lado, en las tramas tipo S, sólo se requiere de un número de acuse. En el código se ve de la siguiente manera:

```
else if(b1==1 && b2==0){
    printf("Su mensaje es de tipo S\n");
    printf("Este es el numero de acuse: %d\n",nack);
```

Figura 14. Acuse trama S.

Se utiliza la misma variable que se había usado con anterioridad para el acuse y se imprime en el condicional else-if destinado para el mensaje tipo S.

La demostración con una trama S se muestra en la siguiente figura.

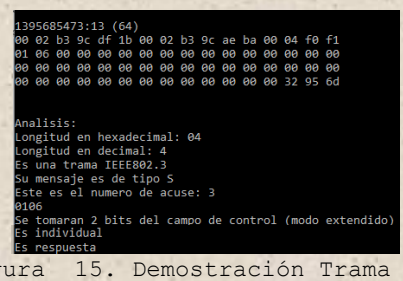


Figura 15. Demostración Trama S.

Ahora se hará un enfoque en el modo extendido. Como se puede observar en la Figura 7b, en el caso de la trama S, se requiere de cierto código representado con la letra S y denominado "código de la trama de supervisión.

Dicho código puede tomar los siguientes valores:

Código Trama S

- 00 Listo para recibir (RR)
- 01 Rechazo (REJ)
- 10 Receptor no listo para recibir (RNR)
- 11 Rechazo selectivo (SREJ)

Figura 16. Código trama S.

En la codificación del programa se puede ver de la siguiente forma:

```
if(b1==1 && b2==0)     {
    int codigo = (pkt_data[16]>>2)&0x03;
    //printf("%d\n",codigo);
    if (codigo== 0){ //00
        printf("Esta listo para recibir\n");
    }
    else if(codigo== 1){  // 01
        printf("Esta listo para rechazar\n");
    }
    else if(codigo== 2){// 10
        printf("Receptor no listo para recibir\n");
    }
    else if(codigo== 3){// 11
        printf("Rechazo selectivo\n");
    }
```

Figura 17. Códigos trama S.

Primero se va a crear una variable para guardar la operación realizada para encontrar los códigos respectivos.

Se inicia con el bit 16 haciendo un corrimiento de 2 bits a la derecha. Luego se le aplica la máscara 0x03 para poder obtener los valores de 00, 01, 10, 11. Es decir, si arroja el valor 0, en realidad lo que mostrará es 00; si arroja 1, se mostrará 01; con el valor 2, se mostrará 10 y finalmente con 3 mostrará 11.

Esto sucede porque se transforman los valores a binarios y se aplica la operación AND.

A continuación, se muestra una pequeña explicación sobre la tabla de verdad de AND y aplicación de la máscara para entenderlo mejor.

Compuerta	Función	Tabla de Verdad
AND	F=A*B	A B F 0 0 0 0 1 0 1 0 0 1 1 1

```
0000000 (0 en binario)
0000011 (3 en binario, por la máscara 0x03)
-------
0000000 (resultado después de aplicar la operación AND)

0000001 (1 en binario)
0000011 (3 en binario, por la máscara 0x03)
-------
0000001 (resultado después de aplicar la operación AND)

0000010 (2 en binario)
0000011 (3 en binario, por la máscara 0x03)
-------
0000010 (resultado después de aplicar la operación AND)

0000011 (3 en binario)
0000011 (3 en binario, por la máscara 0x03)
-------
0000011 (resultado después de aplicar la operación AND)
```

(a) (b)

Figura 18. Explicación. (a) Tabla de verdad AND, (b) Máscara 0x03.

Todo lo anterior se "encierra" en los condicionales if y else-if para poder manejar las opciones adecuadamente.

Cabe destacar que debe ir dentro del condicional marcado para la trama S.

En la siguiente imagen se demuestra el funcionamiento del código.

```
1395685473:30 (64)
00 02 b3 9c df 1b 00 02 b3 9c ae ba 00 04 f0 f1
01 13 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 7c 9b 6d


Analisis:
Longitud en hexadecimal: 04
Longitud en decimal: 4
Es una trama IEEE802.3
Su mensaje es de tipo S
Este es el numero de acuse: 9
0113
Se tomaran 2 bits del campo de control (modo extendido)
Esta listo para recibir
Es individual
Es respuesta
```

Figura 19. Demostración Código Trama S.

En el mismo modo extendido, regresando a la Figura 7b, podemos notar que la trama U, también tiene cierto código denotado con la letra la letra M, que es la función modificadora. Este código puede tomar los siguientes valores:

La forma de plasmar la lógica en código en lenguaje C es de la siguiente manera:

Comandos de control de la trama U con sus respuestas

Código	*Comando*	*Respuesta*	*Significado*
00 001	SNRM		Activación de modo de respuesta normal
11 011	SNRME		Activación de modo de respuesta normal (ampliado)
11 100	SABM	DM	Activación de modo de respuesta asíncrona balanceada
11 110	SABME		Activación de modo de respuesta asíncrona balanceada (ampliada)
00 000	UI	UI	Información sin numerar
00 110		UA	**Reconocimiento sin numerar**
00 010	DISC	RD	Desconexión o **Petición de desconexión**
10 000	SIM	RIM	Activación de modo de iniciación o **Modo de petición de información**
00 100	UP		Muestra sin numerar
11 001	RSET		Reset
11 101	XID	XID	Intercambio de ID
10 001	FRMR	FRMR	Rechazo de trama

Figura 20. Códigos Trama U.

```
else if(b1==1 && b2==1){
int codigo1= (pkt_data[16]>>2)&0x03;
int codigo2= (pkt_data[16]>>5)&0x07;

    switch(codigo1){

        case 0:
              if(codigo2==0){
               printf("Comando UI y Respuesta UI (Información sin numerar)\n");
              }

              else if(codigo2==1){
               printf("Comando SNRM/n");
              }

              else if(codigo2==2){
               printf("Comando DISC(Desconexion o peticion de desconexion) y Respuesta RD\n");
              }

              else if(codigo2==4){
               printf("Comando UP(Muestra sin numerar)\n");
              }
              else if(codigo2==6){
               printf("Respuesta UA(Reconocimiento sin numerar)\n");
              }
            break;
        case 2:
              if(codigo2==0){
               printf("Comando SIM y Respuesta RIM(Activacion de modo de iniciacion)\n");
              }

              else if(codigo2==1){
               printf("Comando FRMR y Respuesta FRMR(Rechazo de trama)\n");
              }

            break;

        case 3:
              if(codigo2==1){
               printf("Comando RSET (Reset)\n");
              }

              else if(codigo2==3){
               printf("Comando SNRME(Activacion de modo de respuesta normal ampliada)\n");
              }

              else if(codigo2==4){
               printf("Comando SABM y Respuesta DM(Activacion de modo respuesta asincrona balanceada)\n");

              }

              else if(codigo2==5){
               printf("Comando XID y Respuesta XID(Intercambio de ID)\n");
              }

              else if(codigo2==6){
               printf("Comando SABME(Activación de modo respuesta asincrona balanceada ampliada)\n");
              }

            break;

        default:
                break;
    }
```

Figura 21. Códigos trama U.

Sólo se van a declarar 2 valores para las 2 partes de los diferentes, ya que como podemos ver en la Figura 7b, aparecen primero 2 bits y luego 3.

Para la primera parte de los 2 bits, se inicia en la posición 16 con un corrimiento de 2 bits a la derecha. Se le aplica una máscara de 0x03 para asegurar que sólo salgan 2 números (03 en binario es 11, están sólo 2 bits "encendidos").

En la segunda parte de los 3 bits se posiciona en el lugar 16 con un corrimiento de 5 bits a la derecha y con la máscara 0x07; esto afirma que saldrán 3 valores (07 en binario es 111, sólo habrá 3 bits encendidos).

Después se aplica la misma lógica que la de la Figura 18b para que salgan los valores deseados (véase la Figura 20).

En este caso se eligió el condicional switch para no anidar if's en exceso. Se toma como referencia la variable codigo1, la cual simboliza la parte del código de la trama U donde sólo hay 2 bits.

Dentro de cada uno de los casos se anidan los condicionales if y else-if para tratar la segunda parte del código de la trama U con 3 bits.

A continuación, se demuestra el funcionamiento del código con una de las tramas U.

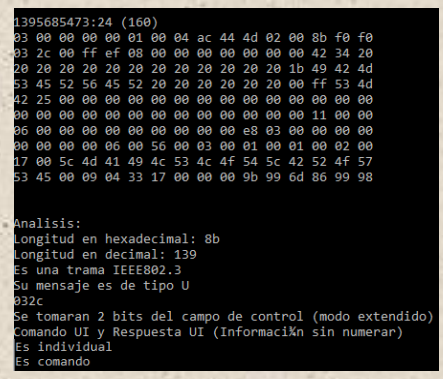


Figura 22. Demostración Códigos Trama U.

Ahora, de igual forma, en el modo extendido, se va a buscar el bit P/F (Pol/Final) de cada una de las tramas I, S y U.

Se iniciará con la trama I. El código para localizar dicho bit es el siguiente.

```
int pf1 = (pkt_data[17]>>1)&0x01;
    if(b1==0){
       if (pf1==0){
        printf("P/F es 0, entonces esta apagado\n");
          }
       else {
        printf("P/F es 1, entonces esta prendido\n");
    }
}
```

Figura 23. P/F para trama I

Primero se declara una variable para guardar el resultado de la operación. Se localiza la posición 17, con corrimiento de 1 bit a la derecha y la aplicación de la máscara 0x01.

Luego se realiza un condicional if-else para mostrar los casos que puede haber.

Es importante destacar que todo esto se encierra en el condicional utilizado anteriormente para la trama I.

A continuación, se presenta la demostración del código de la Figura 23.

```
1395685473:16 (144)
00 02 b3 9c ae ba 00 02 b3 9c df 1b 00 7e f0 f0
06 06 0e 00 ff ef 16 0c 00 00 28 00 28 00 7f 23
ff 53 4d 42 25 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 03 c0 00 00 00 00 82 0a
0e 20 00 00 00 08 00 00 10 00 00 00 00 88 13 00
00 00 00 20 00 4c 00 00 00 00 00 00 00 2d 00 5c
50 49 50 45 5c 4c 41 4e 4d 41 4e 00 68 00 57 72
4c 65 68 44 7a 00 42 31 36 42 42 44 7a 00 01 00
00 10 ff ff ff ff 45 53 43 4f 4d 00 00 6f 96 6d


Analisis:
Longitud en hexadecimal: 7e
Longitud en decimal: 126
Es una trama IEEE802.3
Su mensaje es de tipo I
Este es el numero de acuse: 3
Este es el numero de secuencia: 3
0606
Se tomaran 2 bits del campo de control (modo extendido)
P/F es 1, entonces esta prendido
Es individual
Es comando
```

Figura 24. Demostración P/F trama I.

Pasando a la trama S, en la Figura 7b podemos notar que es similar a la trama I, la única diferencia es que la instrucción se coloca en el condicional else-if preestablecido para la trama S. En la siguiente figura la ejemplificación del código.

```
1395685473:17 (64)
00 02 b3 9c df 1b 00 02 b3 9c ae ba 00 04 f0 f1
01 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 be 96 6d


Analisis:
Longitud en hexadecimal: 04
Longitud en decimal: 4
Es una trama IEEE802.3
Su mensaje es de tipo S
Este es el numero de acuse: 4
0108
Se tomaran 2 bits del campo de control (modo extendido)
Esta listo para recibir
P/F es 0, entonces esta apagado
Es individual
Es respuesta
```

Figura 25. Demostración P/F trama S.

Para la trama U se aplica un poco diferente, en lugar de estar posicionado en el lugar 17 y hacer un corrimiento a la derecha, se inicializa en puesto 16 y se realiza un corrimiento de 4 bits a la derecha.

Esta instrucción se coloca en el condicional antes establecido para la trama U. En la siguiente figura se muestra el código y su ejemplificación.

```
int pf2 = (pkt_data[16]>>4)&0x01;

if (pf2==0){

    printf("P/F es 0, entonces esta apagado\n");

}
else {
    printf("P/F es 1, entonces esta prendido\n");
}
```

(a)

```
1395685473:24 (160)
03 00 00 00 00 01 00 04 ac 44 4d 02 00 8b f0 f0
03 2c 00 ff ef 08 00 00 00 00 00 00 42 34 20
20 20 20 20 20 20 20 20 20 20 20 1b 49 42 4d
53 45 52 56 45 52 20 20 20 20 20 20 00 ff 53 4d
42 25 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 11 00 00
06 00 00 00 00 00 00 00 00 00 e8 03 00 00 00 00
00 00 00 00 06 00 56 00 03 00 01 00 01 00 02 00
17 00 5c 4d 41 49 4c 53 4c 4f 54 5c 42 52 4f 57
53 45 00 09 04 33 17 00 00 00 9b 99 6d 86 99 98


Analisis:
Longitud en hexadecimal: 8b
Longitud en decimal: 139
Es una trama IEEE802.3
Su mensaje es de tipo U
032c
Se tomaran 2 bits del campo de control (modo extendido)
Comando UI y Respuesta UI (Informaci¾n sin numerar)
P/F es 0, entonces esta apagado
Es individual
Es comando
```

(b)

Figura 26. P/F trama U. (a) Código, (b) Demostración

Finalmente pasamos al modo normal de cada una de las tramas. Se empezará a desglosar la trama I.

De la trama I, se sacan sus números de acuse y secuencia y su P/F.

El código se muestra a continuación.

```
int seq2= (pkt_data[16]>>1)&0x07;
int pf3= (pkt_data[16]>>3)&0x01;
int ak2= (pkt_data[16]>>4)&0x07;

if(b1==0){

   printf("Este es su numero de secuencia: %d\n",seq2);
   printf("Este es su numero acuse: %d\n",ak2);

   if(pf3==0){
    printf("Esta P/F apagado: %d\n",pf3);
   }
   else{
    printf("Esta P/F encendido: %d\n",pf3);
   }
}
```

Figura 27. Código Trama I, modo normal.

Se declaran las variables a usar. En todos los casos se inicializará en la posición 16, lo diferente serán las máscaras y el corrimiento.

En el caso de los números de secuencia y acuse se aplica la máscara 0x07, con la diferencia de que en el primero se hace corrimiento de 1 bit a la derecha, mientras que en el segundo es de 4 bits a la derecha.

Para el P/F se coloca en la posición 16 con un corrimiento a la derecha de 3 bits y se le aplica la máscara 0x01.

En el condicional destinado para la trama I se encapsula todo el proceso y los demás condicionales requeridos.

En el caso de la trama S, se requiere número de acuse y número de supervisión.

```
else if(b1==1 && b2==0){

int sc= (pkt_data[16]>>2)&0x03;
printf("Este es el numero de acuse: %d\n",ak2);
printf("Este es el codigo de supervision: %d\n",sc);

    if(pf3==0){
     printf("Esta P/F apagado: %d\n",pf3);
    }
    else{
     printf("Esta P/F encendido: %d\n",pf3);
    }
}
```

Figura 28. Código trama S, modo normal.

Como se puede observar en la Figura 28, se utiliza el mismo número de acuse que la trama I, y para el número de supervisión se posiciona en el lugar 16 con un corrimiento a la derecha de 2 bits y la máscara 0x03. El P/F es el mismo que la trama I.

Por otro lado, el caso de la trama U, regresándose a la Figura 7a, se puede notar que requiere de 2 "pedazos" de bits sin numerar y el P/F.

```
else if(b1==1 && b2==1){

int ub= (pkt_data[16]>>2)&0x03;
printf("Estos son los bits sin numerar: %.2x\n",ak2);
printf("Estos son los bits sin numerar: %.2x\n",ub);

   if(pf3==0){
    printf("Esta P/F apagado: %d\n",pf3);
   }
   else{
    printf("Esta P/F encendido: %d\n",pf3);
   }
}
```

Figura 29. Código trama U, modo normal.

Se observa que uno de los "pedazos" de bits sin numerar están en la misma posición que el número de acuse de las tramas I y S, por tanto, se usa esa misma variable.

Para el otro "cacho" se coloca en el lugar 16 y se aplica un corrimiento de 2 bits a la derecha y la máscara 0x03. El P/F es el mismo que las otras 2 tramas.

2.2. Cálculos

Para comprobar que el funcionamiento del programa sea el correcto, se tomarán 3 tramas de ejemplo (1 trama I, 1 trama S y 1 trama U) y se calculará que, efectivamente son del tipo que se dice.

Se usará el número colocado debajo de donde dice qué tipo de mensaje lleva la trama, el corresponde a lo que hay dentro del campo.

Se iniciará con la trama I.

Figura 30. Ejemplo Trama I.

Como se puede observar el número que se tomará es el 0001.

Se transformará dicho número a binario: 0000 0000 0000 0001

Se toman los primeros 2 bits (leyendo de izquierda a derecha) y se aplica la máscara 0x01, como se explicó la sección anterior.

0000 0000 (00 en binario)
0000 0001 (1 en binario por la máscara aplicada)

0000 000**0** (resultado al aplicar la máscara)

Al observar lo obtenido en negritas, se puede notar que da 0, que es el valor necesario para que la trama sea I, por lo tanto, es correcto.

A continuación, se hablará de la trama S.

Figura 31. Ejemplo trama S.

El número usado será 0101, transformado a binario: 0000 0001 0000 0001.

Se toman los primeros 2 bits (de izquierda a derecha) y se aplica la máscara 0x01.

0000 0001 (01 en binario)
0000 0001 (máscara 0x01)

0000 000**1** (resultado aplicando la máscara)

Se realiza el corrimiento a la derecha y se aplica de nuevo la máscara.

0000 0000 (con corrimiento)
0000 0001 (máscara 0x01)

0000 000**0** (resultado aplicando la máscara)

Como se observa en los resultados obtenidos en negritas, queda finalmente **01**, lo necesario para que sea una trama S. Lo que implica que está correcto.

Finalmente, para la trama U se usará el siguiente ejemplo:

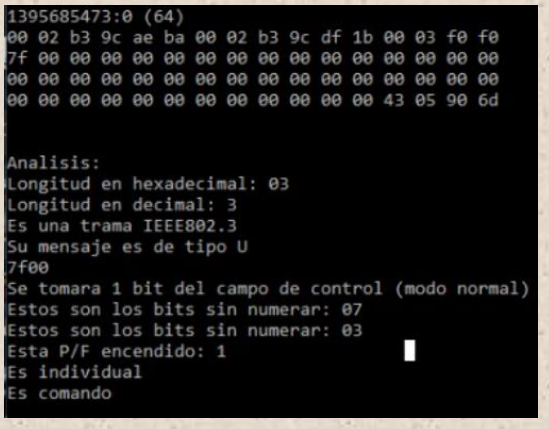


Figura 32. Ejemplo trama U.

El número que se va a utilizar es 7f00, transformado a binario: 0111 1111 0000 0000.

Igual que en los casos anteriores se toman los primeros 2 bits y se aplica la máscara.

0111 1111
0000 0001

0000 000**1**

Ahora se hace el corrimiento a la derecha y se vuelve a aplicar la máscara.

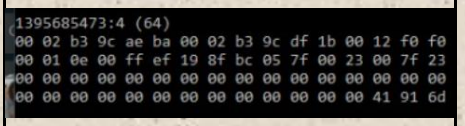
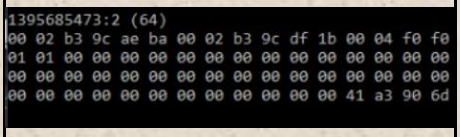
0011 1111
0000 0001

0000 000**1**

Al observar que al final queda **11**, que es lo necesario para la trama sea U, podemos afirmar que el programa trabaja correctamente.

En la Tabla 1 se muestran los datos de las trama anteriores a detalle.

Tabla 1 Datos de las tramas anteriores.

Trama	Tamaño (bytes)	Campo de Control (en binario)	Tipo de Trama	N(s)	N(r)	P/F
	18	00000000-00000001	I	0	0	0
	4	00000001-00000001	S	----	0	0
	3	01111111-00000000	U	----	----	1

3. Conclusiones

3.1. González Mora Erika Giselle

Ethernet es la tecnología LAN más utilizada hoy en día. Es una familia de tecnologías de red que se definen en los estándares IEEE 802.2 y 802.3. Los estándares de Ethernet definen los protocolos de capa 2 y las tecnologías de capa 1. Para los protocolos de capa 2, como con todos los estándares IEEE 802, Ethernet depende de ambas subcapas individuales de la capa de enlace de datos para funcionar: la subcapa de control de enlace lógico (LLC) y la subcapa MAC.

Básicamente al poder analizar diferentes tramas acerca del protocolo LLC , pude entender lo que es el protocolo de Control de enlace lógico LLC ("Logical Link Control"), además de que define la forma en que los datos son transferidos sobre el medio físico, proporcionando servicio a las capas superiores. Es la más alta de las dos subcapas de enlace de datos definidas por el IEEE y la responsable del control de enlace lógico. La subcapa LLC maneja el control de errores, control del flujo, entramado, control de diálogo y direccionamiento de la subcapa MAC. El protocolo LLC más generalizado es IEEE 802.2, que incluye variantes no orientadas a conexión y orientadas a conexión.

Finalmente pude practicar tanto el analizar tramas por LLC, como el poder aplicarlo a un programa en C, y así generar un enlace entre la carrera de Sistemas Computacionales y la unidad de aprendizaje Redes de computadoras.

3.2. Olivares Ménez Gloria Oliva

Sin duda la realización de esta práctica fue complicada. Sobre todo, la parte de localizar correctamente la posición, dado que, si no se ubicaba adecuadamente, el programa se rompía porque sacaba datos completamente a los que estábamos buscando.

Sin embargo, esta práctica me ayudó a entender mejor el funcionamiento en general el protocolo LLC, el cual nos dice cómo los datos son transferidos sobre el medio físico, es decir, algún cable.

Realmente esto es muy útil para comprender cómo funcionan las redes, o al menos encapsuladas en este protocolo, ya que lo utilizamos realmente todos los días y sobre todo con la emancipación del Internet.

Disfruté mucho la programación de esta práctica porque a pesar de que complicado, como lo mencioné anteriormente, me ayudó demasiado a entender mejor el protocolo, y reforzar en su totalidad lo visto en clase.

Espero poder empaparme más de este tipo de conocimientos para entender mejor como funciona mi entorno.

4. Bibliografía

4.1.3. Control de enlace lógico: Conexión con las capas superiores. - Redes de Computadoras. (s. f.). 4.1.3. Control de enlace lógico: Conexión con las capas superiores. - Redes de Computadoras. Recuperado 28 de marzo de 2021, de https://sites.google.com/site/redesdecomputadorashamed/unidad-4-tecnologia-ethernet/4-1-descripcion-general-de-ethernet/4-1-3-control-de-enlace-logico-conexion-con-las-capas-superiore

IEEE 802.2 - IEEE 802.2 - qaz.wiki. (s. f.). IEEE 802.2. Recuperado 28 de marzo de 2021, de https://es.qaz.wiki/wiki/IEEE 802.2

5. Anexos.

5.1 Programa LLC.c

```
#include <stdio.h>
#include <stdlib.h>
#include "C:\\Users\\escom\\Desktop\\REDES\\WpdPack\\Include\\pcap\\pcap.h"
#include <pcap.h>
#define LINE_LEN 16
#define RUTA "C:\\Users\\escom\\Documents\\DEVC\\paquetes3.pcap"
#define  PCAP_OPENFLAG_PROMISCUOUS   1
#define  PCAP_SRC_FILE   2
#define  PCAP_BUF_SIZE   1024

void dispatcher_handler(u_char *, const struct pcap_pkthdr *, const u_char *);

int main(int argc, char **argv)
{
pcap_t *fp;
char errbuf[PCAP_ERRBUF_SIZE];
char source[PCAP_BUF_SIZE];

   /* if(argc != 2){

        printf("usage: %s filename", argv[0]);
        return -1;

    }*/
```

```
    /* Create the source string according to the new WinPcap syntax */
    if ( pcap_createsrcstr( source,         // variable that will keep the source
string
                            PCAP_SRC_FILE,  // we want to open a file
                            NULL,           // remote host
                            NULL,           // port on the remote host
                            RUTA, //argv[1],        // name of the file we want to
open
                            errbuf          // error buffer
                            ) != 0)
    {
        fprintf(stderr,"\nError creating a source string\n");
        return -1;
    }

    /* Open the capture file */
    if ( (fp= (pcap_t *)pcap_open(source,         // name of the device
                          65536,          // portion of the packet to capture
                                          // 65536 guarantees that the whole packet
will be captured on all the link layers
                           PCAP_OPENFLAG_PROMISCUOUS,     // promiscuous mode
                           1000,              // read timeout
                           NULL,             // authentication on the remote machine
                           errbuf         // error buffer
                           ) ) == NULL)
    {
        fprintf(stderr,"\nUnable to open the file %s\n", source);
        return -1;
    }

    // read and dispatch packets until EOF is reached
    pcap_loop(fp, 0, dispatcher_handler, NULL);

    return 0;
}



void dispatcher_handler(u_char *temp1,
                        const struct pcap_pkthdr *header, const u_char *pkt_data)
{
    u_int i=0;

    /*
     * Unused variable
     */
    (VOID)temp1;

    /* print pkt timestamp and pkt len */
    printf("%ld:%ld (%ld)\n", header->ts.tv_sec, header->ts.tv_usec, header->len);

    /* Print the packet */
    for (i=1; (i < header->caplen + 1 ) ; i++)
    {
        printf("%.2x ", pkt_data[i-1]);
        if ( (i % LINE_LEN) == 0) printf("\n");
    }

    printf("\n\n");

}
```

5.2 Analizador Protocolo LLC.

```
//Código previamente cargado.
void dispatcher_handler(u_char *temp1,
                        const  struct  pcap_pkthdr  *header,  const  u_char
*pkt_data)
{
    u_int i=0;

    /*
     * Unused variable
     */
    (VOID)temp1;

    /* print pkt timestamp and pkt len */
    printf("%ld:%ld   (%ld)\n",   header->ts.tv_sec,   header->ts.tv_usec,
header->len);

    /* Print the packet */
    for (i=1; (i < header->caplen + 1 ) ; i++)
    {
        printf("%.2x ", pkt_data[i-1]);
        if ( (i % LINE_LEN) == 0) printf("\n");
    }

    printf("\n\n");

//Analizador.

       printf("Analisis:\n");

       unsigned char longitud = (pkt_data[12]*256)+pkt_data[13];
    printf("Longitud en hexadecimal: %.2x \n",longitud);
    printf("Longitud en decimal: %d \n",longitud);


    if(longitud<=1500){
       printf("Es una trama IEEE802.3\n");
       unsigned char b1 = pkt_data[16]&0x01;
       unsigned char b2 = (pkt_data[16]>>1)&0x01;

           //printf("%x\n\n",b1);
           //printf("%x\n\n",b2);

       int nsec = (pkt_data[16]>>1)&0x7f;
           int nack = (pkt_data[17]>>1)&0x7f;
              int pf1 = (pkt_data[17]>>1)&0x01;

       if(b1==0){
           printf("Su mensaje es de tipo I\n");
                  printf("Este es el numero de acuse: %d\n",nack);
                  printf("Este es el numero de secuencia: %d\n",nsec);


                     }
```

```
                else if(b1==1 && b2==0){
                       printf("Su mensaje es de tipo S\n");
                    printf("Este es el numero de acuse: %d\n",nack);



                }else if(b1==1 && b2==1){
                       printf("Su mensaje es de tipo U\n");


                }

                printf("%.2x",pkt_data[16]);
             printf("%.2x\n",pkt_data[17]);


        if(longitud>3){
                printf("Se  tomaran  2  bits  del  campo  de  control  (modo
extendido)\n");

                if(b1==0){

                          if (pf1==0){

                       printf("P/F es 0, entonces esta apagado\n");

                            }
                          else {
                              printf("P/F    es    1,    entonces    esta
prendido\n");
                       }
        }

                       if(b1==1 && b2==0)        {

                              int codigo = (pkt_data[16]>>2)&0x03;

                              //printf("%d\n",codigo);

                              if (codigo== 0){ //00

                                     printf("Esta listo para recibir\n");

                              }

                              else if(codigo== 1){  // 01

                                  printf("Esta listo para rechazar\n");

                              }

                              else if(codigo== 2){// 10

                                  printf("Receptor    no    listo    para
recibir\n");
```

```
                        }

                        else if(codigo== 3){// 11

                            printf("Rechazo selectivo\n");

                        }
                    if (pf1==0){

                    printf("P/F es 0, entonces esta apagado\n");

                    }
                    else {
                        printf("P/F    es    1,    entonces    esta
prendido\n");

                        }
                    }

                    else if(b1==1 && b2==1){
                    int     codigo1= (pkt_data[16]>>2)&0x03;
                    int codigo2= (pkt_data[16]>>5)&0x07;

                        switch(codigo1){

                              case 0:
                                    if(codigo2==0){
                                     printf("Comando     UI     y
Respuesta UI (Información sin numerar)\n");
                                        }

                                  else if(codigo2==1){
                                     printf("Comando SNRM/n");
                                        }

                                  else if(codigo2==2){
                                     printf("Comando
DISC(Desconexion o peticion de desconexion) y Respuesta RD\n");
                                        }

                                        else if(codigo2==4){
                                     printf("Comando   UP(Muestra
sin numerar)\n");
                                        }

                                  else if(codigo2==6){
                                     printf("Respuesta
UA(Reconocimiento sin numerar)\n");
                                        }
                              break;
                  case 2:
                       if(codigo2==0){
                                     printf("Comando    SIM    y
Respuesta RIM(Activacion de modo de iniciacion)\n");
                                        }
```

```
                                                    else if(codigo2==1){
                                                    printf("Comando    FRMR    y
Respuesta FRMR(Rechazo de trama)\n");
                                                    }

                                                    break;

                                            case 3:
                                                   if(codigo2==1){
                                                    printf("Comando        RSET
(Reset)\n");
                                                    }

                                                    else if(codigo2==3){
                                                    printf("Comando
SNRME(Activacion de modo de respuesta normal ampliada)\n");
                                                    }

                                                    else if(codigo2==4){
                                                    printf("Comando    SABM    y
Respuesta DM(Activacion de modo respuesta asincrona balanceada)\n");

                                                    }

                                                    else if(codigo2==5){
                                                    printf("Comando    XID    y
Respuesta XID(Intercambio de ID)\n");
                                                    }

                                                    else if(codigo2==6){
                                                    printf("Comando
SABME(Activación de modo respuesta asincrona balanceada ampliada)\n");
                                                    }

                                                    break;

                                            default:
                                                     break;
                                    }

                              int pf2 = (pkt_data[16]>>4)&0x01;

                              if (pf2==0){

                              printf("P/F es 0, entonces esta apagado\n");

                              }
                              else {
                                    printf("P/F    es    1,    entonces    esta
prendido\n");
                              }

                              }
                        }

                        else{
```

```
                    printf("Se tomara 1 bit del campo de control (modo
normal)\n");

                    int seq2= (pkt_data[16]>>1)&0x07;
                    int pf3= (pkt_data[16]>>3)&0x01;
                    int ak2= (pkt_data[16]>>4)&0x07;

                    if(b1==0){

                       printf("Este  es  su  numero  de  secuencia:
%d\n",seq2);
                       printf("Este es su numero acuse: %d\n",ak2);

                       if(pf3==0){
                           printf("Esta P/F apagado: %d\n",pf3);
                       }
                       else{
                           printf("Esta P/F encendido: %d\n",pf3);
                       }
                    }

                    else if(b1==1 && b2==0){

                    int sc= (pkt_data[16]>>2)&0x03;
                printf("Este es el numero de acuse: %d\n",ak2);
                printf("Este es el codigo de supervision: %d\n",sc);

                     if(pf3==0){
                           printf("Esta P/F apagado: %d\n",pf3);
                       }
                       else{
                           printf("Esta P/F encendido: %d\n",pf3);
                       }
                    }

                    else if(b1==1 && b2==1){

                    int ub= (pkt_data[16]>>2)&0x03;
                printf("Estos son los bits sin numerar: %.2x\n",ak2);
                printf("Estos son los bits sin numerar: %.2x\n",ub);

                    if(pf3==0){
                           printf("Esta P/F apagado: %d\n",pf3);
                       }
                       else{
                           printf("Esta P/F encendido: %d\n",pf3);
                       }
                     }
            }

            unsigned char dsap=pkt_data[14]&0x01;

            if(dsap==0){
                printf("Es individual\n");
            }else if(dsap==1){
                   printf("Es grupal\n");
```

```
                }

                unsigned char ssap=pkt_data[15]&0x01;
                if(ssap==0){
                        printf("Es comando\n\n");
                }else{
                        printf("Es respuesta\n\n");
                }
}
        else if(longitud>1500){
                printf("Es una trama Ethernet\n");
        }else{
                printf("Inválido\n");
        }


}
```