



**INSTITUTO POLITÉCNICO NACIONAL  
ESCUELA SUPERIOR DE CÓMPUTO**



**Redes de Computadoras**

**“Analizador de tramas: LLC, ARP, IP, ICMP, TCP y UDP”**

**Por:**

**Mora Ayala Jose Antonio**

**Torres Carrillo Josehf Miguel Angel**

**Profesor:**

**M. en C. NIDIA ASUNCIÓN CORTEZ DUARTE**

**21 de Diciembre de 2021**

## Práctica: Analizador de tramas

### Dar una introducción sobre el modelo OSI y la arquitectura TCP/IP (media cuartilla)

#### OSI

El significado de las siglas OSI es **Modelo de interconexión de Sistemas Abiertos** este es abreviado mediante sus siglas en ingles OSI, es un modelo conceptual creado por la Organización internacional de Normalización (mejor conocida como ISO), la cual permite qe diversos sistemas de comunicación entablen una comunicación mutua usando protocolos estándar, llegando a poder definir que este modelo nos proporciona principalmente la ventaja de realizar que diferentes sistemas informáticos tengan un **estándar para comunicarse entre sí**

El modelo OSi en escencia es una normativa formada por siete capas que define las diferentes fases por las que deben pasar los datos para viajar de un dispositvio a otro sobre una red de comunicaciones.

- Capa 7: La capa de aplicación
- Capa 6: La capa de presentación
- Capa 5: La capa de sesión
- Capa 4: La capa de transporte
- Capa 3: La capa de red
- Capa 2: La capa de enlace de datos
- Capa 1: La capa física

#### TCP/IP

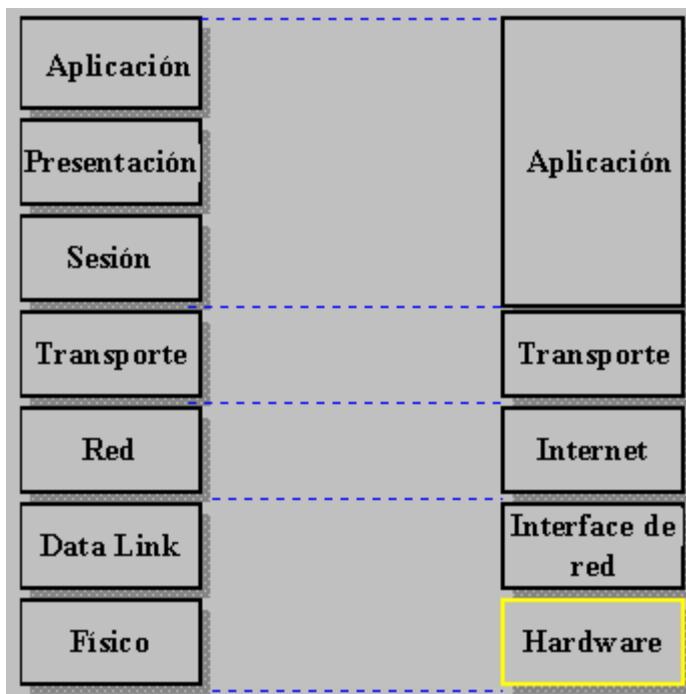
El protocolo TCP/IP se emplea en Internet y algunas veces en redes más pequeñas, especialmente en las que conectan sistemas de computación que corren el sistema operativo UNIX.

TCP/IP permite que diferentes tipos de dispositivos y de proveedores interoperen con cualquier otro, soportando una gran variedad de dispositivos; pero siempre se pueden presentar problemas substanciales por compatibilidad.

Las funciones del software empleadas en los dispositivos en red son divididos dentro del nivel independiente de funciones. La comitiva del protocolo TCP/IP realiza una arquitectura por niveles, teniendo los 4 niveles

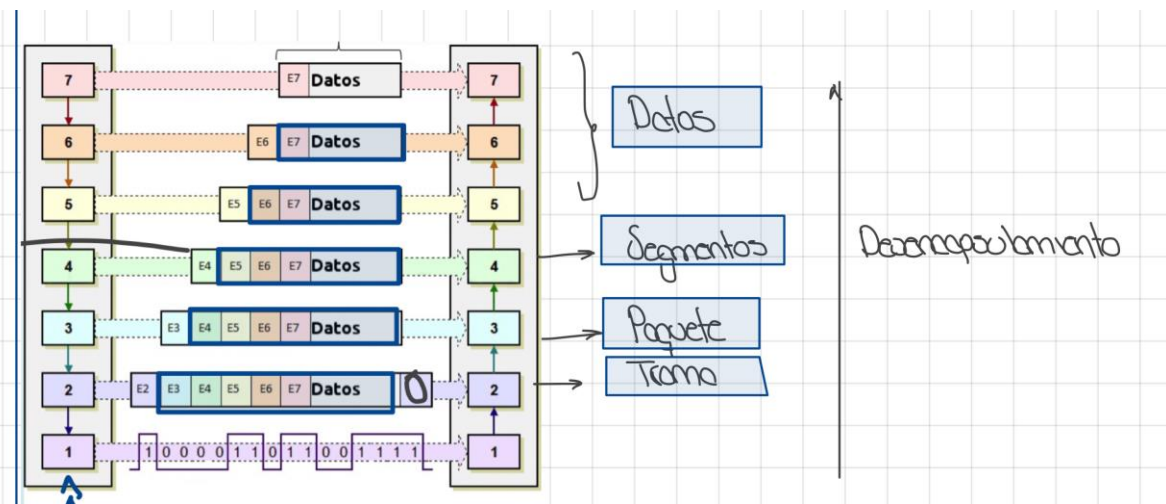
Cuando se emplea TCP/IP, la información viaja en segmentos creados por TCP entre emisor y receptor para acceder a alguna aplicación. Los segmentos creados por TCP son encapsulados por IP, y esta encapsulación es llamada datagramas IP. El datagrama IP permite que los segmentos TCP que fueron hechos por alguna aplicación, sean transmitidos o ruteados en la Red de Área Local o en la Red de Área Extendida.

**Incluir una imagen comparativa entre Modelo OSI y la arquitectura TCP/IP**



**Explicación e imagen sobre el proceso de encapsulado (modelo OSI) indicando el nombre de la información en cada capa**

**Encapsulamiento:** El encapsulamiento de los datos se va dando conforme nuestro mensaje original va viajando a lo largo de las 7 capas del modelo OSI, pues el emisor envía un mensaje y al momento de que este se va moviendo entre las capas sucede el encapsulamiento mediante el añadido de una cabecera propia de cada una de las capas, al momento de que se agrega la cabecera propia en la primera capa y esta pasando a la siguiente (ejemplo de 7 a 6) la capa numero 6 recibe la información como un todo y es a este todo al cual le agregara su cabecera propia, tal como se puede observar en la siguiente imagen



## Explicación de cada una de las capas:

### Capa 1: Física

Este nivel se encarga directamente de los elementos físicos de la conexión. Gestiona los procedimientos a nivel electrónico para que la cadena de bits de información viaje desde el transmisor al receptor sin alteración alguna.

### Capa 2: Enlace de datos

Este nivel se encarga de proporcionar los medios funcionales para establecer la comunicación de los elementos físicos. Se ocupa del direccionamiento físico de los datos, el acceso al medio y especialmente de la detección de errores en la transmisión.

Esta capa construye las tramas de bits con la información y además otros elementos para controlar que la transmisión se haga de forma correcta. El elemento típico que realiza las funciones de esta capa es el switch o también el router, que se encarga de recibir y enviar datos desde un transmisor a un receptor

### Capa 3: Red

Esta capa se encarga de la identificación del enrutamiento entre dos o más redes conectadas. Este nivel hará que los datos puedan llegar desde el transmisor al receptor siendo capaz de hacer las conmutaciones y encaminamientos necesarios para que el mensaje llegue.

#### **Capa 4: Transporte**

Este nivel se encarga de realizar el transporte de los datos que se encuentran dentro del paquete de transmisión desde el origen al destino. Esto se realiza de forma independiente al tipo de red que haya detectado el nivel inferior.

#### **Capa 5: Sesión**

Mediante este nivel se podrá controlar y mantener activo el enlace entre las máquinas que están transmitiendo información. De esta forma se asegurará que una vez establecida la conexión, esta se mantenga hasta que finalice la transmisión.

#### **Capa 6: Presentación**

Como su propio nombre intuye, esta capa se encarga de la representación de la información transmitida. Asegurará que los datos que nos llegan a los usuarios sean entendibles

#### **Capa 7: Aplicación**

Este es el último nivel, y es encargado de permitir a los usuarios ejecutar acciones y comandos en sus propias aplicaciones como por ejemplo un botón para enviar un email o un programa para enviar archivos mediante FTP. Permite también la comunicación entre el resto de capas inferiores.

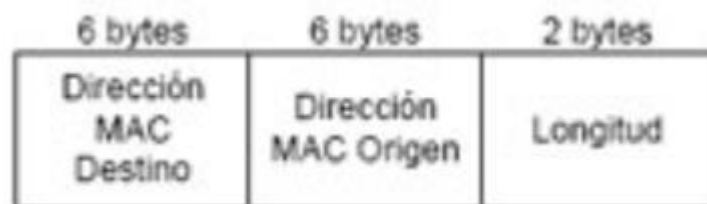
## Cabecera Ethernet

- Introducción

El encabezado de la trama contiene la información de control que especifica el protocolo de capa de enlace de datos para la topología lógica y los medios específicos utilizados.

La información de control de trama es única para cada tipo de protocolo. Es utilizada por el protocolo de la Capa 2 para proporcionar las características demandadas por el entorno de comunicación.

- Imagen de la Cabecera completa  
Cabecera extraída del archivo 4Nid\_CRC en la diapositiva 2



- Descripción breve de cada campo.

Los campos que conforman la cabecera Ethernet son los siguientes

1. MAC Destino – 6 bytes  
Nos indica la dirección de destino física
2. MAC Origen – 6 bytes  
Nos indica la dirección física de origen
3. Longitud o tipo – 2 bytes  
Indica el tipo de trama que sigue puede ser IP, LLC o ARP

Mapa de memoria en donde se vea el campo y el código necesario para consultar dicho campo (como imagen)

Cabecera Ethernet	T[0]								MAC Destino	Printf("%.2x,%.2x,%.2x,%.2x", T[0], T[1], T[2], T[3], T[4], T[5])
	...									
	t[5]								Mac ORIGEN	Printf("%.2x,%.2x,%.2x,%.2x", T[7], T[8], T[9], T[10], T[11], T[12])
	t[6]									
	...									
	t[11]								Tipo	tot = T[12]<<8   T[13]
	t[12]									
	t[13]									

## Protocolo LLC

- Incluir un párrafo de definición de siglas y principal funcionamiento del protocolo, definir en que capa trabaja y su función(es) principal(es)

Las siglas LLC significan **Logical Link Control** o en español **Enlace de Control lógico** es un protocolo para el funcionamiento de las redes LAN, este trabaja en la capa 2 del modelo OSI es decir en la capa de enlace de datos

- Imagen de la Cabecera completa

## Cabecera LLC

1 byte	1 byte	1 ó 2 bytes	variable	variable
SAP Destino	SAP Origen	Control	Información	Relleno

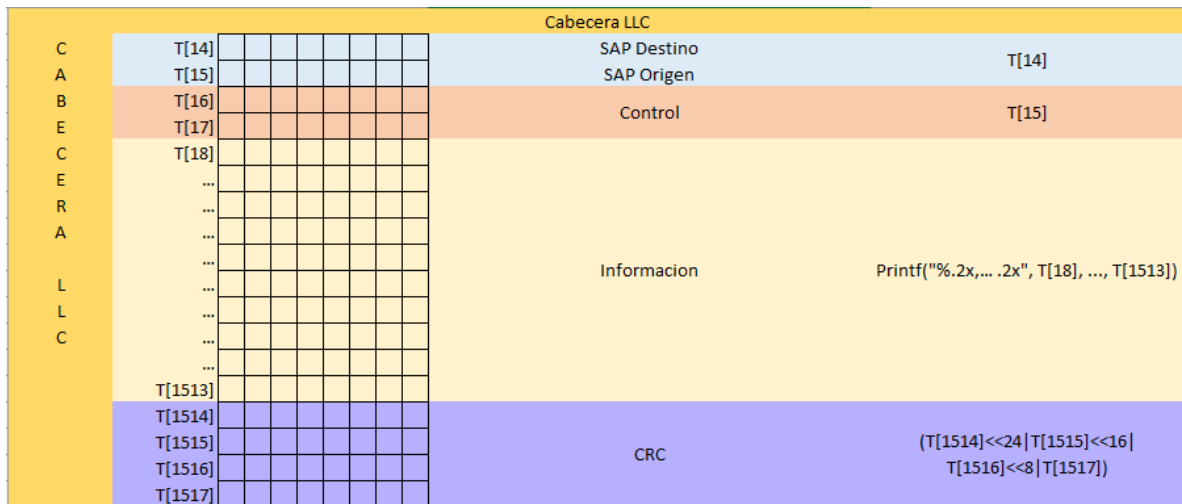
Cabecera LLC extraída del documento 7Nid\_HDLC y LLC Diapositiva 38

- Descripción breve de cada campo.

Los campos que conforman esta cabecera son los siguientes

1. DSAP – 1 byte  
Punto de acceso de servicio del destino (SAP Destino)
2. SSAP – 1 byte  
Consiste en 6 bits de dirección y un bit del usuario además un comando respuesta p/f
3. Control – 1 o 2 bytes  
Campo de control de flujo y error, define que trama se usara, si es una trama de supervision, de información o una no numerada
4. Información – variable  
Es un campo que puede presentarse en la trama con un tamaño máximo de 1500 bytes

- Mapa de memoria (como imagen)



- Incluir lo solicitado en la entrega en classroom

## Introducción protocolo llc

### Definición

Se refiere al control de enlace lógico LLC o Logical Link Control, define la forma en que los datos son transferidos sobre el medio físico proporcionando servicio a las capas superiores, este protocolo esta logado al protocolo HDLC

### Cabecera

Para la cabecera LLC tenemos la siguiente estructura

SAP Destino	SAP Origen	Control	Información	Relleno
-------------	------------	---------	-------------	---------

Donde cada apartado tiene el siguiente tamaño en bytes

1 byte	1 byte	1 o 2 bytes	Variable	Variable
--------	--------	-------------	----------	----------

Para identificar si estamos en una trama LLC o u otras debemos tener en cuenta el campo Tot de la trama ethernet, si el valor de ese apartado es menor a 1500 sabemos que la trama tendrá una cabecera del tipo LLC

### Estructura de los campos de control de T-I

La estructura para este tipo de tramas es la siguiente:

Si la versión del campo de control es de tamaño de 1 byte la trama es la siguiente

LSB	N(S)	N(S)	N(S)	P/F	N(R)	N(R)	N(R)
-----	------	------	------	-----	------	------	------

Si el campo de control es de un tamaño de 2 bytes tenemos la siguiente estructura

LSB	N(S)	N(S)	N(S)	N(S)	N(S)	N(S)	N(S)	P/F	N(R)	N(R)	N(R)	N(R)	N(R)	N(R)	N(R)
-----	------	------	------	------	------	------	------	-----	------	------	------	------	------	------	------



**Donde:**

N(s) : Numero de Secuencia Enviada N(r) : Numero de secuencia recibida

P/F : Bit de sondeo Final (Pull / Final)

**Estructura de los campos de control de T-S**

La estructura para el tipo de tramas “S” o bien Tramas de Supervisión se basa en la siguiente representación:

LSB (1)	LSB (0)	S	S	P/F	N(r)	N(r)	N(r)
---------	---------	---	---	-----	------	------	------

LSB	LSB	S	S	0	0	0	0	N(r)	N(r)	N(r)	N(r)	N(r)	N(r)	N(r)	P/F
-----	-----	---	---	---	---	---	---	------	------	------	------	------	------	------	-----

**Donde:**

N(s) : Numero de Secuencia Enviada N(r) : Numero de secuencia recibida

S : Bits para tramas de supervisión P/F : Bit de sondeo Final (Pull / Final)

Los bits S son aquellos bits que tiene las tramas de supervisión, de acuerdo a la combinación la trama ejecuta cierta acción (00)Receive Ready (10) Receive not Ready (01) Reject (11) Selective Reject

**Estructura de los campos de control de T-U**

LSB (1)	LSB (1)	M	M	P/F	M	M	M
---------	---------	---	---	-----	---	---	---

Son conocidas como tramas no numeradas por la falta de N(r) y N(s) en la estructura de estas, estas no alteran la secuencia o flujo de las tramas de información.

Los bits M son aquellos que determinan la acción que cada una de estas tramas realizara las cuales son órdenes y respuestas, estas se componen de combinaciones de 5 bits

Código Comando Respuesta

11011	SNRME	00000	UI	UI
11000	SARM DM	00110		UA
11010	SARME	00010	DISC	RD
11100	SABM	10000	SIM	RIM
11110	SABME	00100	UP	
		11001	RSET	
		11101	XID	XID
		10001		FRMR

**Bit P/F**

Todas las tramas LLC en su campo de control tienen este bit P/F que se refiere al bit de sondeo/fin también conocido como Poll/Final, su uso depende del valor que tenga y la trama en la que se encuentre, los valores de que puede tener este bit son 0 y 1 donde cada uno corresponde a P o la solicitud de una respuesta y un F para identificar la trama de respuesta devuelta tras la recepción del orden respectivamente

## **Procesos de enmascaramiento para la obtención del tipo de trama**

Para la obtención del tipo de trama que vamos a obtener tomamos la posición numero 16 dentro de la trama (siempre recordemos que el 0 también se está incluyendo en la numeración, dado que estamos hablando de programación en C), ya que tenemos el valor de esta posición podemos realizar una operación binaria con el numero 3, ya que solo nos interesa analizar los 2 últimos bits (viendo de izquierda a derecha, o bien los dos primeros de derecha a izquierda), dado que las combinaciones son (00 TI , 01 TS, 11 TU), son combinaciones que a lo más nos darán el numero 3, por lo cual metemos esto en un switch y dependiendo el caso obtenido se procede a la obtención de cada información que puede proporcionar el tipo de trama

## **¿Cómo fueron obtenidos los bits s y los bits m?**

Para la obtención de los bits S se realizo un enmascaramiento mediante operaciones binarias, es aquí donde pudimos sacar mayor provecho de estas evitando realizar un consumo de memoria innecesario por la creación de variables o por la necesidad de realizar un bucle for, dado que para las tramas S, solo nos interesa el campo de control se realizó la selección de la posición de la trama T[16] que en realidad seria la 17 pero en los programas en C la numeración la comenzamos desde 0, no es necesario tomar la siguiente posición dado que los bits que nos interesan se encuentran precisamente en esa posición del arreglo, ya que tenemos el numero le debemos realizar un corrimiento en 2 posiciones hacía la derecha, pues los bits que nos interesan dentro de este binario de 8 son los que se encuentran en la posición 2 y 3, por ultimo debemos realizar una operación binaria de tipo and, con el numero 3, pues es el que nos devolverá los valores de los bits para saber la combinación que estos generan y poder obtener el comando que ejecuta dicha trama

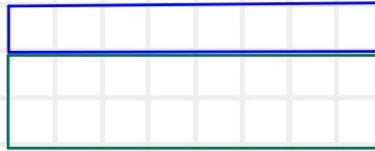
Sucede lo mismo con las tramas de tipo U, nos centramos principalmente en la posición 16 de nuestro arreglo y realizamos un corrimiento en 2 posición primeramente realizamos un corrimiento en 2 posiciones y una operación binaria con el numero 3, para de esta forma poder obtener el valor de 2 de las 5 posiciones necesarias que componen el código de las operaciones de los bits M, posteriormente realizamos un corrimiento en 3 posiciones para recorrer los bits que no nos interesa analizar y realizamos nuevamente una operación binaria con el numero 28 de tal forma que podremos obtener el código correspondiente y así realizar la impresión.

## Mapa de Memoria



### Mapa de Memoria

unsigned char i  
unsigned short int  
"Tot"



Variable utilizada para  
el ciclo "for"

Variable que almacena  
el total de las posiciones  
12 y 13 para determinar  
si es IP, UCL, ARP u otro  
tipo.

## Captura de pantalla salida del programa

```
C:\Windows\system32\cmd.exe

C:\JomianTC\ESCOM\5 Semestre\Redes de computadoras\Clase 20-10-21>t.exe
Alumnos: Mora Ayala Jose Antonio
          Torres Carrillo Josehf Miguel Angel

=====Cabecera ETHERNET 1=====
MAC Destino: 00:02:b3:9c:ae:ba:
MAC Origen: 00:02:b3:9c:df:1b:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 3 bytes
T-U -P, SAMBE

=====Cabecera ETHERNET 2=====
MAC Destino: 00:02:b3:9c:df:1b:
MAC Origen: 00:02:b3:9c:ae:ba:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 3 bytes
T-U -F UA

=====Cabecera ETHERNET 3=====
MAC Destino: 00:02:b3:9c:ae:ba:
MAC Origen: 00:02:b3:9c:df:1b:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 4 bytes
T-S RR -P

=====Cabecera ETHERNET 4=====
MAC Destino: 00:02:b3:9c:df:1b:
MAC Origen: 00:02:b3:9c:ae:ba:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 4 bytes
T-S RR -F

=====Cabecera ETHERNET 5=====
MAC Destino: 00:02:b3:9c:ae:ba:
MAC Origen: 00:02:b3:9c:df:1b:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 18 bytes
T- I, N(s)=0, N(r)=0 -P

=====Cabecera ETHERNET 6=====
MAC Destino: 00:02:b3:9c:df:1b:
MAC Origen: 00:02:b3:9c:ae:ba:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 18 bytes
T- I, N(s)=0, N(r)=1 -P
```

```

C:\Windows\system32\cmd.exe

=====Cabecera ETHERNET 7=====
MAC Destino: 00:02:b3:9c:df:1b:
MAC Origen: 00:02:b3:9c:ae:ba:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 4 bytes
T-S RR -F
=====Cabecera ETHERNET 8=====
MAC Destino: 00:02:b3:9c:ae:ba:
MAC Origen: 00:02:b3:9c:df:1b:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 4 bytes
T-S RR -F
=====Cabecera ETHERNET 9=====
MAC Destino: 00:02:b3:9c:ae:ba:
MAC Origen: 00:02:b3:9c:df:1b:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 172 bytes
T- I, N(s)=1, N(r)=1
=====Cabecera ETHERNET 10=====
MAC Destino: 00:02:b3:9c:df:1b:
MAC Origen: 00:02:b3:9c:ae:ba:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 4 bytes
T-S RR
=====Cabecera ETHERNET 11=====
MAC Destino: 00:02:b3:9c:df:1b:
MAC Origen: 00:02:b3:9c:ae:ba:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 95 bytes
T- I, N(s)=1, N(r)=2
=====Cabecera ETHERNET 12=====
MAC Destino: 00:02:b3:9c:ae:ba:
MAC Origen: 00:02:b3:9c:df:1b:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 4 bytes
T-S RR
=====Cabecera ETHERNET 13=====
MAC Destino: 00:02:b3:9c:ae:ba:
MAC Origen: 00:02:b3:9c:df:1b:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 145 bytes
T- I, N(s)=2, N(r)=2

```

```

C:\Windows\system32\cmd.exe

=====Cabecera ETHERNET 14=====
MAC Destino: 00:02:b3:9c:df:1b:
MAC Origen: 00:02:b3:9c:ae:ba:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 4 bytes
T-S RR
=====Cabecera ETHERNET 15=====
MAC Destino: 00:02:b3:9c:df:1b:
MAC Origen: 00:02:b3:9c:ae:ba:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 70 bytes
T- I, N(s)=2, N(r)=3
=====Cabecera ETHERNET 16=====
MAC Destino: 00:02:b3:9c:ae:ba:
MAC Origen: 00:02:b3:9c:df:1b:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 4 bytes
T-S RR
=====Cabecera ETHERNET 17=====
MAC Destino: 00:02:b3:9c:ae:ba:
MAC Origen: 00:02:b3:9c:df:1b:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 126 bytes
T- I, N(s)=3, N(r)=3
=====Cabecera ETHERNET 18=====
MAC Destino: 00:02:b3:9c:df:1b:
MAC Origen: 00:02:b3:9c:ae:ba:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 4 bytes
T-S RR
=====Cabecera ETHERNET 19=====
MAC Destino: 00:02:b3:9c:ae:ba:
MAC Origen: 00:02:b3:9c:df:1b:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 4 bytes
T-S RR
=====Cabecera ETHERNET 20=====
MAC Destino: 00:02:b3:9c:ae:ba:
MAC Origen: 00:02:b3:9c:df:1b:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 126 bytes
T- I, N(s)=4, N(r)=4

```

```

C:\Windows\system32\cmd.exe

=====Cabecera ETHERNET 21=====
MAC Destino: 00:02:b3:9c:df:1b:
MAC Origen: 00:02:b3:9c:ae:ba:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 4 bytes
T-S RR
=====Cabecera ETHERNET 22=====
MAC Destino: 00:02:b3:9c:ae:ba:
MAC Origen: 00:02:b3:9c:df:1b:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 4 bytes
T-S RR
=====Cabecera ETHERNET 23=====
MAC Destino: 00:02:b3:9c:ae:ba:
MAC Origen: 00:02:b3:9c:df:1b:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 18 bytes
T- I, N(s)=5, N(r)=5 -P
=====Cabecera ETHERNET 24=====
MAC Destino: 00:02:b3:9c:df:1b:
MAC Origen: 00:02:b3:9c:ae:ba:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 4 bytes
T-S RR -F
=====Cabecera ETHERNET 25=====
MAC Destino: 03:00:00:00:00:01:
MAC Origen: 00:04:ac:44:4d:02:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 139 bytes
T-U
=====Cabecera ETHERNET 26=====
MAC Destino: 00:02:b3:9c:ae:ba:
MAC Origen: 00:02:b3:9c:df:1b:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 53 bytes
T- I, N(s)=6, N(r)=5
=====Cabecera ETHERNET 27=====
MAC Destino: 00:02:b3:9c:df:1b:
MAC Origen: 00:02:b3:9c:ae:ba:
=====Cabecera LLC=====
Tamaño de cabecera LLC: 53 bytes
T- I, N(s)=5, N(r)=7

```

```

C:\Windows\system32\cmd.exe

=====Cabecera ETHERNET 28=====
MAC Destino: 00:02:b3:9c:ae:ba:

MAC Origen: 00:02:b3:9c:df:1b:

=====Cabecera LLC=====

Tamaño de cabecera LLC: 18 bytes

T- I, N(s)=7, N(r)=6 -P

=====Cabecera ETHERNET 29=====
MAC Destino: 00:02:b3:9c:df:1b:

MAC Origen: 00:02:b3:9c:ae:ba:

=====Cabecera LLC=====

Tamaño de cabecera LLC: 4 bytes

T-S RR -F

=====Cabecera ETHERNET 30=====
MAC Destino: 00:02:b3:9c:ae:ba:

MAC Origen: 00:02:b3:9c:df:1b:

=====Cabecera LLC=====

Tamaño de cabecera LLC: 18 bytes

T- I, N(s)=8, N(r)=6 -P

=====Cabecera ETHERNET 31=====
MAC Destino: 00:02:b3:9c:df:1b:

MAC Origen: 00:02:b3:9c:ae:ba:

=====Cabecera LLC=====

Tamaño de cabecera LLC: 4 bytes

T-S RR -F

=====Cabecera ETHERNET 32=====
MAC Destino: 00:02:b3:9c:ae:ba:

MAC Origen: 00:02:b3:9c:df:1b:

=====Cabecera LLC=====

Tamaño de cabecera LLC: 3 bytes

T-U -P, DISC

=====Cabecera ETHERNET 33=====
MAC Destino: 00:02:b3:9c:df:1b:

MAC Origen: 00:02:b3:9c:ae:ba:

=====Cabecera LLC=====

Tamaño de cabecera LLC: 3 bytes

T-U -F UA

=====Cabecera ETHERNET 34=====
MAC Destino: ff:ff:ff:ff:ff:ff:

MAC Origen: 00:23:8b:46:e9:ad:

TIPO ARP Tamaño de: 2054 bytes

```



```
C:\Windows\system32\cmd.exe

=====Cabecera ETHERNET 35=====
MAC Destino: 00:23:8b:46:e9:ad:

MAC Origen: 00:1f:45:9d:1e:a2:

TIPO ARP Tamaño de: 2054 bytes

=====Cabecera ETHERNET 36=====
MAC Destino: 00:1f:45:9d:1e:a2:

MAC Origen: 00:23:8b:46:e9:ad:

TIPO IP Tamaño de: 2048 bytes
```

## Conclusiones

¿Habías trabajado a nivel de bits?

No, a lo largo de la carrera nunca había tenido la oportunidad de trabajar a un nivel binario, generalmente siempre he optado por utilizar variables de tipo entero, pues es así como generalmente enseñan, simplemente declarar una variable, sin importar la cantidad de memoria que esta este ocupando o sin importar si realmente es lo más benéfico para la cuestión de la calidad y espacio de memoria del programa, mientras esta te permita realizar lo que deseas no importa el tipo, lo cual ahora me doy cuenta es totalmente erróneo y ahora opto por tener más cuidado con respecto a este tema, me parece increíble lo que se puede realizar con los operadores binarios así como la forma en que podemos sacar ventaja de ellos, pues presentan grandes comodidades y ventajas con respecto a los recursos que serán consumidos con lo que concierne a la implementación el programa.

*-Mora Ayala José Antonio*

No, jamás había trabajado a nivel de bits anteriormente, normalmente siempre trabajaba con únicamente variables de tipo entero y casi no usaba variables char ya que pensaba que era exclusivamente para letras y que no podía almacenar número o que podría operar con ellos, también pensaba que usar los operadores de comparaciones tales como && o || era considerado trabajar con compuertas lógicas y por definición trabajar a nivel de bits esto no es así, en este semestre es el primero que trabajo de esta manera y no solo en esta materia, también justo en la materia de análisis de algoritmo tuve que trabajar con bits para un programa de compresión, así que lo visto en esta materia hasta el momento sobre bits me ayudó mucho para ese programa en concreto

*-Torres Carrillo Josehf Miguel Angel*

¿Generalmente consideras el gasto de memoria?

No, como ya mencione generalmente declaraba variables conforme las necesitaba y sin importar el tipo (aunque no fuese a ocupar toda la cantidad de memoria que esta me aportaba, siempre y cuando me permitiera realizar un bucle o algo por el estilo bastaba para mi, lo cual ahora puedo ver es erróneo y debemos ser mas conscientes con respecto al tema, y no solo nosotros como estudiantes si no que también todos aquellos profesores que estan acostumbrados a simplemente enseñar a declarar valores enteros o flotantes cuando necesites de un valor en punto decimal, deberían propiciar mas consciencia acerca de la memoria consumida con lo que respecta a cada tipo de dato existe en los distintos lenguajes de programación

*-Mora Ayala Jose Antonio*

No, generalmente no me preocupaba la capacidad de memoria que usaba ya que el poder de computo actual es mas que de sobra para hacer estas tareas a este nivel, sin embargo, hoy entiendo por que el manejo de memoria es importante al momento de desarrollar estos programas por que de esta manera usamos menos recursos haciendo más eficiente nuestros programas, no solo en memoria si no también en tiempo de ejecución

*-Torres Carrillo Josehf Miguel Angel*

¿Qué ventajas ofrecen los operadores binarios?

A pesar de ser muy poco usados ofrecen una ventaja con respecto a la cantidad de memoria que te ahorras en comparación de otro tipo de operaciones así como la velocidad que estas proporcionan, pues bien, si en programas pequeños no podemos llegar a apreciarlo de una forma notoria al momento de realizar la implementación de un código estaremos enfrentándonos a este tipo de situaciones, en las cuales debemos cuidar cada uno de los bits que se nos estan proporcionando, así como el tiempo que tarda el programa en poder ser ejecutado y realizar los procedimientos pertinentes que debe de.

*-Mora Ayala Jose Antonio*

Los operadores binarios si bien son operaciones muy poco usadas sirven mucho al momento de querer hacer mas eficiente un programa usando menos recursos, todas las operaciones que un lenguaje ejecuta incluso las definiciones propias de las funciones de las librerías por ejemplo del Lenguaje C, usando estos operadores binarios para ejecutar las operaciones pertinentes, un claro ejemplo es la función pow alojada en math.h que usa operadores binarios para ejecutar el numero de potencias aplicando corrimientos, otra función que usa operadores binarios es la función strcmp alojada en string.h que nos permite comparar 2 cadenas, esta función usara el principio de que si ambas cadenas tienen los mismos bits por lo tanto son iguales, si no, serán diferentes, las comparaciones las hace a nivel de bits aplicando corrimientos, comparaciones con la compuerta and etc, en general los operadores binarios ofrecen una amplia gama de aplicaciones eficientes si se saben utilizar

*-Torres Carrillo Josehf Miguel Angel*

¿La práctica te resulto fácil o difícil?

Una vez que se llega a la comprensión del manejo de bits, así como un poco de práctica con los mismos, esta resulta muy sencilla, puede llegar a ser un tanto confusa, dado que no estamos acostumbrados a trabajar con tanto if-else ya que llegamos a creer que un ciclo sería lo mejor o ir almacenando resultados en cada variable distinta, pero realmente esta practica sirve de mucha practica y proporciona gran conocimiento así como formas de realizar la implementación de los distintos apartados que nos ofrecen los programas

*-Mora Ayala Jose Antonio*

La practica una vez entiendes su funcionamiento es muy sencilla, sin embargo en clase al momento de pensar en como obtener los valores de las tramas, como hacer las comparaciones, no resulta tan sencillo de primera mano, se tiene que tener un conocimiento sobre cómo están construidas las tramas de manera general, además de saber como aplicar los operadores binarios y corrimientos, en clase me resultaba un poco confuso el como obtener las cosas, pero una vez terminada la explicación, lo que se refiere a la programación fue muy sencilla, solamente había que tener cuidado en como se estaban usando las funciones y como manipulábamos las tramas, de ahí en fuera la practica resulto sencilla pero se requiere pensar el como obtendremos las cosas antes de programar

*-Torres Carrillo Josehf Miguel Angel*

## Código

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

void analizaTrama(unsigned char T[]);

void analizaLLC(unsigned char T[]);
void imprimeTI(unsigned char T[]);
void imprimeTS(unsigned char T[]);
void imprimeTU(unsigned char T[]);

int PoF(unsigned char T[]);

void main(int argc, char const *argv[]){

    unsigned char
T[][200]={0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x
00,0x03,0xf0,0xf0,

           0x7f,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,0x00,0x00,

           0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,0x00,0x00,

           0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
05,0x90,0x6d}, //trama1

           {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0
x03,0xf0,0xf1,

0x73,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x54,0x9
0,0x6d}, //trama2

           {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0
x04,0xf0,0xf0,

0x01,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x41,0xa3,0x9
0,0x6d}, //trama3
```

```

        {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0
x04,0xf0,0xf1,

0x01,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xf2,0x9
0,0x6d}}, //trama4

        {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0
x12,0xf0,0xf0,

0x00,0x01,0x0e,0x00,0xff,0xef,0x19,0x8f,0xbc,0x05,0x7f,0x00,0x23,0x00,0x7
f,0x23,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x41,0x9
1,0x6d}}, //trama5

        {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0
x12,0xf0,0xf0,

0x00,0x03,0x0e,0x00,0xff,0xef,0x17,0x81,0xbc,0x05,0x23,0x00,0x7f,0x00,0x2
3,0x7f,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x90,0x9
1,0x6d}}, //trama6

        {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0
x04,0xf0,0xf1,

0x01,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xdf,0x9
1,0x6d}}, //trama7

        {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0
x04,0xf0,0xf1,

```

```

0x01,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x18,0xac,0x9
2,0x6d}}, //trama8

        {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0
xac,0xf0,0xf0,

0x02,0x02,0x0e,0x00,0xff,0xef,0x16,0x04,0x00,0x00,0x00,0x00,0x28,0x00,0x7
f,0x23,

0xff,0x53,0x4d,0x42,0x72,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x8
2,0x09,

0x00,0x77,0x00,0x02,0x50,0x43,0x20,0x4e,0x45,0x54,0x57,0x4f,0x52,0x4b,0x2
0,0x50,

0x52,0x4f,0x47,0x52,0x41,0x4d,0x20,0x31,0x2e,0x30,0x00,0x02,0x4d,0x49,0x4
3,0x52,

0x4f,0x53,0x4f,0x46,0x54,0x20,0x4e,0x45,0x54,0x57,0x4f,0x52,0x4b,0x53,0x2
0,0x33,

0x2e,0x30,0x00,0x02,0x44,0x4f,0x53,0x20,0x4c,0x4d,0x31,0x2e,0x32,0x58,0x3
0,0x30,

0x32,0x00,0x02,0x44,0x4f,0x53,0x20,0x4c,0x41,0x4e,0x4d,0x41,0x4e,0x32,0x2
e,0x31,

0x00,0x02,0x57,0x69,0x6e,0x64,0x6f,0x77,0x73,0x20,0x66,0x6f,0x72,0x20,0x5
7,0x6f,

0x72,0x6b,0x67,0x72,0x6f,0x75,0x70,0x73,0x20,0x33,0x2e,0x31,0x61,0x00,0x0
2,0x4e,

0x54,0x20,0x4c,0x4d,0x20,0x30,0x2e,0x31,0x32,0x00,0x00,0xfb,0x92,0x6d,0x8
6,0xdf}}, //trama9

        {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0
x04,0xf0,0xf1,

0x01,0x04,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

```

```
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7b,0x9
3,0x6d}}, //trama10
```

```
    {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0
x5f,0xf0,0xf0,
```

```
0x02,0x04,0x0e,0x00,0xff,0xef,0x16,0x0c,0x00,0x00,0x28,0x00,0x28,0x00,0x2
3,0x7f,
```

```
0xff,0x53,0x4d,0x42,0x72,0x00,0x00,0x00,0x00,0x80,0x00,0x00,0x00,0x00,0x0
0,0x00,
```

```
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x8
2,0x09,
```

```
0x11,0x05,0x00,0x02,0x02,0x00,0x01,0x00,0x68,0x0b,0x00,0x00,0x00,0x00,0x0
1,0x00,
```

```
0x7f,0x07,0x00,0x80,0x03,0x02,0x00,0x00,0x00,0xe5,0xfe,0x29,0x25,0x7c,0xc
2,0x01,
```

```
0x2c,0x01,0x08,0x08,0x00,0x7f,0x07,0x00,0x80,0x32,0x3e,0xb9,0x3d,0x00,0xc
a,0x93}}, //trama11
```

```
    {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0
x04,0xf0,0xf1,
```

```
0x01,0x04,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,
```

```
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,
```

```
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7c,0x9
4,0x6d}}, //trama12
```

```
    {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0
x91,0xf0,0xf0,
```

```
0x04,0x04,0x0e,0x00,0xff,0xef,0x16,0x0c,0x00,0x00,0x28,0x00,0x28,0x00,0x7
f,0x23,
```

```
0xff,0x53,0x4d,0x42,0x73,0x00,0x00,0x00,0x00,0x10,0x00,0x00,0x00,0x00,0x0
0,0x00,
```

```
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x8
2,0x09,
```

```
0x0d,0x75,0x00,0x5d,0x00,0x68,0x0b,0x02,0x00,0x00,0x00,0x7f,0x07,0x00,0x8
0,0x00,
```

```
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x20,0x00,0x00,0x0
0,0x45,
```

```

0x53,0x43,0x4f,0x4d,0x00,0x57,0x69,0x6e,0x64,0x6f,0x77,0x73,0x20,0x34,0x2
e,0x30,

0x00,0x57,0x69,0x6e,0x64,0x6f,0x77,0x73,0x20,0x34,0x2e,0x30,0x00,0x04,0xf
f,0x00,

0x00,0x00,0x02,0x00,0x02,0x00,0x17,0x00,0x20,0x00,0x5c,0x5c,0x50,0x52,0x4
f,0x47,

0x59,0x44,0x45,0x53,0x41,0x5c,0x49,0x50,0x43,0x24,0x00,0x49,0x50,0x43,0x0
0,0x00}}, //trama13

        {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0
x04,0xf0,0xf1,

0x01,0x06,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x32,0x9
5,0x6d}}, //trama14

        {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0
x46,0xf0,0xf0,

0x04,0x06,0x0e,0x00,0xff,0xef,0x16,0x0c,0x00,0x00,0x28,0x00,0x28,0x00,0x2
3,0x7f,

0xff,0x53,0x4d,0x42,0x73,0x00,0x00,0x00,0x00,0x90,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0xc0,0x00,0x00,0x00,0x00,0x8
2,0x09,

0x03,0x75,0x00,0x29,0x00,0x00,0x00,0x00,0x00,0x02,0xff,0x00,0x00,0x00,0x0
4,0x00,

0x49,0x50,0x43,0x00,0x00,0x81,0x95,0x6d,0x86,0xcb,0x94,0x6d,0x86,0x0d,0x0
9,0x0e}}, //trama15

        {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0
x04,0xf0,0xf1,

0x01,0x06,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x9
6,0x6d}}, //trama16

```



```

        {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0
x7e,0xf0,0xf0,

0x06,0x06,0x0e,0x00,0xff,0xef,0x16,0x0c,0x00,0x00,0x28,0x00,0x28,0x00,0x7
f,0x23,

0xff,0x53,0x4d,0x42,0x25,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0xc0,0x00,0x00,0x00,0x00,0x8
2,0x0a,

0x0e,0x20,0x00,0x00,0x00,0x08,0x00,0x00,0x10,0x00,0x00,0x00,0x00,0x88,0x1
3,0x00,

0x00,0x00,0x00,0x20,0x00,0x4c,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x2d,0x0
0,0x5c,

0x50,0x49,0x50,0x45,0x5c,0x4c,0x41,0x4e,0x4d,0x41,0x4e,0x00,0x68,0x00,0x5
7,0x72,

0x4c,0x65,0x68,0x44,0x7a,0x00,0x42,0x31,0x36,0x42,0x42,0x44,0x7a,0x00,0x0
1,0x00,

0x00,0x10,0xff,0xff,0xff,0xff,0x45,0x53,0x43,0x4f,0x4d,0x00,0x00,0x6f,0x9
6,0x6d}, //trama17

```

```

        {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0
x04,0xf0,0xf1,

0x01,0x08,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,0xbe,0x9
6,0x6d}, //trama18

```

```

        {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0
x04,0xf0,0xf1,

0x01,0x08,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x5
d,0x9
7,0x6d}, //trama19

```

```

        {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0
x7e,0xf0,0xf0,

0x08,0x08,0x0e,0x00,0xff,0xef,0x16,0x0c,0x00,0x00,0x28,0x00,0x28,0x00,0x7
f,0x23,

0xff,0x53,0x4d,0x42,0x25,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0xc0,0x00,0x00,0x00,0x00,0x0
2,0x0b,

0x0e,0x20,0x00,0x00,0x00,0x08,0x00,0x00,0x10,0x00,0x00,0x00,0x00,0x88,0x1
3,0x00,

0x00,0x00,0x00,0x20,0x00,0x4c,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x2d,0x0
0,0x5c,

0x50,0x49,0x50,0x45,0x5c,0x4c,0x41,0x4e,0x4d,0x41,0x4e,0x00,0x68,0x00,0x5
7,0x72,

0x4c,0x65,0x68,0x44,0x7a,0x00,0x42,0x31,0x36,0x42,0x42,0x44,0x7a,0x00,0x0
1,0x00,

0x00,0x10,0x00,0x00,0x00,0x80,0x45,0x53,0x43,0x4f,0x4d,0x00,0x00,0xac,0x9
7,0x6d}, //trama20

```

```

        {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0
x04,0xf0,0xf1,

0x01,0x0a,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xf
b,0x9
7,0x6d}, //trama21

```

```

        {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0
x04,0xf0,0xf1,

0x01,0x0a,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x4a,0x9
8,0x6d}, //trama22

```

```

        {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0
x12,0xf0,0xf0,

```

```

0x0a,0x0b,0x0e,0x00,0xff,0xef,0x14,0x00,0x00,0x00,0x28,0x00,0x00,0x00,0x7
f,0x23,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x99,0x9
8,0x6d}, //trama23

        {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0
x04,0xf0,0xf1,

0x01,0x0d,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x45,0x9
9,0x6d}, //trama24

        {0x03,0x00,0x00,0x00,0x00,0x01,0x00,0x04,0xac,0x44,0x4d,0x02,0x00,0
x8b,0xf0,0xf0,

0x03,0x2c,0x00,0xff,0xef,0x08,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x42,0x3
4,0x20,

0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x1b,0x49,0x4
2,0x4d,

0x53,0x45,0x52,0x56,0x45,0x52,0x20,0x20,0x20,0x20,0x20,0x20,0x00,0xff,0x5
3,0x4d,

0x42,0x25,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x11,0x0
0,0x00,

0x06,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xe8,0x03,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x06,0x00,0x56,0x00,0x03,0x00,0x01,0x00,0x01,0x00,0x0
2,0x00,

0x17,0x00,0x5c,0x4d,0x41,0x49,0x4c,0x53,0x4c,0x4f,0x54,0x5c,0x42,0x52,0x4
f,0x57,

0x53,0x45,0x00,0x09,0x04,0x33,0x17,0x00,0x00,0x00,0x9b,0x99,0x6d,0x86,0x9
9,0x98}, //trama25

        {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0
x35,0xf0,0xf0,

```

```

0x0c,0x0a,0x0e,0x00,0xff,0xef,0x16,0x04,0x00,0x00,0x00,0x00,0x28,0x00,0x7
f,0x23,

0xff,0x53,0x4d,0x42,0x71,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0xc0,0x00,0x00,0x00,0x00,0x0
1,0x50,

0x00,0x00,0x00,0x45,0xf1,0x99,0x6d,0x86,0x45,0x99,0x6d,0x86,0x1f,0x09,0x5
2,0x5b}}, //trama26

        {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0
x35,0xf0,0xf0,

0x0a,0x0e,0x0e,0x00,0xff,0xef,0x16,0x0c,0x00,0x00,0x28,0x00,0x28,0x00,0x2
3,0x7f,

0xff,0x53,0x4d,0x42,0x71,0x00,0x00,0x00,0x00,0x80,0x01,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0xc0,0x00,0x00,0x00,0x00,0x0
1,0x50,

0x00,0x00,0x00,0x00,0x40,0x9a,0x6d,0x86,0x9b,0x99,0x6d,0x86,0x20,0x09,0x7
5,0x5b}}, //trama27

        {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0
x12,0xf0,0xf0,

0x0e,0x0d,0x0e,0x00,0xff,0xef,0x14,0x00,0x00,0x00,0x28,0x00,0x00,0x00,0x7
f,0x23,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x8f,0x9
a,0x6d}}, //trama28

        {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0
x04,0xf0,0xf1,

0x01,0x11,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xde,0x9
a,0x6d}}, //trama29

```

```

        {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0
x12,0xf0,0xf0,

0x10,0x0d,0x0e,0x00,0xff,0xef,0x18,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7
f,0x23,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x2d,0x9
b,0x6d}}, //trama30

        {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0
x04,0xf0,0xf1,

0x01,0x13,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7c,0x9
b,0x6d}}, //trama31

        {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0
x03,0xf0,0xf0,

0x53,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xcb,0x9
b,0x6d}}, //trama32

        {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0
x03,0xf0,0xf1,

0x73,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x77,0x9
c,0x6d}},

        {0xff,0xff,0xff,0xff,0xff,0xff,0x00,0x23,0x8b,0x46,0xe9,0xad,0x08,0
x06,0x00,0x04,

```

```

0x08,0x00,0x06,0x04,0x00,0x01,0x00,0x23,0x8b,0x46,0xe9,0xad,0x94,0xcc,0x3
9,0xcb,

0x00,0x00,0x00,0x00,0x00,0x00,0x94,0xcc,0x39,0xfe },
/*Trama a */

    {0x00,0x23,0x8b,0x46,0xe9,0xad,0x00,0x1f,0x45,0x9d,0x1e,0xa2,0x08,0
x06,0x00,0x01, /*TRAMA b */

0x08,0x00,0x06,0x04,0x00,0x02,0x00,0x1f,0x45,0x9d,0x1e,0xa2,0x94,0xcc,0x3
9,0xfe,

0x00,0x23,0x8b,0x46,0xe9,0xad,0x94,0xcc,0x39,0xcb,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00 },

    {0x00,0x1f,0x45,0x9d,0x1e,0xa2,0x00,0x23,0x8b,0x46,0xe9,0xad,0x08,0
x00,0x46,0x00, /* TRAMA c */

0x80,0x42,0x04,0x55,0x34,0x11,0x80,0x11,0x6b,0xf0,0x94,0xcc,0x39,0xcb,0x9
4,0xcc,

0x67,0x02,0xaa,0xbb,0xcc,0xdd,0x04,0x0c,0x00,0x35,0x00,0x2e,0x85,0x7c,0xe
2,0x1a,

0x01,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x77,0x77,0x77,0x0
3,0x69,

0x73,0x63,0x05,0x65,0x73,0x63,0x6f,0x6d,0x03,0x69,0x70,0x6e,0x02,0x6d,0x7
8,0x00, 0x00,0x1c,0x00,0x01}};

for (unsigned char i = 0; i < 36; ++i){

printf("====Cabecera ETHERNET %d=====\n", i+1);
analizaTrama(T[i]);
}
}

void analizaTrama (unsigned char T[]){

    unsigned short int tot = T[12]<<8 | T[13];

    printf("MAC Destino: %.2x:%.2x:%.2x:%.2x:%.2x:%.2x:\n\n",
T[0],T[1],T[2],T[3],T[4],T[5]);
    printf("MAC Origen: %.2x:%.2x:%.2x:%.2x:%.2x:%.2x:\n\n",
T[6],T[7],T[8],T[9],T[10],T[11]);

    if (tot<1500){

printf("====Cabecera LLC=====\n\n");
        printf("Tama%co de cabecera LLC: %d bytes\n\n", 164, tot);
        analizaLLC(T);
    }
}

```

```

else if (tot == 2048){

    printf("TIPO IP Tama%co de: %d bytes\n", 164, tot);
}
else if (tot == 2054){

    printf("TIPO ARP Tama%co de: %d bytes\n", 164, tot);
}
else{

    printf("OTRO TIPO \n");
}

printf("\n\n");
}

void analizaLLC (unsigned char T[]){

    switch (T[16]&3){

        case 0:
            imprimeTI(T);
            break;

        case 1:
            imprimeTS(T);
            break;

        case 2:
            imprimeTI(T);
            break;

        case 3:
            imprimeTU(T);
            break;

        default:
            printf("ERROR\n");
    }
}

void imprimeTI (unsigned char T[]){

    printf("T- I, N(s)=%d, N(r)=%d", T[16]>>1, T[17]>>1);

    PoF(T);
}

void imprimeTS (unsigned char T[]){

    char ss[][4] = {"RR", "RNR", "REJ", "SREJ"};

    printf("T-S %s", ss[(T[16]>>2)&3]);

    PoF(T);
}

```

```

void imprimeTU (unsigned char T[]){

    char uc[][6]={ "UI", "SIM", "-", "SARM", "UP", "-", "-", "SABM",
                    "DISC", "-", "-", "SARME", "-", "-", "-", "SAMBE",
                    "SNRM", "-", "-", "RSET", "-", "-", "-", "XID", "-",
                    "-", "-", "SRME", "-", "-", "-", "-"};

    char ur[][6]={ "UI", "RIM", "-", "DM", "-", "-", "-", "-", "RD", "-",
                    "-", "-", "UA", "-", "-", "-", "-", "FRMR", "-", "-",
                    "-", "-", "-", "XID", "-", "-", "-", "-", "-", "-", "-", "-"};

    printf("T-U");

    if (T[16] & 16){

        if (T[15] & 1)
            printf(" -F %s", ur[((T[16]>>2)&3) | ((T[16]>>3)&28)]);
        else
            printf(" -P, %s",uc[((T[16]>>2)&3) | ((T[16]>>3)&28)]);
    }
}

int PoF(unsigned char T[]){

    if (T[17] & 1){

        if (T[15]&1)
            printf(" -F");
        else
            printf(" -P");
    }
}

```



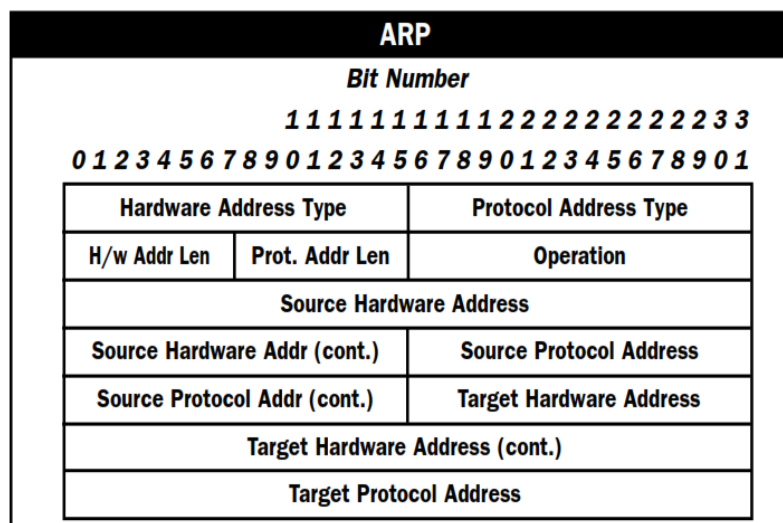
- Junto con el cuestionario y respuestas sobre LLC
1. Tamaño de la cabecera Ethernet \_\_\_\_\_ bytes  
14 bytes
  2. Tamaño de la cabecera LLC \_\_\_\_\_ bytes  
1500 bytes
  3. Puede haber \_\_\_\_\_ bytes de información como máximo en una T-I  
16 bytes
  4. Bytes de relleno en una T-U \_\_\_\_\_  
43 bytes
  5. Campo que indica tipo de trama LLC  
Campo de control
  6. ¿Como saber si una trama es P o F?  
Por el SAPo, si es 0 es P si es 1 es F
  7. ¿Significado de SAPo?  
Punto de acceso al servicio de origen
  8. ¿Cuántos comandos M existen?  
12
  9. ¿Como se obtiene N(s) en una T-I?  
Si es de 1 byte se toman 3 bits después del P/F, si es de 2 los primeros 7
  10. tamaño máximo de la ventana si se usa LLC y no se le permite recibir fuera de secuencia  
 $2^n - 1$

## Protocolo ARP

Incluir un párrafo de definición de siglas y principal funcionamiento del protocolo, definir en que capa trabaja y su función(es) principal(es)

Las siglas ARP significan **Protocolo de resolución de direcciones** o en ingles **Addres Resolution Protocol** este protocolo nos permite obtener el conocimiento de la dirección física de una dirección IP dada, de tal forma que para que las direcciones físicas se puedan conectar con las direcciones lógicas este protocolo en especial opera bajo la capa de enlace de datos del modelo OSI

- Imagen de la Cabecera completa



- Descripción breve de cada campo

Los campos que operan bajo la cabecera ARP son los siguientes

11. Tipo de dirección de Hardware – 2 bytes  
Indica el tipo de dirección física que se tiene
12. Tipo de dirección de IP – 2 bytes  
Indica el tipo de dirección lógica que se tiene
13. Tamaño de dirección de hardware - 1byte  
Nos indica el tamaño de la dirección de hardware
14. Tamaño de dirección IP – 1 byte  
Nos indica el tamaño de la dirección lógica
15. Operación – 2 bytes  
Nos indica la operación que esta realizando, puede ser request, reply, inverse reply e inverse request

## 16. MAC Origen – 6 bytes

Nos indica la dirección física de origen

## 17. IP Origen – 4 bytes

Nos indica la dirección lógica de origen

## 18. MAC Destino – 6 bytes

Nos indica la dirección de destino física

## 19. IP Destino – 24 bytes

Nos indica la dirección lógica de destino

### • Mapa de memoria (como imagen)

Cabecera ARP											
C	T[14]										
	T[15]										
A											
B	T[16]										
E	T[17]										
C	T[18]										
E	T[19]										
R	T[20]										
A	T[21]										
A R P	T[22]										
	T[23]										
	T[24]										
	T[25]										
	T[26]										
	T[27]										
	T[28]										
	T[29]										
	T[30]										
	T[31]										
	T[32]										
	T[33]										
	T[34]										
	T[35]										
	T[36]										
	T[37]										
	T[38]										
	T[39]										
	T[40]										
	T[41]										

Tipo de direccion de Hardware		(T[14]<<8 T[15])
01 01	Ethernet	
00 06	IEEE 802	
00 0f	Frame Relay	
00 10	ATM	

Tipo de direccion IP		(T[16]<<8 T[17])
Tamano de Direccion de Hardware		T[18]
Tamano de Direccion direccion IP		T[19]
Operacion		(T[20]<<8 T[21])

MAC Origen		Printf("%x,%x,%x,%x,%x,%x", T[22], T[23], T[24], T[25], T[26], T[27])
1	Request	
2	Reply	
8 o 3	Request inverse	
9 o 4	Reply Inverse	

IP Origen		Printf("%x,%x,%x,%x", T[28], T[29], T[30], T[31])
MAC Destino		Printf("%x,%x,%x,%x,%x,%x", T[32], T[33], T[34], T[35], T[36], T[37])
IP Destino		Printf("%x,%x,%x,%x,%x,%x", T[38], T[39], T[40], T[41])

- Incluir lo solicitado en la entrega en Classroom

### **Frase Inspiradora**

*“Dios sabe que las personas necesitan un héroe, gente valiente que se sacrifique, poniendo el ejemplo a todos*

*Todo el mundo ama a un héroe, se forman para verlos, aclamarlos, gritar su nombre, y con los años relatan como soportaron horas de lluvia solo para ver al que les enseñó a resistir un segundo mas*

*Me parece que hay un héroe en todos nosotros, nos da fuerza, nos hace nobles, nos mantiene honestos, y al final nos permite morir con orgullo*

*Aunque a veces haya que ser firmes, y renunciar a aquello que mas amamos, incluso hasta nuestros sueños” ~Tía May*

*Arad, A., Laura, Z., (Productores) y Raimi, S., (Director). (2004). Spiderman 2 [Cinta Cinematográfica]. EU.: Columbia Pictures*

## **Siglas ARP**

Las siglas ARP significan Protocolo de resolución de direcciones o en ingles Address Resolution Protocol

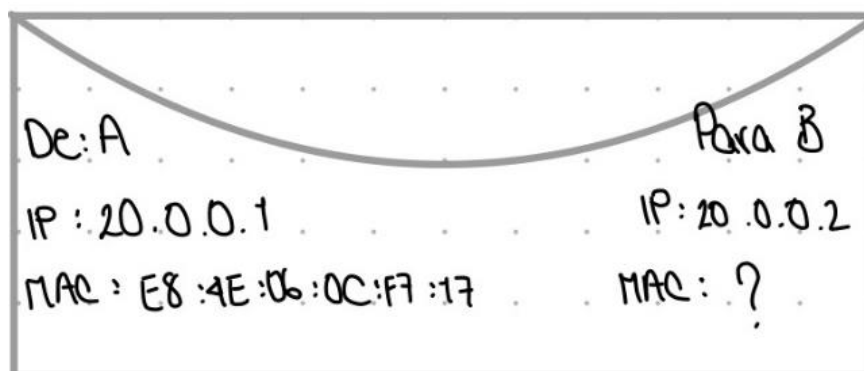
### **Descripción del Protocolo**

Este protocolo nos permite obtener el conocimiento de la dirección física de una dirección IP dada, de tal forma que para que las direcciones físicas se puedan conectar con las direcciones lógicas el protocolo ARP va a realizar la interrogante a todos los equipos de la red para averiguar sus direcciones físicas y poder obtener la que se está requiriendo en ese momento, entonces cuando un equipo debe comunicarse con otro, se realiza una solicitud ARP request con la dirección IP destino a todos los de la red (mediante una dirección de broadcast como destinatario), de tal forma que todos los equipos pertenecientes a este subred puedan comparar la IP destino con la propia en caso de que coincidiera va a mandar la respuesta y si no se encuentra en la misma subred, el remitente se dirige a la puerta de enlace estándar, donde se buscara en todas las tablas de enrutamiento y saldrá por la correspondiente, (la que coincida con la parte correspondiente de la ip) y es momento de volver a mandar un mensaje broadcast buscando la dirección IP adecuada, cuando esta es encontrada responderá y mandara su dirección física .

## Dibujar 2 sobres y con ellos explicar la diferencia de ARP y ARP inverso

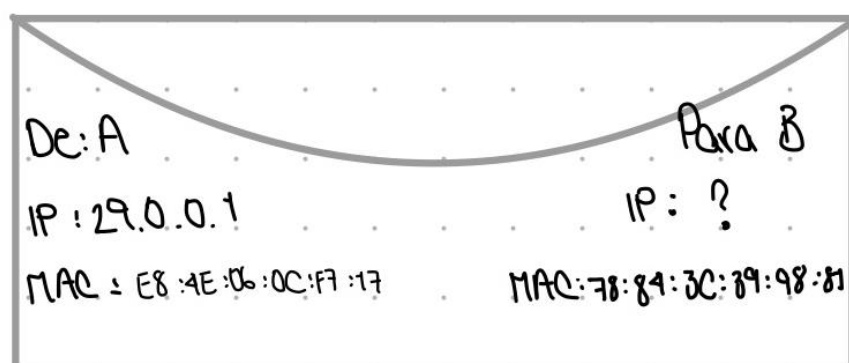
ARP tiene 2 formas de enviar y de recibir los datos estos 2 son los modos ARP y ARP inverso, ambos son sencillos pero con una pequeña diferencia entre sí, tenemos el primer sobre, donde conocemos el destino y el origen, conocemos del origen tanto su IP como su dirección MAC, pero en caso contrario no conocemos la dirección MAC de B por ello primero enviaremos mensaje a todos los que se encuentren en nuestra Red, preguntando sobre quien tiene la IP 20.0.0.2, como todos los usuarios dentro de nuestra red tienen una IP única solo uno podrá responder nuestra solicitud, y ahora que sabemos la MAC de destino podemos mandar el mensaje sin ningún problema a nuestro destino

Pero que pasaria si en lugar de conocer la IP conocemos la MAC y no conocemos la IP? Entonces



este seria un caso inverso de ARP, donde ahora en lugar de preguntar quien tiene la direccion IP, ahora preguntamos quien tiene la direccion MAC que tenemos, de igual manera la respuesta sera unica, es decir una direccion MAC es igual unica por cada dispositivo, por ello la respuesta unicamente la hara un dispositivo, permitiendo identificar la MAC y establecer esta conexion

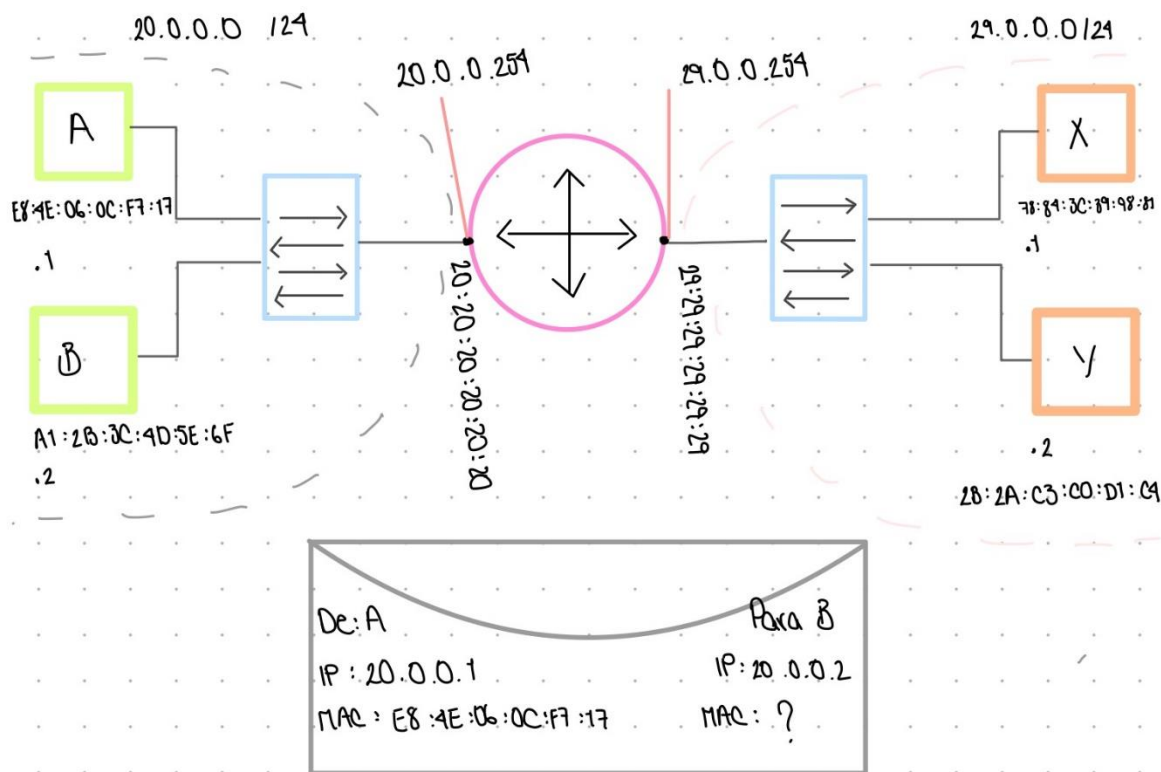
En pocas palabras la diferencia entre un ARP y un ARP inverso es que, en el inverso en lugar de preguntar por la MAC con la IP, preguntaremos la IP con la MAC, es decir en uno conocemos la IP pero no la MAC y en el otro conocemos la MAC pero no la IP



### Poner un escenario (Tú puedes poner las direcciones IP y MAC que gustes)

- Explicar con tus propias palabras cuando se comunicará de A -> B y dibujas el sobre para la solicitud ARP y la respuesta ARP (en el sobre poner las direcciones correspondientes)

Como ya pudimos observar gracias a la previa explicación nos damos cuenta de la forma en que son mandados los mensajes cuando se trata de este tipo de Tramas, la forma de ver esto con sobre resulta



bastante práctica, pues nos da una idea gráfica de lo que está sucediendo al momento de estar realizando el envío de un mensaje con el tipo ARP.

Comenzamos viendo quien es quien desea realizar el envío de información, ya que hemos identificado que se trata de nuestra terminal A y que busca tener una comunicación con B procedemos a realizar las consideraciones necesarias que se llevan a cabo para poder completar esta operación.

Como observamos necesitamos de información indispensable para poder entablar dicha comunicación, gráficamente la conocemos toda, aunque en la realidad no sucede de esta forma.

Realmente solo se dispone de

- MAC Origen
- IP Origen
- IP Destino

Como observamos nos falta la **MAC destino** la cual buscamos obtener para entablar la comunicación correspondiente por lo que se procede a mandar el mensaje de forma 1 a Todos, haciendo de cierta forma la pregunta de: “Oigan quiero mandar este mensaje y busco a tal IP (ya que esta si es conocida y es la que nos llevara a poder realizar una búsqueda)”, tal que se realizar un mensaje de tipo

BROADCAST ya que como tal desconocemos el específico de quien debe recibir el mensaje, aunque recordemos que si sabemos la IP en cuestión.

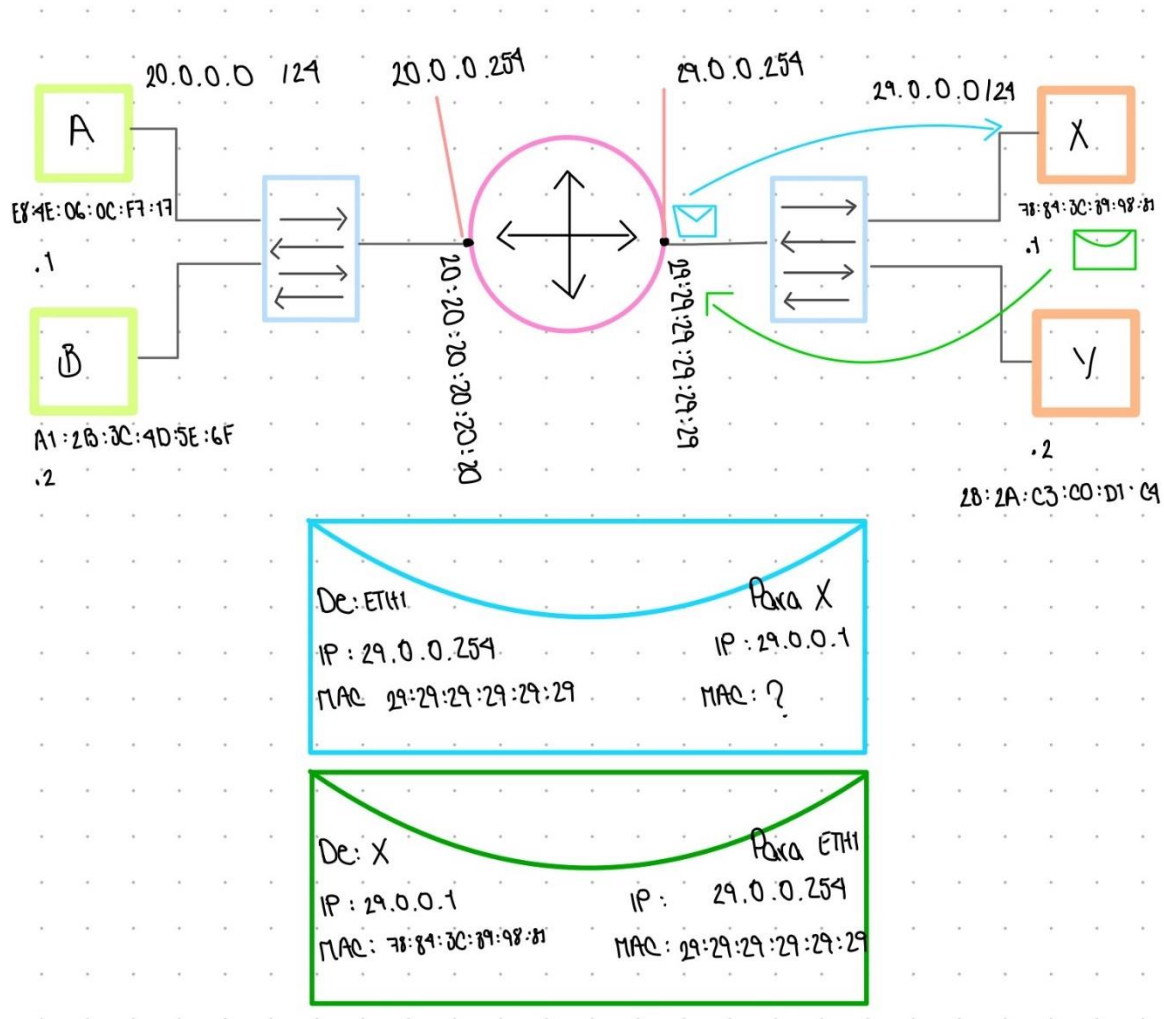
Dado que las IP corresponden al mismo rango como tal el router no interviene en esta operación, pues no es necesario que realice la búsqueda en su tabla, dado que el mensaje no va a salir ya que se encuentran en el mismo rango.

Posteriormente una de todas nuestras terminales conectadas reconoce que le están buscando mediante la IP especificada, es esta la que realiza la respuesta de tipo **unicast**, quedando como (Origen: B Destino: A), dado que se está realizando la comunicación directa en este momento ya se puede mandar la información completa, pues nuestra terminal B evidentemente conoce su MAC por lo cual al momento de realizar la respuesta a A ya la puede mandar



- Explicar de A -> X (realizar el mismo procedimiento que se solicitó en el punto anterior)

Ya que vimos cómo se manda un ARP en la misma RED ahora lo haremos para cuando el destino se



encuentra en otra RED totalmente diferente, para ello lo que haremos será primero mandar una solicitud desde nuestro origen A a todos los miembros de nuestra red mediante broadcast, en este caso como ningún de los miembros de la red tiene la dirección IP que buscamos ninguno va a responder

Ahí es cuando entra nuestro Router con su Default Gateway, como la dirección IP que intentamos buscar no se encuentra en nuestra red interna lo que haremos es mandarle nuestro mensaje al Router mediante su DG, de forma que el mensaje se está enviando a través del Router, esto con el fin de evitar cambios de IP, o incluso cambios de MAC si se cambia el dispositivo, pero la IP sigue siendo la misma

Entonces tendremos un sobre donde nuestro Origen será nuestro dispositivo A, con la dirección IP 20.0.0.1 con la MAC E8:4E:06:0C:F7:17, y nuestra dirección destino será la dirección de IP que buscamos siendo la IP 29.0.0.1 y la dirección MAC que no sabemos

Como no esta en nuestra no obtendremos respuesta directamente de la dirección IP que estamos buscando, en cambio como esto lo tiene que gestionar nuestro Router la respuesta que tendremos de

este será de nuestro Router, entonces el sobre correspondiente que tendremos como respuesta para nuestra primera petición será el siguiente donde tenemos como dirección IP origen 20.0.0.254 que es la DG de nuestro Router y tendremos como MAC la siguiente dirección 20:20:20:20:20:20 que es la dirección MAC de nuestro Router y como dirección destino tendremos la dirección del usuario que hizo la petición en este caso tendremos la dirección IP de A que es 20.0.0.1 y su dirección MAC que es la E8:4E:06:0C:F7:17

Como ya vimos no nos fue posible establecer una comunicación con la IP de nuestro destino, dado que el router detecto que no se encontraba en el host, razón por la cual el router revisa la tabla de enrutamiento y se da cuenta dada la ip que recibió que

1. No se encuentra dentro del mismo rango que aquella que realizo el envío de mensaje
2. Que la ip buscada si está en su tabla de enrutamiento por lo cual pasamos a la interfaz de ETH1 (Primer sobre de la imagen)

Ahora nuestra interfaz ETH1 está realizando una comunicación de tipo **Broadcast** buscando a aquella terminal que posee la **IP** deseada en un inicio, dado que nuevamente tenemos la mayoría de la información necesaria del sobre solo hace falta obtener la dirección MAC, la cual es obtenida mediante la respuesta que se da una vez que la terminal se da cuenta que está siendo buscada, cuando esta responde es momento en el cual realiza la inserción de su dirección MAC para que todo el proceso ya pueda ser realizado.

## Colocar una trama ARP de solicitud (Courier new) y la salida de tu programa (captura de pantalla)

Usaremos como trama solicitud una de las tramas de los ejemplos anteriores siendo la siguiente

```
FF FF FF FF FF FF E8 4E 06 0C F7 17 08 06 00 01
08 00 06 04 00 01 E8 4E 06 0C F7 17 14 00 00 01
00 00 00 00 00 00 1D 00 00 01 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
C:\JomianTC\ESCOM\5 Semestre\Redes de computadoras\Clase 20-10-21>t.exe
Alumnos: Mora Ayala Jose Antonio
          Torres Carrillo Josehf Miguel Angel

=====Cabecera ETHERNET 1=====
MAC Destino: ff:ff:ff:ff:ff:ff:
MAC Origen: e8:4e:06:0c:f7:17:

=====Cabecera ARP=====

TIPO ARP Tamaño de: 2054 bytes, Ethernet 0x08 0x00 0x06 0x04 ARP REQUEST
MAC Origen: e8:4e:06:0c:f7:17
IP Origen: 20.0.0.1
MAC Destino: 00:00:00:00:00:00
IP Destino: 29.0.0.1
```

## Colocar una trama ARP (Courier new) y la salida de tu programa (captura de pantalla)

Usaremos la trama de respuesta cuando enviamos un mensaje de A -> X usando la que va de X -> DG Router

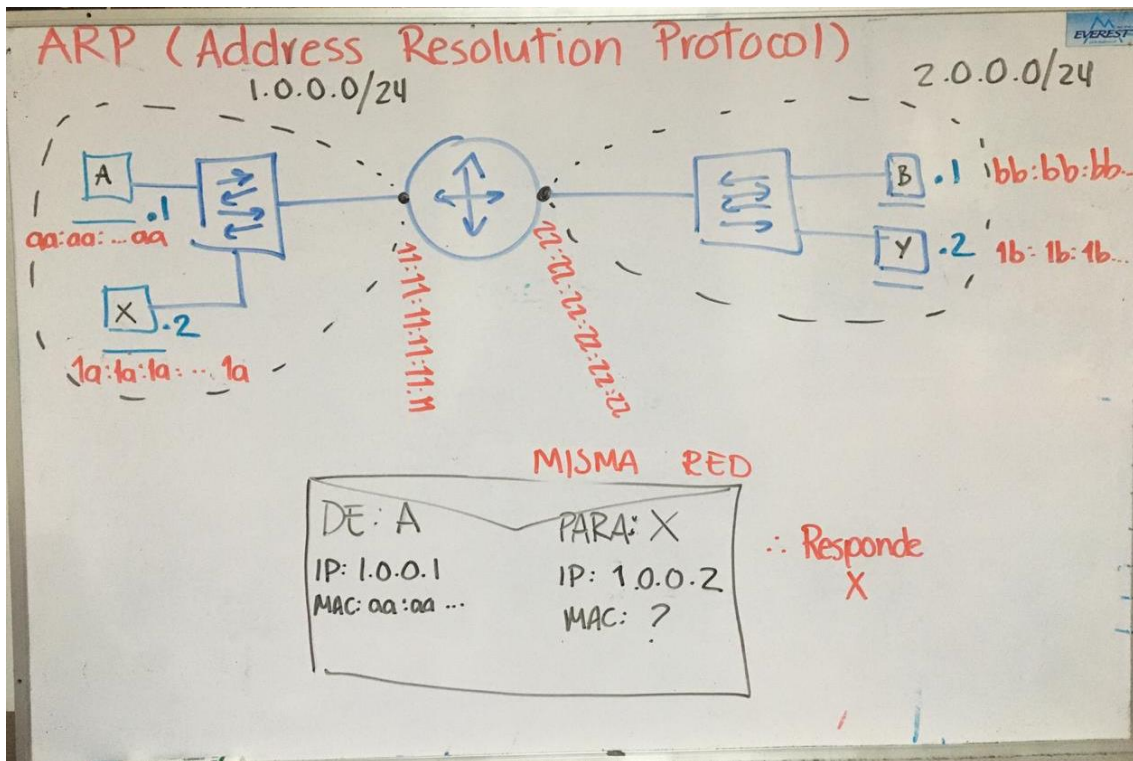
```
29 29 29 29 29 29 78 84 3C 39 98 81 08 06 00 01
08 00 06 04 00 02 78 84 3C 39 98 81 1D 00 00 01
29 29 29 29 29 29 1D 00 00 FE 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
C:\JomianTC\ESCOM\5 Semestre\Redes de computadoras\Clase 20-10-21>gcc TramasLLC.c -o t.exe
C:\JomianTC\ESCOM\5 Semestre\Redes de computadoras\Clase 20-10-21>t.exe
Alumnos: Mora Ayala Jose Antonio
          Torres Carrillo Josehf Miguel Angel

=====Cabecera ETHERNET 1=====
MAC Destino: 29:29:29:29:29:29:
MAC Origen: 78:84:3c:39:98:81:

=====Cabecera ARP=====

TIPO ARP Tamaño de: 2054 bytes, Ethernet 0x08 0x00 0x06 0x04 ARP REPLY
MAC Origen: 78:84:3c:39:98:81
IP Origen: 29.0.0.1
MAC Destino: 29:29:29:29:29:29
IP Destino: 29.0.0.254
```



ARP SOLICITUD (Alicia -> TODOS ) Broadcast

```
FF FF FF FF FF FF AA AA AA AA AA AA 08 06 00 01
08 00 06 04 00 01 AA AA AA AA AA AA 01 00 00 01
00 00 00 00 00 00 01 00 00 02 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
Alumnos: Mora Ayala Jose Antonio
          Torres Carrillo Josehf Miguel Angel

=====Cabecera ETHERNET 3=====
MAC Destino: ff:ff:ff:ff:ff:ff:
MAC Origen: aa:aa:aa:aa:aa:aa:

=====Cabecera ARP=====

TIPO ARP Tamaño de: 2054 bytes, Ethernet 0x08 0x00 0x06 0x04 ARP REQUEST
MAC Origen: aa:aa:aa:aa:aa:aa
IP Origen: 1.0.0.1
MAC Destino: 00:00:00:00:00:00
IP Destino: 1.0.0.2
```

## ARP RESPUESTA (Ximena -> Alicia ) Unicast

```
AA AA AA AA AA AA 1A 1A 1A 1A 1A 1A 08 06 00 01
08 00 06 04 00 02 1A 1A 1A 1A 1A 1A 01 00 00 02
AA AA AA AA AA AA 01 00 00 01 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
=====Cabecera ETHERNET 4=====
MAC Destino: aa:aa:aa:aa:aa:aa:
MAC Origen: 1a:1a:1a:1a:1a:1a:
=====Cabecera ARP=====
TIPO ARP Tamaño de: 2054 bytes, Ethernet 0x08 0x00 0x06 0x04 ARP REPLY
MAC Origen: 1a:1a:1a:1a:1a:1a
IP Origen: 1.0.0.2
MAC Destino: aa:aa:aa:aa:aa:aa
IP Destino: 1.0.0.1
```

## ARP DISTINTAS REDES

### PRIMERA PARTE

#### ARP SOLICITUD (ALICIA -> TODOS)

```
FF FF FF FF FF FF AA AA AA AA AA AA 08 06 00 01
08 00 06 04 00 01 AA AA AA AA AA AA 01 00 00 01
00 00 00 00 00 00 02 00 00 02 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
=====Cabecera ETHERNET 5=====
MAC Destino: ff:ff:ff:ff:ff:ff:
MAC Origen: aa:aa:aa:aa:aa:aa:
=====Cabecera ARP=====
TIPO ARP Tamaño de: 2054 bytes, Ethernet 0x08 0x00 0x06 0x04 ARP REQUEST
MAC Origen: aa:aa:aa:aa:aa:aa
IP Origen: 1.0.0.1
MAC Destino: 00:00:00:00:00:00
IP Destino: 2.0.0.2
```

ARP RESPUESTA (DG ROUTER1 (ETH0) -> ALICIA)

```
AA AA AA AA AA AA 11 11 11 11 11 11 08 06 00 01
08 00 06 04 00 02 11 11 11 11 11 11 01 00 00 FE
AA AA AA AA AA AA 01 00 00 01 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
=====Cabecera ETHERNET 6=====
MAC Destino: aa:aa:aa:aa:aa:aa:
MAC Origen: 11:11:11:11:11:11:
=====Cabecera ARP=====
TIPO ARP Tamaño de: 2054 bytes, Ethernet 0x08 0x00 0x06 0x04 ARP REPLY
MAC Origen: 11:11:11:11:11:11
IP Origen: 1.0.0.254
MAC Destino: aa:aa:aa:aa:aa:aa
IP Destino: 1.0.0.1
```

SEGUNDA PARTE

ARP SOLICITUD (DG ROUTER2 (ETH1) -> TODOS)

```
FF FF FF FF FF FF 22 22 22 22 22 22 08 06 00 01
08 00 06 04 00 01 22 22 22 22 22 22 02 00 00 FE
00 00 00 00 00 00 02 00 00 02 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
=====Cabecera ETHERNET 7=====
MAC Destino: ff:ff:ff:ff:ff:ff:
MAC Origen: 22:22:22:22:22:22:
=====Cabecera ARP=====
TIPO ARP Tamaño de: 2054 bytes, Ethernet 0x08 0x00 0x06 0x04 ARP REQUEST
MAC Origen: 22:22:22:22:22:22
IP Origen: 2.0.0.254
MAC Destino: 00:00:00:00:00:00
IP Destino: 2.0.0.2
```

ARP RESPUESTA (YOYIS -> DG ROUTER2 (ETH1))

```

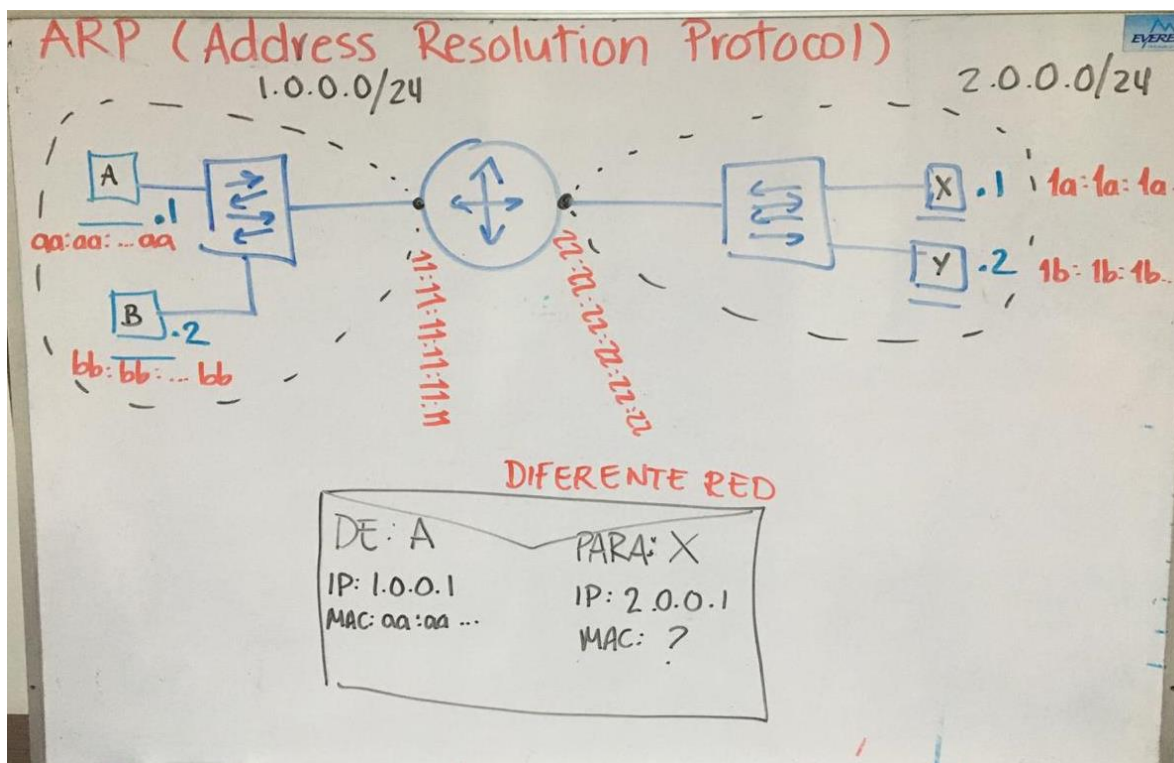
22 22 22 22 22 22 1B 1B 1B 1B 1B 1B 08 06 00 01
08 00 06 04 00 02 1B 1B 1B 1B 1B 1B 02 00 00 02
22 22 22 22 22 22 02 00 00 FE 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

```

=====Cabecera ETHERNET 8=====
MAC Destino: 22:22:22:22:22:22:
MAC Origen: 1b:1b:1b:1b:1b:1b:
=====Cabecera ARP=====
TIPO ARP Tamaño de: 2054 bytes, Ethernet 0x08 0x00 0x06 0x04 ARP REPLY
MAC Origen: 1b:1b:1b:1b:1b:1b
IP Origen: 2.0.0.2
MAC Destino: 22:22:22:22:22:22
IP Destino: 2.0.0.254

```



Construir las primeras tramas (DIFERENTE RED – salto 1)

ARP SOLICITUD (Alicia -> TODOS )

```
FF FF FF FF FF FF AA AA AA AA AA AA 08 06 00 01
08 00 06 04 00 01 AA AA AA AA AA AA 01 00 00 01
00 00 00 00 00 00 02 00 00 01 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
=====Cabecera ETHERNET 9=====
MAC Destino: ff:ff:ff:ff:ff:ff:
MAC Origen: aa:aa:aa:aa:aa:aa:
=====Cabecera ARP=====
TIPO ARP Tamaño de: 2054 bytes, Ethernet 0x08 0x00 0x06 0x04 ARP REQUEST
MAC Origen: aa:aa:aa:aa:aa:aa
IP Origen: 1.0.0.1
MAC Destino: 00:00:00:00:00:00
IP Destino: 2.0.0.1
```

ARP RESPUESTA (DG router -> Alicia )

```
AA AA AA AA AA AA 11 11 11 11 11 11 08 06 00 01
08 00 06 04 00 02 11 11 11 11 11 11 01 00 00 02
AA AA AA AA AA AA 01 00 00 01 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
=====Cabecera ETHERNET 10=====
MAC Destino: aa:aa:aa:aa:aa:aa:
MAC Origen: 11:11:11:11:11:11:
=====Cabecera ARP=====
TIPO ARP Tamaño de: 2054 bytes, Ethernet 0x08 0x00 0x06 0x04 ARP REPLY
MAC Origen: 11:11:11:11:11:11
IP Origen: 1.0.0.2
MAC Destino: aa:aa:aa:aa:aa:aa
IP Destino: 1.0.0.1
```



Construir las primeras tramas (DIFERENTE RED – salto 2)

ARP SOLICITUD (router -> TODOS en red 2.0.0.0 )

```
FF FF FF FF FF FF 22 22 22 22 22 22 08 06 00 01
08 00 06 04 00 01 22 22 22 22 22 22 02 00 00 FE
00 00 00 00 00 00 02 00 00 01 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
=====Cabecera ETHERNET 11=====
MAC Destino: ff:ff:ff:ff:ff:ff:
MAC Origen: 22:22:22:22:22:22:
=====Cabecera ARP=====
TIPO ARP Tamaño de: 2054 bytes, Ethernet 0x08 0x00 0x06 0x04 ARP REQUEST
MAC Origen: 22:22:22:22:22:22
IP Origen: 2.0.0.254
MAC Destino: 00:00:00:00:00:00
IP Destino: 2.0.0.1
```

ARP RESPUESTA (Ximena -> DG router )

```
22 22 22 22 22 22 1A 1A 1A 1A 1A 1A 08 06 00 01
08 00 06 04 00 02 1A 1A 1A 1A 1A 1A 02 00 00 01
22 22 22 22 22 22 02 00 00 FE 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
=====Cabecera ETHERNET 12=====
MAC Destino: 22:22:22:22:22:22:
MAC Origen: 1a:1a:1a:1a:1a:1a:
=====Cabecera ARP=====
TIPO ARP Tamaño de: 2054 bytes, Ethernet 0x08 0x00 0x06 0x04 ARP REPLY
MAC Origen: 1a:1a:1a:1a:1a:1a
IP Origen: 2.0.0.1
MAC Destino: 22:22:22:22:22:22
IP Destino: 2.0.0.254
```

- Código y salidas después de probar sus tramas.

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

void analizaTrama(unsigned char T[]);

void analizaLLC(unsigned char T[]);
void analizaARP(unsigned char T[]);
void imprimeTI(unsigned char T[]);
void imprimeTS(unsigned char T[]);
void imprimeTU(unsigned char T[]);

int PoF(unsigned char T[]);

void main(int argc, char const *argv[]){

//Tramas Reporte

///  

    unsigned char  

T[][200]={0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xE8,0x4E,0x06,0x0C,0xF7,0x17,0x  

08,0x06,0x00,0x01, //Inventadas  

0x08,0x00,0x06,0x04,0x00,0x01,0xE8,0x4E,0x06,0x0C,0xF7,0x17,0x14,0x00,0x0  

0,0x01,  

0x00,0x00,0x00,0x00,0x00,0x00,0x1D,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x0  

0,0x00,  

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0  

0,0x00},  

    {0x29,0x29,0x29,0x29,0x29,0x29,0x78,0x84,0x3C,0x39,0x98,0x81,0x08,0  

x06,0x00,0x01, //Inventadas  

0x08,0x00,0x06,0x04,0x00,0x02,0x78,0x84,0x3C,0x39,0x98,0x81,0x1D,0x00,0x0  

0,0x01,  

0x29,0x29,0x29,0x29,0x29,0x29,0x1D,0x00,0x00,0xFE,0x00,0x00,0x00,0x00,0x0  

0,0x00,  

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0  

0,0x00},  

    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0x08,0  

x06,0x00,0x01, //en clase  

0x08,0x00,0x06,0x04,0x00,0x01,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0x01,0x00,0x0  

0,0x01,  

0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x02,0x00,0x00,0x00,0x00,0x0  

0,0x00,
```

```

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00},

    {0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0x1A,0x1A,0x1A,0x1A,0x1A,0x1A,0x08,0
x06,0x00,0x01, //en clase

0x08,0x00,0x06,0x04,0x00,0x02,0x1A,0x1A,0x1A,0x1A,0x1A,0x1A,0x01,0x00,0x0
0,0x02,

0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0x01,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00},

    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0x08,0
x06,0x00,0x01, //tramas clase P1S

0x08,0x00,0x06,0x04,0x00,0x01,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0x01,0x00,0x0
0,0x01,

0x00,0x00,0x00,0x00,0x00,0x00,0x02,0x00,0x00,0x02,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00},

    {0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0x11,0x11,0x11,0x11,0x11,0x11,0x08,0
x06,0x00,0x01, //tramas clase P1R

0x08,0x00,0x06,0x04,0x00,0x02,0x11,0x11,0x11,0x11,0x11,0x11,0x01,0x00,0x0
0,0xFE,

0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0x01,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00},

    {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x22,0x22,0x22,0x22,0x22,0x22,0x08,0
x06,0x00,0x01, //tramas clase P2S

0x08,0x00,0x06,0x04,0x00,0x01,0x22,0x22,0x22,0x22,0x22,0x22,0x02,0x00,0x0
0,0xFE,

0x00,0x00,0x00,0x00,0x00,0x00,0x02,0x00,0x00,0x02,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00},

```

```

        {0x22,0x22,0x22,0x22,0x22,0x22,0x1B,0x1B,0x1B,0x1B,0x1B,0x1B,0x08,0
x06,0x00,0x01, //tramas clase P2R

0x08,0x00,0x06,0x04,0x00,0x02,0x1B,0x1B,0x1B,0x1B,0x1B,0x1B,0x02,0x00,0x0
0,0x02,

0x22,0x22,0x22,0x22,0x22,0x22,0x02,0x00,0x00,0xFE,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00}},

        {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0x08,0
x06,0x00,0x01, //Ultimas 4 tramas del reporte

0x08,0x00,0x06,0x04,0x00,0x01,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0x01,0x00,0x0
0,0x01,

0x00,0x00,0x00,0x00,0x00,0x00,0x02,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00}},

        {0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0x11,0x11,0x11,0x11,0x11,0x11,0x08,0
x06,0x00,0x01, //Ultimas 4 tramas del reporte

0x08,0x00,0x06,0x04,0x00,0x02,0x11,0x11,0x11,0x11,0x11,0x11,0x01,0x00,0x0
0,0x02,

0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0x01,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00}},

        {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x22,0x22,0x22,0x22,0x22,0x22,0x08,0
x06,0x00,0x01, //Ultimas 4 tramas del reporte

0x08,0x00,0x06,0x04,0x00,0x01,0x22,0x22,0x22,0x22,0x22,0x22,0x02,0x00,0x0
0,0xFE,

0x00,0x00,0x00,0x00,0x00,0x00,0x02,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00}},

        {0x22,0x22,0x22,0x22,0x22,0x22,0x1A,0x1A,0x1A,0x1A,0x1A,0x1A,0x08,0
x06,0x00,0x01, //Ultimas 4 tramas del reporte

```

```

0x08,0x00,0x06,0x04,0x00,0x02,0x1A,0x1A,0x1A,0x1A,0x1A,0x1A,0x02,0x00,0x0
0,0x01,

0x22,0x22,0x22,0x22,0x22,0x22,0x02,0x00,0x00,0xFE,0x00,0x00,0x00,0x00,0x0
0,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00}

};

/**/

printf("Alumnos: Mora Ayala Jose Antonio\n\t Torres Carrillo Josehf
Miguel Angel\n\n");

for (unsigned char i = 8; i < 12; i++){

    printf("====Cabecera ETHERNET
%d====\n", i+1);
    analizaTrama(T[i]);
}

void analizaTrama (unsigned char T[]){

    unsigned short int tot = T[12]<<8 | T[13];

    printf("MAC Destino: %.2x:%.2x:%.2x:%.2x:%.2x:%.2x:\n\n",
T[0],T[1],T[2],T[3],T[4],T[5]);
    printf("MAC Origen: %.2x:%.2x:%.2x:%.2x:%.2x:%.2x:\n\n",
T[6],T[7],T[8],T[9],T[10],T[11]);

    if (tot<1500){

        printf("====Cabecera
LLC====\n\n");
        printf("Tama%co de cabecera LLC: %d bytes\n\n", 164, tot);
        analizaLLC(T);
    }
    else if (tot == 2048){

        printf("TIPO IP Tama%co de: %d bytes\n", 164, tot);
    }
    else if (tot == 2054){

        printf("====Cabecera
ARP====\n\n");
        printf("TIPO ARP Tama%co de: %d bytes, ", 164, tot);
        analizaARP(T);
    }
    else{

        printf("OTRO TIPO \n");
    }

    printf("\n\n");
}

```

```

}

void analizaLLC (unsigned char T[]){

    switch (T[16]&3){

        case 0:
            imprimeTI(T);
            break;

        case 1:
            imprimeTS(T);
            break;

        case 2:
            imprimeTI(T);
            break;

        case 3:
            imprimeTU(T);
            break;

        default:
            printf("ERROR\n");
    }
}

void imprimeTI (unsigned char T[]){

    printf("T- I, N(s)=%d, N(r)=%d", T[16]>>1, T[17]>>1);

    PoF(T);
}

void imprimeTS (unsigned char T[]){

    char ss[][4] = {"RR", "RNR", "REJ", "SREJ"};

    printf("T-S %s", ss[(T[16]>>2)&3]);

    PoF(T);
}

void imprimeTU (unsigned char T[]){

    char uc[][6]={ "UI", "SIM", "-", "SARM", "UP", "-", "-", "SABM",
                    "DISC", "-", "-", "SARME", "-", "-", "-", "SAMBE",
                    "SNRM", "-", "-", "RSET", "-", "-", "-", "XID", "-",
                    "-", "-", "SRME", "-", "-", "-", "-"};

    char ur[][6]={ "UI", "RIM", "-", "DM", "-", "-", "-", "-", "RD", "-",
                    "-", "-", "UA", "-", "-", "-", "-", "FRMR", "-", "-",
                    "-", "-", "-", "XID", "-", "-", "-", "-", "-", "-", "-"};

    printf("T-U");

    if (T[16] & 16){

```

```

        if (T[15] & 1)
            printf(" -F %s", ur[((T[16]>>2)&3) | ((T[16]>>3)&28)]);
        else
            printf(" -P, %s",uc[((T[16]>>2)&3) | ((T[16]>>3)&28)]);
    }
}

int PoF(unsigned char T[]){

    if (T[17] & 1){

        if (T[15]&1)
            printf(" -F");
        else
            printf(" -P");
    }
}

void analizaARP (unsigned char T[]){

    if (T[15]&1) printf("Ethernet ");
    else if (T[15]&6) printf("IEEE 802 ");
    else if (T[15]&15) printf("Frame Relay ");
    else if (T[15]&20) printf("ATM ");
    else printf("Otro ");

    if (T[16]== 8 && T[17] == 0) printf("0x08 0x00 ");
    else printf("Otro ");

    if (T[18]== 6 && T[19] == 4) printf("0x06 0x04 ");
    else printf("Otro ");

    if (T[21]&1) printf("ARP REQUEST\n\n");
    else if (T[21]&2) printf("ARP REPLY\n\n");
    else if (T[21]&3 || T[21]&8) printf("INVERSE ARP REQUEST\n\n");
    else if (T[21]&4 || T[21]&9) printf("INVERSE ARP REPLY\n\n");
    else printf("0x%.2x\n\n", T[21]);

    printf("MAC Origen: %.2x:%.2x:%.2x:%.2x:%.2x:%.2x\n\n",
T[22],T[23],T[24],T[25],T[26],T[27]);
    printf("IP Origen: %d.%d.%d.%d\n\n", T[28],T[29],T[30],T[31]);

    printf("MAC Destino: %.2x:%.2x:%.2x:%.2x:%.2x:%.2x\n\n",
T[32],T[33],T[34],T[35],T[36],T[37]);
    printf("IP Destino: %d.%d.%d.%d", T[38],T[39],T[40],T[41]);

}

```

## Protocolo IP

Incluir un párrafo de definición de siglas y principal funcionamiento del protocolo, definir en que capa trabaja y su función(es) principal(es)

El protocolo IP o **Internet Protocol** en español **Protocolo de Internet** es un protocolo que permite la funcionalidad de red que hace posible la existencia de redes a gran escala además de ser enrutable, este protocolo conforma la capa Internet de la arquitectura TCP/IP o en la capa de Red del modelo OSI

- Imagen de la Cabecera completa

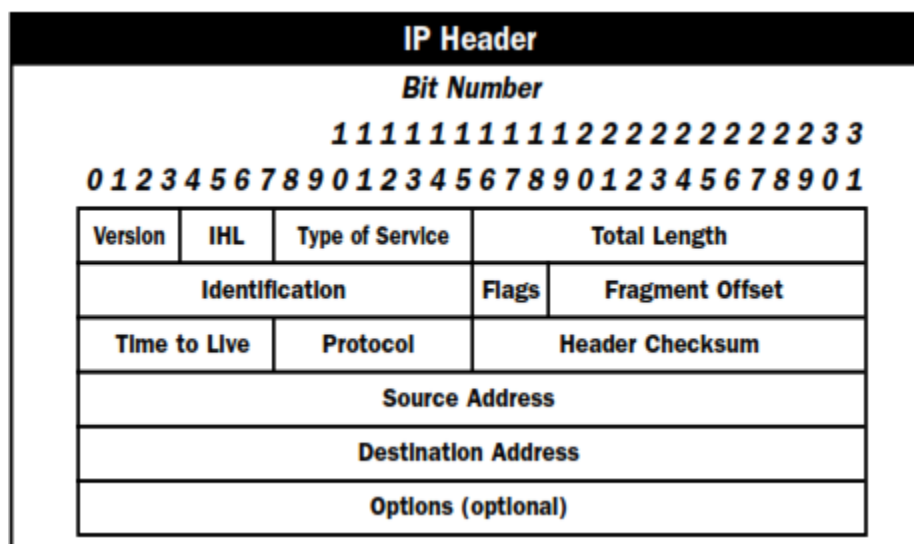


Imagen Obtenida del Documento proporcionado por la profesora, Creado por The Sans Technology Institute (STI)

- Descripción breve de cada campo

Los campos que conforman la cabecera IP son los siguientes

1. Versión - 4 bits  
Es un campo utilizado para indicar la versión del encabezado para este encabezado es 4 para redes corporativas
2. IHL - 4 bits  
Indica el tamaño total del encabezado IP, el valor máximo de este parámetro es 5 y el mínimo es 5
3. Tipo de Servicio - 1 byte  
Se utiliza para indicar la calidad de servicio con la que el datagrama será entregado
4. Tamaño total - 2 bytes  
Se usa para indicar el tamaño del datagrama IP
5. Identificador - 2 bytes  
Se usa para el paquete IP específico enviado entre el nodo origen y destino



6. Bandera – 3 bits  
Contiene indicadores de fragmentación, siendo DF para no fragmentar y MF para más Fragmentos
7. Desplazamiento de fragmento – 13 bits  
Indica el desplazamiento del lugar donde empieza ese fragmento en relación con los datos IP original no fragmentada
8. Tiempo de vida – 1 byte  
Es un numero que indica los enlaces inversos, el Host emisor lo define como el máximo numero de ruteadores por los que puede viajar el datagrama
9. Protocolo – 1 byte  
Indica el protocolo o subcapa en la carga IP estos pueden ser ICMP, TCP o UDP
10. Checksum IP – 2 bytes  
Suma de comprobación que realiza una comprobación de la integridad del encabezado IP
11. Dirección IP Destino - 4 bytes  
Indica la dirección IP de Host de origen
12. Dirección IP Origen - 4 bytes  
Indica la dirección IP de Host de destino
13. Opciones y relleno – variable  
Campos adicionales anexados al encabezado estándar de 20 bytes pueden ser tener un tamaño de 0 bytes hasta 40 bytes máximo

- Mapa de memoria (como imagen)

Cabecera IP	t[14]	1	1	0	0	0	1	0	1	Versión T[14]>>4;	IHL=(T[14]&15)*4 [bytes]
	t[15]	0	1	0	0	1	0	0	0	Tipo de Servicio	
	t[16]	1	1	1	1	1	1	1	1	Tamaño total	TamTotal= t[16 ]<<8   T[17] (bytes)
	t[17]	1	1	1	1	1	1	1	1		
	t[18]									identificador	Id= t[18 ]<<8   T[19]
	t[19]										
	t[20]	x	D	M	0	0	0	0	0	BANDERAS	Banderas
	t[21]	1	0	1	1	1	1	1	0	DESPLAZAMIENTO	Desplazamiento
	t[22]									Tiempo de Vida	Tvida=T[22]
	t[23]	0	1	1	1	0	0	1	1	Protocolo	Switch(T[23]){case 1, case2, ... , case115}
	t[24]									Checksum	printf("0x%.2x,0x%.2x", T[24],T[25])
	t[25]										
	t[26]									Direccion IP Origen	Imprimiendo valores desde T[26] a T[29] printf("%d.%d.%d.%d", T[26], T[27], T[28],T[29])
	t[27]										
	t[28]										
	t[29]										
	t[30]									Direccion IP Destino	Imprimiendo valores: T[30] a T[33] printf("%d.%d.%d.%d", T[30], T[31], T[32], T[33])
	t[31]										
	t[32]										
	t[33]										
	t[34]										
	t[35]										
	t[36]										
	t[37]										
	T[14+IHL-1]										

Sentencia SWITCH	
case 1	printf("IGMP")
case 2	printf("IGMP")
case 6	printf("TCP")
case 9	printf("IGRP")
case 17	printf("UDP")
case 47	printf("GRE")
case 50	printf("ESP")
case 51	printf("AH")
case 57	printf("SKIP")
case 88	printf("EIGRP")
case 89	printf("OSPF")
case 115	printf("L2TP")
Desplazamiento	
T[20]<<3	Quitando los bits de XDM
T[20]>>3	Colocando 0 en las posiciones
T[20]<<8   T[21]	Realizando suma con siguiente byte
Banderas	
X=	(T[20]>>5)&4
D=	(T[20]>>5)&2
M=	(T[20]>>5)&1

- Responder a las siguientes preguntas
- Tamaño mínimo de la cabecera IP  
**5 (20 bytes)**
- Tamaño máximo de la cabecera IP  
**15 (60 bytes)**
- Tamaño mínimo para las opciones IP  
**0**
- Tamaño máximo para las opciones IP  
**40 bytes**
- ¿Cómo saber si un paquete IP tiene opciones?

Para verificar si una trama tiene opciones tenemos que recurrir al campo IHL el cual nos servirá para determinar el tamaño total de nuestra cabecera si el campo *IHL* < 20 (tamaño mínimo) quiere decir que no tenemos opciones en la cabecera en cambio si este campo es *IHL* > 20 quiere decir que en la trama tendremos campos de opciones que puede llegar como máximo a 40 bytes teniendo un tamaño total de la trama igual a 60 bytes

- Poner los 4 incisos solicitados en classroom así como el código correspondiente (hacer uso de algún editor de código) y las salidas después de la ejecución de su programa.

## Instrucciones

En un archivo de word incluye los 4 ejercicios sobre el protocolo IP

- Pega el código correspondiente resultante
- Modifica tramas IP (crea) para que entren a las condiciones solicitadas
- Pon al menos 5 salidas (1 por cada inciso) como captura de pantalla.
- Una con checksum correcto y otra con incorrecto

## Tramas Probadas

```
//Trama IP con opciones de tipo ICMP - Imprimir las opciones en hexadecimal

unsigned char T1[] = {0x00,0x1f,0x45,0x9d,0x1e,0xa2,
//MAC Destino T[0] - T[5]
//MAC Origen T[6] - T[11]
0x00,0x23,0x8b,0x46,0xe9,0xad,
//Tipo T[12] - T[13]
0x08,0x00,
//Version/IHL T[14]
0x4F,
//Tipo de servicio T[15]
0x00,
//Tamano total T[16] - T[17]
0x80,0x42,
//Identificador T[18] - T[19]
0x04,0x55,
//Bandera/ Desplazamiento Fragmento T[20] / T[20] - T[21]
0x34,0x11,
//Tiempo de vida T[22]
0x80,
//Protocolo T[23]
0x01,
//Checksum T[24] - T[25]
0x6b,0xf0,
//IP Origen T[26] - T[29]
0x94,0xcc,0x39,0xcb,
//Ip Destino T[30] - T[33]
0x94,0xcc,0x67,0x02,
//Opciones y relleno T[34] - T[73]
0xaa,0xbb,0xcc,0xdd,0x04,0x0c,0x00,0x35,0x00,
0x2e,0x85,0x7c,0xe2,0x1a,0x01,0x00,0x00,0x01,
0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x77,0x77,
0x77,0x03,0x69,0x73,0x63,0x05,0x65,0x73,0x63,
0x6f,0x6d,0x03,0x29};
//MAC Destino T[0] - T[5]
```

```

unsigned char T2[] = {0x00,0x1f,0x45,0x9d,0x1e,0xa2,
                      //MAC Origen T[6] - T[11]
                      0x00,0x23,0x8b,0x46,0xe9,0xad,
                      //Tipo T[12] - T[13]
                      0x08,0x00,
                      //Version/IHL T[14]
                      0x40,
                      //Tipo de servicio T[15]
                      0x02,
                      //Tamano total T[16] - T[17]
                      0x80,0x42,
                      //Identificador T[18] - T[19]
                      0x04,0x55,
                      //Bandera/ Desplazamiento Fragmento
                      0x34,0x11,
                      //Tiempo de vida T[22]
                      0x80,
                      //Protocolo T[23]
                      0x01,
                      //Checksum T[24] - T[25]
                      0xbc,0xec,
                      //IP Origen T[26] - T[29]
                      0x94,0xcc,0x39,0xcb,
                      //Ip Destino T[30] - T[33]
                      0x94,0xcc,0x67,0x02,
                      //Opciones y relleno T[34] - T[73]
                      };

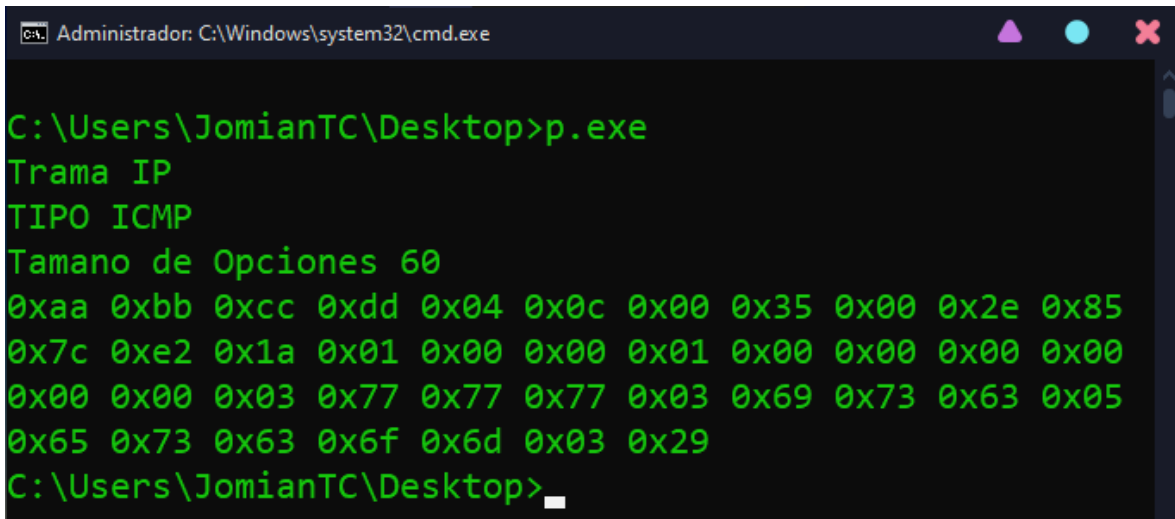
unsigned char T4[] = {0x00,0x1f,0x45,0x9d,0x1e,0xa2,
                      //MAC Origen T[6] - T[11]
                      0x00,0x23,0x8b,0x46,0xe9,0xad,
                      //Tipo T[12] - T[13]
                      0x08,0x00,
                      //Version/IHL T[14]
                      0x40,
                      //Tipo de servicio T[15]
                      0x02,
                      //Tamano total T[16] - T[17]
                      0x80,0x42,
                      //Identificador T[18] - T[19]
                      0x04,0x55,
                      //Bandera/ Desplazamiento Fragmento T[20] / T[20] - T[21]
                      0x34,0x11,
                      //Tiempo de vida T[22]
                      0x80,
                      //Protocolo T[23]
                      0x11,
                      //Checksum T[24] - T[25]
                      0xbc,0xec,
                      //IP Origen T[26] - T[29]
                      0x94,0xcc,0x39,0xcb,
                      //Ip Destino T[30] - T[33]
                      0x94,0xcc,0x67,0x02,
                      //Opciones y relleno T[34] - T[73]
                      };

```

## Una trama IP con opciones ICMP

Imprimir las opciones en hexadecimal

### Captura de pantalla



```
Administrador: C:\Windows\system32\cmd.exe

C:\Users\JomianTC\Desktop>p.exe
Trama IP
TIPO ICMP
Tamano de Opciones 60
0xaa 0xbb 0xcc 0xdd 0x04 0x0c 0x00 0x35 0x00 0x2e 0x85
0x7c 0xe2 0x1a 0x01 0x00 0x00 0x01 0x00 0x00 0x00 0x00
0x00 0x00 0x03 0x77 0x77 0x77 0x03 0x69 0x73 0x63 0x05
0x65 0x73 0x63 0x6f 0x6d 0x03 0x29
C:\Users\JomianTC\Desktop>
```

### Código del programa

```
void programal() {

    unsigned short int tot = T1[12]<<8 | T1[13];

    if (tot == 2048){

        printf("Trama IP \n");

        if (T1[23] == 1) printf("TIPO ICMP \n");
        else printf("TIPO OTRO\n");

        unsigned char IHL = (T1[14]&15)*4;

        if (IHL > 20){

            printf("Tamano de Opciones %d\n", IHL);

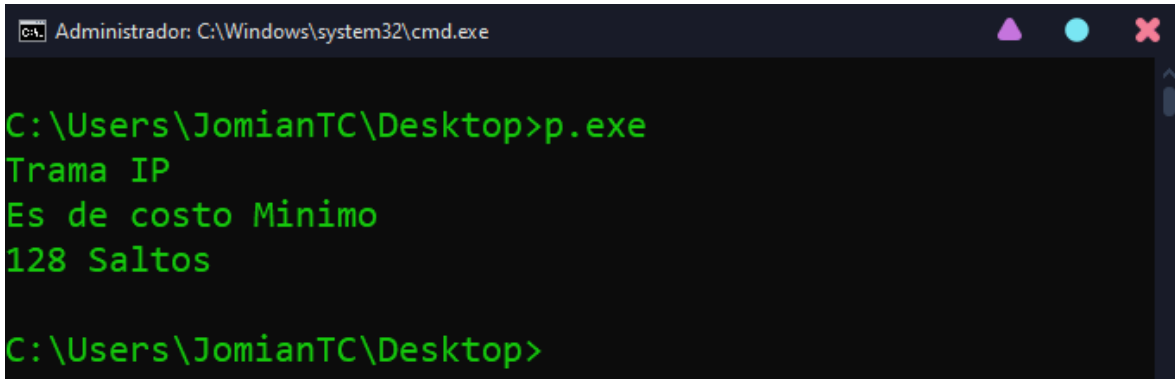
            for (unsigned char i = 34; i <= (14 + IHL -1); i++)
                printf("0x%.2x ", T1[i]);
            }
        else printf("No tiene Opciones\n");
    }
    else{

        printf("No es una trama IP\n");
    }
}
```

## Una trama IP de costo mínimo

Imprimir TTL

### Captura de pantalla



```
Administrador: C:\Windows\system32\cmd.exe

C:\Users\JomianTC\Desktop>p.exe
Trama IP
Es de costo Minimo
128 Saltos

C:\Users\JomianTC\Desktop>
```

### Código del programa

```
void programa2(){

    unsigned short int tot = T2[12]<<8 | T2[13];

    if (tot == 2048){

        printf("Trama IP \n");

        if (T2[15]&2){

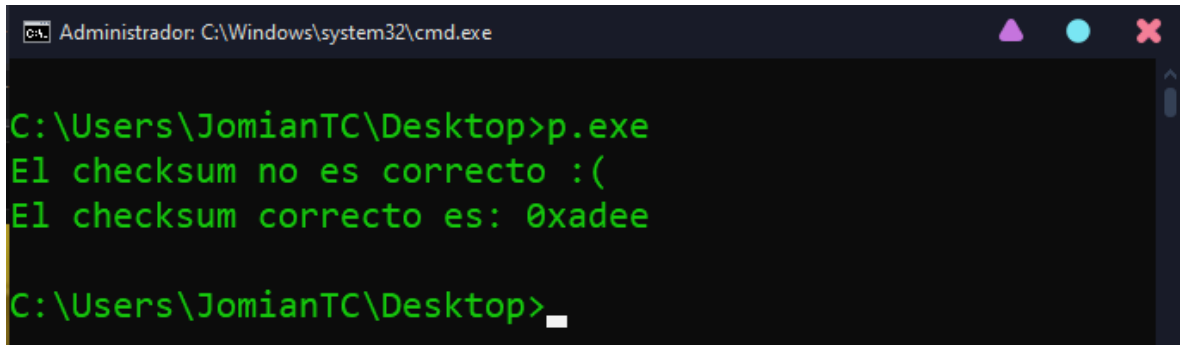
            printf("Es de costo Minimo\n");
            printf("%d Saltos\n", T2[22]);
        }
        else printf("Otro\n");
    }
    else{

        printf("No es una trama IP\n");
    }
}
```

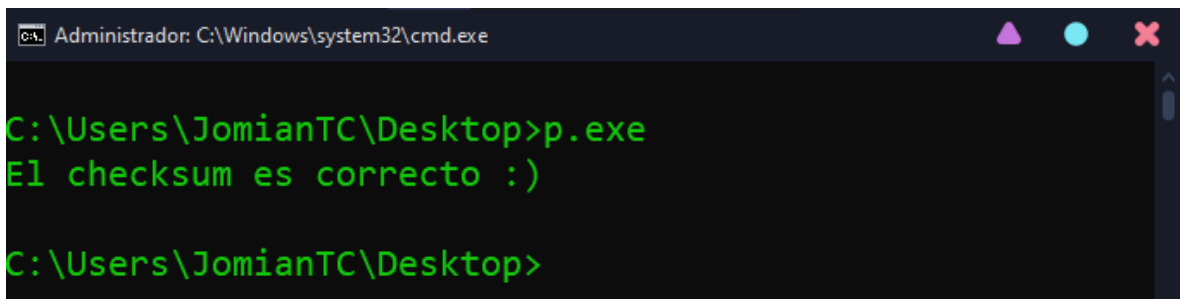
## Verificar el checksum de las tramas IP

- En caso de que este correcto imprimir :) en caso de que sea incorrecto ☹ e imprimir el checksum correcto. Imprimir las opciones en hexadecimal
- Llamar a su función Checksum

## Captura de pantalla



A screenshot of a Windows command prompt window titled "Administrador: C:\Windows\system32\cmd.exe". The prompt shows the command `p.exe` being executed. The output is displayed in green text: `El checksum no es correcto :(` followed by `El checksum correcto es: 0xadee`. The prompt is ready for the next command.



A screenshot of a Windows command prompt window titled "Administrador: C:\Windows\system32\cmd.exe". The prompt shows the command `p.exe` being executed. The output is displayed in green text: `El checksum es correcto :)`. The prompt is ready for the next command.

## Código del programa

```
void checksum() {  
  
    size_t n = sizeof(T1)/sizeof(T1[0]);  
  
    unsigned char T3Aux[n-14];  
  
    for (unsigned char i = 0; i < n-14; i++)  
        T3Aux[i] = T1[i+14];  
  
    unsigned int suma = 0x00, acarreo = 0x00;  
    unsigned char i, tam;  
  
    i = 0x00;  
    tam = 0x14;  
    // Agarrando valores pares  
    while (tam > 1)  
    {  
        acarreo = (((T3Aux[i] << 8) & 0xFF00) | ((T3Aux[i + 1]) &  
0xFF));  
        suma += acarreo;  
        if ((suma & 0xFFFF0000) > 0)
```

```

        {
            suma = suma & 0xFFFF;
            suma += 1;
        }
        i += 2;
        tam -= 2;
    }
    suma = ~suma;
    suma = suma & 0xFFFF;
    if (suma == 0)
    {
        printf("El checksum es correcto :)\n");
    }
    else
    {
        printf("El checksum no es correcto :(\n");
        suma = 0x00, acarreo = 0x00;
        i = 0x00;
        tam = 0x14;
        T3Aux[10] = 0;
        T3Aux[11] = 0;
        while (tam > 1)
        {
            acarreo = (((T3Aux[i] << 8) & 0xFF00) | ((T3Aux[i + 1]) &
0xFF));
            suma += acarreo;
            if ((suma & 0xFFFF0000) > 0)
            {
                suma = suma & 0xFFFF;
                suma += 1;
            }
            i += 2;
            tam -= 2;
        }

        suma = ~suma;
        suma = suma & 0xFFFF;
        printf("El checksum correcto es: 0x%.4x\n", suma);
    }
}

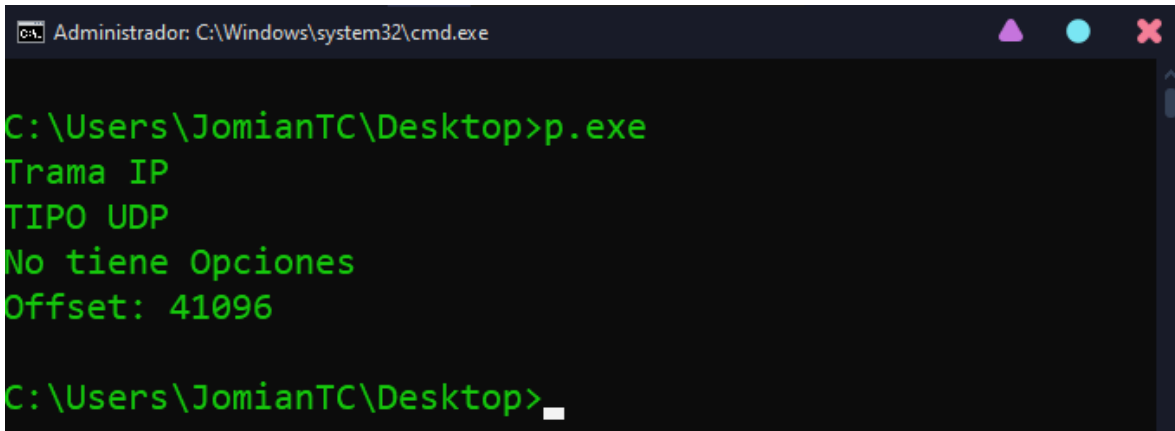
```



## Una trama UDP cuyo encapsulado IP no tenía opciones

Devolver valor del offset en decimal

### Captura de pantalla



A screenshot of a Windows command prompt window titled "Administrador: C:\Windows\system32\cmd.exe". The window has a dark background and green text. The output of the program is as follows:

```
C:\Users\JomianTC\Desktop>p.exe
Trama IP
TIPO UDP
No tiene Opciones
Offset: 41096

C:\Users\JomianTC\Desktop>_
```

### Código del programa

```
void programa4() {

    unsigned short int tot = T1[12]<<8 | T1[13];

    if (tot == 2048){

        printf("Trama IP \n");

        if (T4[23] == 17) printf("TIPO UDP \n");
        else printf("TIPO OTRO\n");

        unsigned char IHL = (T4[14]&15)*4;

        if (IHL > 20){

            printf("Tiene Opciones %d\n", IHL);
        }
        else printf("No tiene Opciones\n");

        printf("Offset: %d\n", (((T4[20]&31)<<8) | T4[21])*8);
    }
    else{

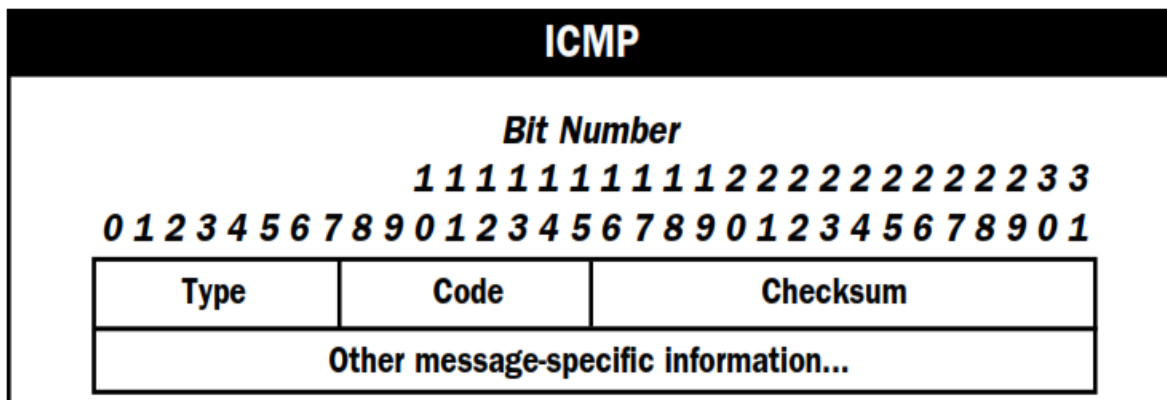
        printf("No es una trama IP\n");
    }
}
```

## Protocolo ICMP

Incluir un párrafo de definición de siglas y principal funcionamiento del protocolo, definir en que capa trabaja y su función(es) principal(es)

Las siglas ICMP significan “**Internet Control Message Protocol (Protocolo de Mensajes de Control de Internet)**” el cual entre sus principales funciones tiene proporcionar la información de las condiciones de error y control a diferencia de IP el cual entrega datagramas de un extremo a otro, este si proporciona la utilidad de poder brindar información acerca de la existencia de errores de enrutamiento o entrega

- ICMP es un protocolo extensible que también proporciona funciones para **comprobar la conectividad IP** y ayudar en la configuración automática de los HOST
  - **Solo se envían para el primer fragmento de un datagrama**
  - Los mensajes de ICMP no se envían para problemas encontrados por mensajes de error ICMP o por datagramas de difusión o multidifusión1
- Imagen de la Cabecera completa



Imágen Obtenida del Documento proporcionado por la profesora, Creado por The Sans Technology Institute (STI)

- Mapa de memoria **general** (como excel en verde)  
Poner la tabla de los Tipos mas representativos (sacar de mis diapositivas)

Cabecera ICMP	T[ICHL+14]							Tipo	
	T[ICHL+15]							Código	
	T[ICHL+16]							Checksum	t=t[ICHL+16]<<8   t[ICHL+17]
	T[ICHL+17]								
	T[ICHL+18]								
	T[ICHL+19]							Datos específicos del tipo	Variable
	T[ICHL+20]								
...									

- Mapa de memoria **ECO** (como en excel verde agua)

CABECERA ICMP ECO (SOLICITUD O RESPUESTA)												
Cabecera [CMP]	T[14+IHL]											Tipo 0=resp 8= solicitud
	T[14+IHL+1]											Código = 0
	T[14+IHL+2]											Checksum if(t[14+IHL]==8 && t[14+IHL+1]==0) {"Solicitud ECO"}
	T[14+IHL+3]											
	T[14+IHL+4]											Identificador if(t[14+IHL]==0 && t[14+IHL+1]==0) { Respuesta ECO}
	T[14+IHL+5]											
	T[14+IHL+6]											Num. Secuencia
	T[14+IHL+7]											
	T[14+IHL+8]											Datos
...												
T[14+IHL+39]												

### Descripción de cada campo:

1. **Tipo:** Campo que ocupa 8 bits y nos informa el tipo de mensaje que esta próximo a ser tratado, entre los cuales tenemos:

TIPO ICMP	Descripción
0	Respuesta de ECO
3	Destino inalcanzable
4	Flujo de origen
5	Redirección
8	Solicitud de ECO
9	Anuncio de enrutador
10	Solicitud de enrutador
11	Tiempo de espera
12	Problema de parámetros

Entre estos mensajes hay algunos de suma importancia, como los mensajes de petición de ECO (tipo 8) y los de respuesta de Eco (tipo 0). Las peticiones y respuestas de eco se usan en redes para comprobar si existe una comunicación entre dos host a nivel de capa de red, por lo que nos pueden servir para identificar fallos en este nivel, ya que verifican si las capas física (cableado), de enlace de datos (tarjeta de red) y red (configuración IP) se encuentran en buen estado y configuración.

2. **Código:** El campo de código de 8 bits igualmente en ocasiones nos ofrece una descripción del error con concreto que se ha producido.

TIPO ICMP	Descripción
0	No se puede llegar a la red
1	No se puede llegar al host o aplicación de destino
2	El destino no dispone del protocolo solicitado
3	No se puede llegar al puerto destino o la aplicación destino no está libre
4	Se necesita aplicar fragmentación, pero el flag correspondiente indica lo contrario
5	La ruta de origen no es correcta
6	No se conoce la red destino
7	No se conoce el host destino
8	El host origen está aislado

### 3. Mensajes de error

En el caso de obtener un mensaje ICMP de destino inalcanzable, con campo "tipo" de valor 3, el error concreto que se ha producido vendrá dado por el valor del campo "código", entre los cuales tenemos

4. **Checksum:** Tal como en otras cabeceras nos encontramos con el campo de suma de comprobación, la cual consta de 16 bits, esta es la suma de verificación de errores de transmisión
5. **Identificador:** Campo de 2 bytes que se encuentra presente en la solicitud o respuesta de eco, el cual se utiliza para reunir la solicitud de eco con su correspondiente respuesta de eco
6. **Numero de Secuencia:** Campo de 2 bytes que almacena un numero adicional que se utiliza para reunir la solicitud de ECO con su correspondiente respuesta de ECO

- Código y las salidas correspondientes para el análisis de las tramas Solicitud y respuesta de ECO.

A continuación, se anexa únicamente la función que imprime los datos para una trama de tipo ICMP de solicitud y respuesta ECHO

```
void ICMP(unsigned char T[]){

    printf("====Protocolo
    ICMP====\n");

    unsigned char IHL = (T[14]&15)*4;

    if ((T[IHL+14] == 8) && (T[IHL+15] == 0)){

        printf("Solicitud ECHO\n");
    }

    if ((T[IHL+14] == 0) && (T[IHL+15] == 0)){

        printf("Respuesta ECHO\n");
    }

    printf("Checksum: %.2x\n", (T[IHL+16]<<8 | T[IHL+17]));

    printf("Identificador: %.2x\n", (T[IHL+18]<<8 | T[IHL+19]));

    printf("Numero de Secuencia: %d\n", (T[IHL+20]<<8 | T[IHL+21]));

    printf("Datos Opcionales: ");
    for (unsigned char i = IHL+22; i < IHL+54; i++)
        printf("%c", T[i]);
}
```

## Captura Solicitud ECHO

```
Administrador: C:\Windows\system32\cmd.exe
C:\Users\JomianTC\Desktop>gcc tramasFinal.c -o p.exe

C:\Users\JomianTC\Desktop>p.exe
Alumnos: Mora Ayala Jose Antonio
          Torres Carrillo Josehf Miguel Angel

=====Cabecera ETHERNET =====
MAC Destino: 00:01:f4:43:c9:19:00,0x50,0xba,0xb2,0xf3,0x7b,0x00,0x45,0x00,
0x00,0x3c,0x09,0xed,0x00,0x00,0x00,0x01,0x7b,0xfe,0x94,0xcc,0x19,0x1b,0x94,0xcc,
0x73,0x07,0x05,0x00,0x50,0x50,0x02,0x00,0xf3,0x01,0x61,0x62,0x63,0x64,0x65,0x66,
0x67,0x68,0x69,0x6a,0x6b,0x6c,0x6d,0x6e,0x6f,0x70,0x71,0x72,0x73,0x74,0x75,0x76,
0x77,0x61,0x62,0x63,0x64,0x65,0x66,0x67,0x68,0x69};
=====Cabecera IP=====
Servicio: OTRO
Checksum 7bfe
Direccion IP Origen: 148 204 25 27
Direccion IP Destino: 148 204 115 2
No tiene Opciones
=====Protocolo ICMP=====
Solicitud ECHO
Checksum: 585a Antonio\n\t Torres Carrillo Josehf Miguel Angel\n\n");
Identificador: 200
Numero de Secuencia: 62209
Datos Opcionales: abcdefghijklmnopqrstuvwxyzabcdefghijklmnopghi
=====Cabecera ETHERNET =====\n");
ma(T);
C:\Users\JomianTC\Desktop>_

(unsigned char T[]){
```

## Captura Respuesta ECHO

```
Administrador: C:\Windows\system32\cmd.exe
C:\Users\JomianTC\Desktop>gcc tramasFinal.c -o p.exe

C:\Users\JomianTC\Desktop>p.exe
Alumnos: Mora Ayala Jose Antonio
Torres Carrillo Josehf Miguel Angel

=====Cabecera ETHERNET =====
MAC Destino: 00:50:ba:b2:f3:7b:
MAC Origen: 00:01:f4:43:c9:19:
=====Cabecera IP=====
Servicio: OTRO
Checksum a9ae
Direccion IP Origen: 148.204.115.2
Direccion IP Destino: 148.204.25.27
No tiene Opciones
=====Protocolo ICMP=====
Respuesta ECHO
Checksum: 605a
Identificador: 200
Numero de Secuencia: 62209
Datos Opcionales: abcdefghijklmnopqrstuvwxyzwvabcdefghijklmnop
=====Cabecera ETHERNET =====
=====Cabecera ETHERNET =====
C:\Users\JomianTC\Desktop>
```

## Capa de Transporte

- Introducción sobre la capa de transporte

El protocolo de la capa de transporte proporciona un servicio de entrega confiable de transferencia de datos de extremos a extremo

La capa de transporte es responsable de establecer una sesión de comunicación temporal entre dos aplicaciones y de transmitir datos

entre ellas. TCP/IP utiliza dos protocolos para lograrlo:

- Protocolo de control de transmisión (TCP)
- Protocolo de datagramas de usuario (UDP)
- Principales responsabilidades de los protocolos de la capa de transporte
- Rastreo de comunicación individual entre aplicaciones en los hosts de origen y destino
- División de los datos en segmentos para su administración y reunificación de los datos segmentados en streams de datos de aplicación en el destino

CAPA DE TRANSPORTE	
Aplicación	Aplicación
Presentación	
Sesión	
Transporte	Transporte
Red	Internet
Enlace de Datos	Acceso a la Red
Física	

- Un párrafo de explicación de los sistemas orientados a conexión junto con un diagrama con las 3 etapas en donde se muestre claramente la terminal de Alicia y la terminal de Betito.

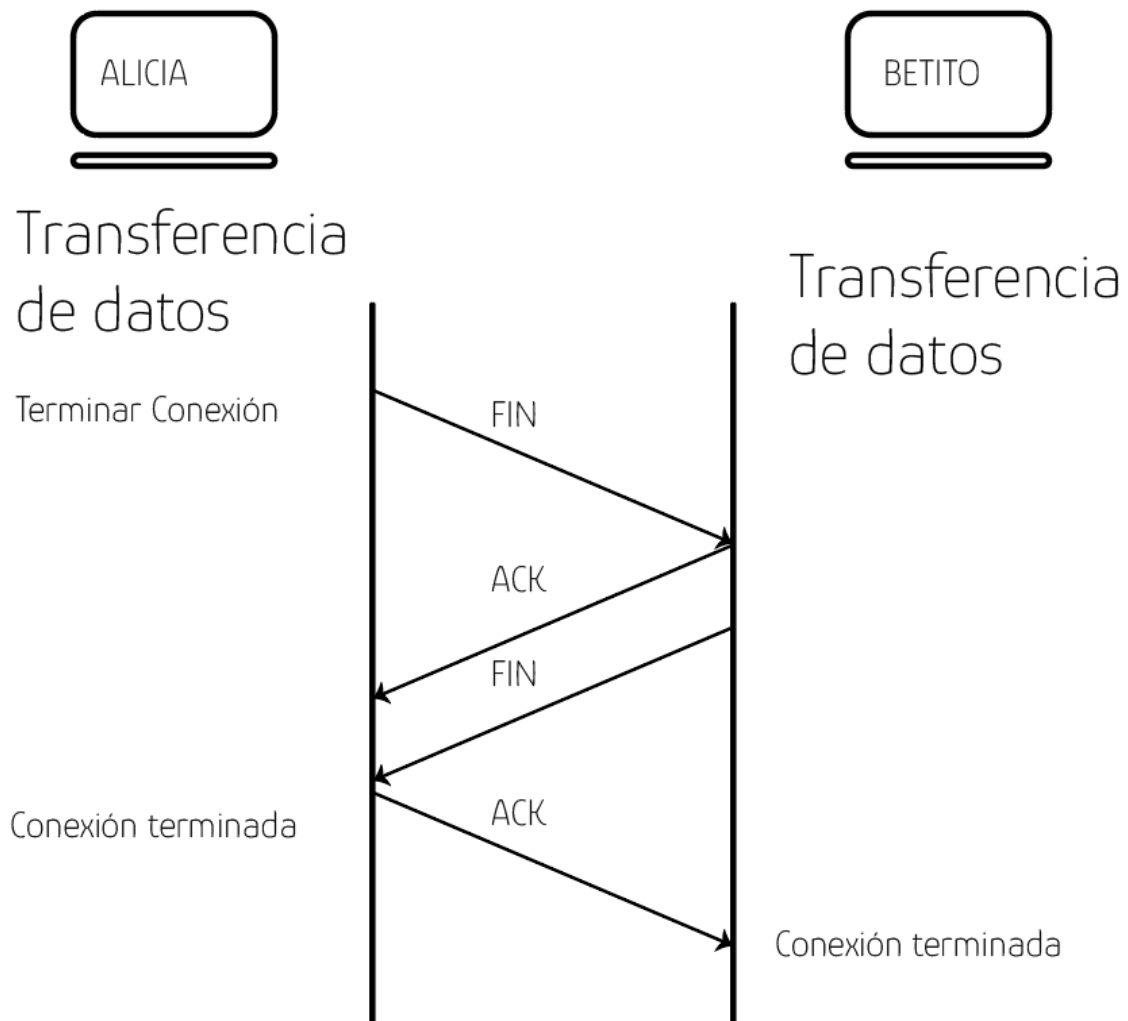
### Orientado a conexión

- Antes de poder transferir los datos, dos procesos (local y remoto) deben negociar una conexión TCP mediante un proceso de establecimiento de conexión.
- Las conexiones TCP se cierran formalmente mediante el proceso de finalización de conexión TCP.
- TCP segmenta los datos obtenidos a partir del proceso de la capa de
- Aplicación para adaptarlos a un datagrama IP enviado por el enlace de la capa de Interfaz de Red.
- Las terminales TCP intercambian el segmento de tamaño máximo que puede recibir cada uno y ajustan el tamaño máximo del segmento TCP



Con respecto a esta cabecera debemos tener en cuenta las ventajas que nos proporciona como tal

- Proporciona una entrega confiable que asegura que todos los datos lleguen al destino.
- Utiliza el acuse de recibo (ACK) y otros procesos para asegurar la entrega.
- Segmentación de datos de la capa de aplicación
- Mayores demandas sobre la red: mayor sobrecarga.



## Protocolo TCP

Incluir un párrafo de definición de siglas y principal funcionamiento del protocolo, definir en que capa trabaja y su función(es) principal(es)

Las siglas TCP significan **Transmission Control Protocol** o en español **Protocolo de control de transmisión** es un protocolo cuyo funcionamiento principal es proporcionar un servicio de entrega confiable de transferencia de datos de extremo a extremo el cual trabaja en la capa de transporte

- Imagen de la Cabecera completa

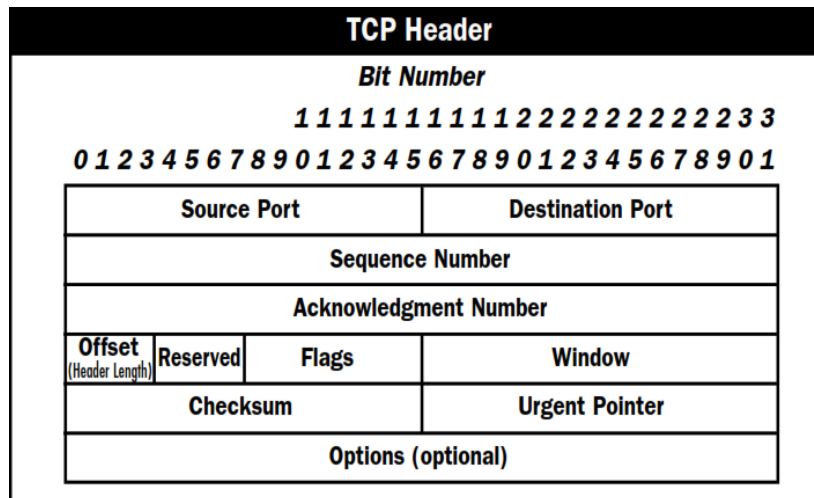


Imagen Obtenida del Documento proporcionado por la profesora, Creado por The Sans Technology Institute (STI)

- Descripción breve de cada campo.

Los campos que conforman la cabecera TCP son los siguientes:

1. Puerto Origen – 2 Bytes  
Indica el protocolo de la capa de aplicación de origen que se envía
2. Puerto Destino – 2 Bytes  
Indica el protocolo de la capa de aplicación de destino
3. Numero de Secuencia – 4 Bytes  
Indica el numero de secuencia, basado en secuencias de bytes de salida del primer byte de segmento
4. Numero de Reconocimiento – 4 Bytes  
Indica el numero del siguiente byte que el receptor espera recibir
5. Offset (Tamaño de la cabecera) – 4 bits  
Indica el numero de palabras de 32 bits de la cabecera TCP, cuyo valor mínimo es de 5 y máximo de 15

6. Reservado – 6 Bits

Son bits reservados por la cabecera inicializados en 0

7. Bandera – 6 Bits

Indican dependiendo de la bandera una acción a realizar, algunos de las banderas que se pueden encontrar son URG, ACK, PSH, RST etc.

8. Ventana – 2 Bytes

Indica el numero de bytes que hay disponible en el buffer del receptor

9. Checksum – 2 Bytes

Suma de comprobación para detectar errores en la transmisión de datos

10. Puntero Urgente – 2 Bytes

Indica la ubicación de los datos urgentes en el segmento

11. Opciones - Variable

Relleno de la cabecera TCP de tamaño variable con un tamaño máximo de 40 bytes

• Mapa de memoria (como imagen)

Cabecera TCP									
C	T[14+IHL]								
A	T[14+IHL+1]								
B	T[14+IHL+2]								
E	T[14+IHL+3]								
C	T[14+IHL+4]								
E	T[14+IHL+5]								
R	T[14+IHL+6]								
A	T[14+IHL+7]								
	T[14+IHL+8]								
T	T[14+IHL+9]								
C	T[14+IHL+10]								
P	T[14+IHL+11]								
	T[14+IHL+12]								
	T[14+IHL+13]	O	U	A	P	R	S	F	
	T[14+IHL+14]								
	T[14+IHL+15]								
	T[14+IHL+16]								
	T[14+IHL+17]								
	T[14+IHL+18]								
	T[14+IHL+19]								
	T[14+IHL+20]								
	T[14+IHL+21]								
	T[14+IHL+22]								
	.								
	.								
	.								
	T[14+IHL+59]								
		Puerto Origen				(T[14+IHL]<<8 T[14+IHL+1])			
		Puerto Destino				(T[14+IHL+2]<<8 T[14+IHL+3])			
		Numero de secuencia				(T[14+IHL+4]<<32 T[14+IHL+5]<<16 T[14+IHL+6]<<8 T[14+IHL+7])			
		Numero de reconocimiento				(T[14+IHL+8]<<32 T[14+IHL+9]<<16 T[14+IHL+10]<<8 T[14+IHL+11])			
		HL = ((T[14+IHL+12]>>4)&15)*4				4 bits en 0		HL valor minimo de 5	
		Bandera							
		Ventana				(T[14+IHL+14]<<8 T[14+IHL+15])			
		Checksum				(T[14+IHL+16]<<8 T[14+IHL+17])			
		Puntero Urgente				(T[14+IHL+18]<<8 T[14+IHL+19])			
		Si hay opciones							
		Opciones							
		Inicia en T[14+IHL+20]							
		Termina en T[14+IHL+HL-1]							

U=consulta de puntero urgente  
A=Consulta del acuse ACK  
P=Push  
R=Reset  
S=Sincronizacion  
F=Finalizar Conexion

U=consulta de puntero urgente  
A=Consulta del acuse ACK  
P=Push  
R=Reset  
S=Sincronizacion  
F=Finalizar Conexion

- Construcción sobre la pseudocabecera para el checksum TCP y ejemplo.

Para comprobar el checksum de un protocolo TCP deberemos construir una cabecera especial, dicha cabecera esta construida de la siguiente manera

1. Se deberán tener los 4 bytes de la dirección IP origen obtenida de la cabecera IP
2. Obtener los 4 bytes de la dirección IP Destino obtenida de la cabecera IP
3. Se deberá tener un byte entero igual a 0
4. Se deberá tener 1 byte con el protocolo que se esta usando, obtenido de la cabecera TCP, como sabemos que estamos trabajando TCP el valor de este byte será igual a 0x06
5. 2 bytes que indican el tamaño en bytes de la cabecera TCP
6. Se deberá concatenar la cabecera TCP entera

Ejemplo

Tenemos la siguiente trama TCP

```

00 14 d1 c2 38 b3 00 18 e7 33 3d c3 08 00 45 00
00 30 94 71 40 00 80 06 f9 8c c0 a8 02 3c 4a 7d
5f 68 10 52 00 50 03 c7 5a a1 00 00 00 00 70 02
40 00 67 4b 00 00 02 04 05 b4 01 01 04 02

```

Donde la parte de amarillo corresponde a la cabecera Ethernet, la parte de rosa corresponde a la cabecera IP y finalmente la cabecera de azul corresponde a la cabecera TCP

Para crear nuestra pseudocabecera para el checksum seguiremos las instrucciones anteriormente descritas

1. Primero deberemos tener la dirección IP origen de la cabecera IP

c0 a8 02 3c

2. Después deberemos tener la dirección IP Destino

4a 7d 5f 68

3. Deberemos tener un byte en 0

00

4. Se tendrá un byte con el protocolo utilizado en este caso será igual a

06

5. Se tendrán 2 bytes que indican el tamaño de la cabecera TCP

7 \* 4 = 28 bytes = 00 1c hexadecimal

6. Por último, se concatena la cabecera TCP al completo

```

10 52 00 50 03 c7 5a a1 00 00 00 00 70 02
40 00 67 4b 00 00 02 04 05 b4 01 01 04 02

```

Teniendo como resultado la siguiente pseudocabecera

```
c0 a8 02 3c 4a 7d 5f 68 00 06 00 1c 10 52 00 50
03 c7 5a a1 00 00 00 00 70 02 40 00 67 4b 00 00
02 04 05 b4 01 01 04 02
```

- Colocar el código correspondiente para la construcción de la pseudocabecera TCP. Y resultados de su ejecución.

Para la construcción de la pseudocabecera se unifico junto con una función de checksum especial para trabajar de manera específica con esta trama

A continuación, solo se adjunta la función que crea la pseudocabecera y calcula el checksum a la vez

```
void checksum(unsigned char T[]){

    //INICIO CONSTRUCCION PSEUDOCABECERA
    unsigned char IHL = (T[14]&15)*4;
    unsigned char offset = ((T[14+IHL+12]>>4)&15)*4;

    unsigned char auxiliar[12+offset];

    auxiliar[0] = T[26];
    auxiliar[1] = T[27];
    auxiliar[2] = T[28];
    auxiliar[3] = T[29];
    auxiliar[4] = T[30];
    auxiliar[5] = T[31];
    auxiliar[6] = T[32];
    auxiliar[7] = T[33];
    auxiliar[8] = 0x00;
    auxiliar[9] = T[23];
    auxiliar[10] = 0x00;
    auxiliar[11] = offset;

    for (unsigned char i = 0; i < offset ; i++)
        auxiliar[i+12] = T[14+IHL+i];
    //FIN CONSTRUCCION PSEUDOCABECERA

    unsigned int suma = 0x00, acarreo = 0x00;
    unsigned char i, tam;
    i = 0x00;
    tam = 12+offset;
    // Agarrando valores pares
    while (tam > 1)
    {
        acarreo = (((auxiliar[i] << 8) & 0xFF00) | ((auxiliar[i + 1]) &
0xFF));
        suma += acarreo;
        if ((suma & 0xFFFF0000) > 0)
        {
            suma = suma & 0xFFFF;
            suma += 1;
        }
    }
}
```

```

    }
    i += 2;
    tam -= 2;
}
suma = ~suma;
suma = suma & 0xFFFF;

if (suma == 0)
{
    printf("El checksum es correcto :)\n");
}
else
{
    printf("El checksum no es correcto :(\n");
    suma = 0x00, acarreo = 0x00;
    i = 0x00;
    tam = 12+offset;
    auxiliar[28] = 0;
    auxiliar[29] = 0;
    while (tam > 1)
    {
        acarreo = (((auxiliar[i] << 8) & 0xFF00) | ((auxiliar[i + 1])
& 0xFF));
        suma += acarreo;
        if ((suma & 0xFFFF0000) > 0)
        {
            suma = suma & 0xFFFF;
            suma += 1;
        }
        i += 2;
        tam -= 2;
    }

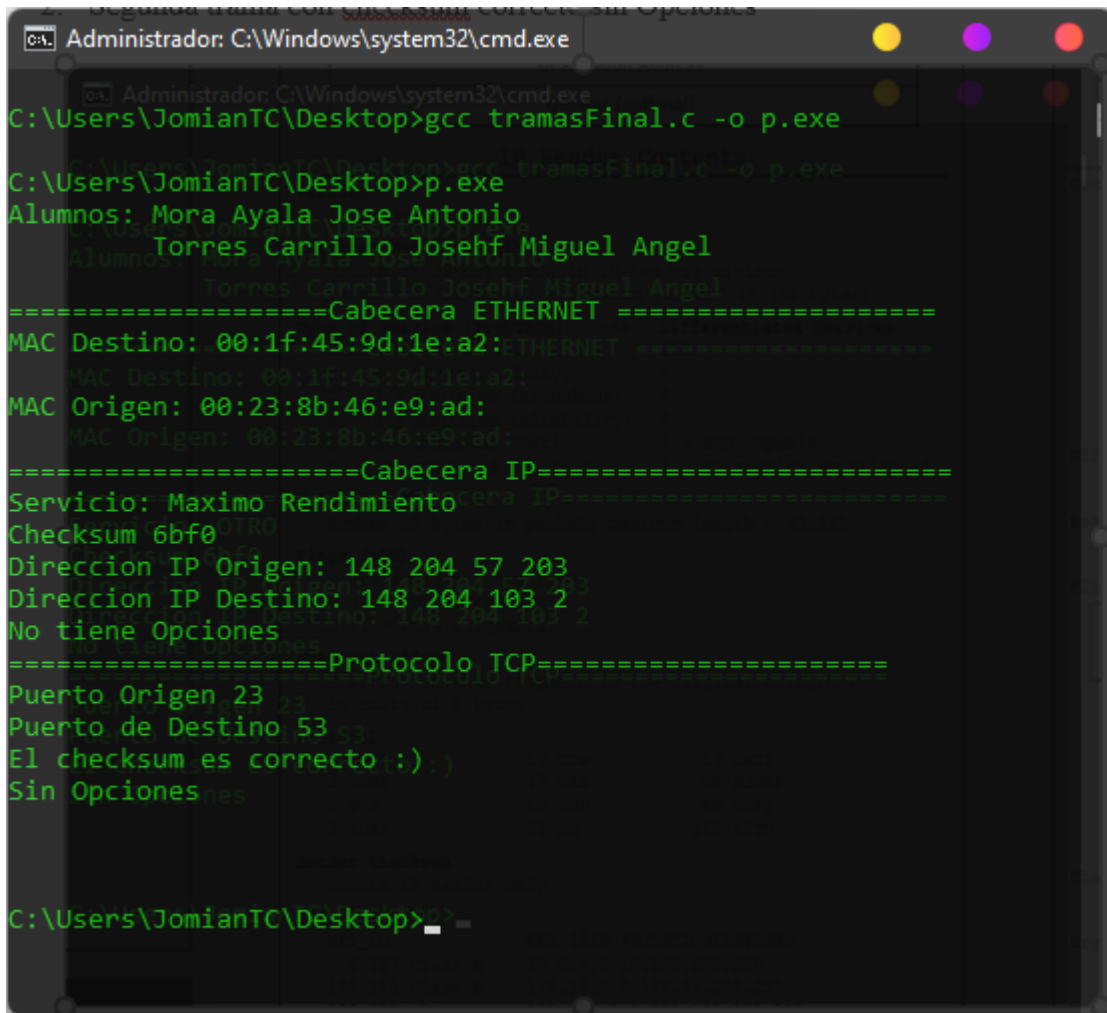
    suma = ~suma;
    suma = suma & 0xFFFF;
    printf("El checksum correcto es: 0x%.4x\n", suma);
}
}

```

## 1. Primera trama con opciones y checksum incorrecto

```
C:\Users\JomianTC\Desktop>gcc tramasFinal.c -o p.exe
C:\Users\JomianTC\Desktop>p.exe
Alumnos: Mora Ayala Jose Antonio
         Torres Carrillo Josehf Miguel Angel
=====Cabecera ETHERNET=====
MAC Destino: 00:14:d1:c2:38:be: 0x04,0x05,0xb4,0x01,0x01,0x04,0x02);
MAC Origen: 00:18:e7:33:3d:c3:
=====Cabecera IP=====
Servicio: Costo minimo
Checksum f98c
Direccion IP Origen: 192 168 2 60
Direccion IP Destino: 74 125 95 104
Tamano de Opciones 4 bytes
0x10 0x52 0x00 0x50
=====Protocolo TCP=====
Puerto Origen 80
Puerto de Destino 1080
El checksum no es correcto :(
El checksum correcto es: 0x8ea3
Opciones 1010402
0x02,
0x40, 0x00,
0x1b, 0x91,
0x00, 0x00,
C:\Users\JomianTC\Desktop>
```

## 2. Segunda trama con checksum correcto sin Opciones



```
Administrador: C:\Windows\system32\cmd.exe
C:\Users\JomianTC\Desktop>gcc tramasFinal.c -o p.exe
C:\Users\JomianTC\Desktop>p.exe
Alumnos: Mora Ayala Jose Antonio
          Torres Carrillo Josehf Miguel Angel
=====Cabecera ETHERNET =====
MAC Destino: 00:1f:45:9d:1e:a2:
MAC Origen: 00:23:8b:46:e9:ad:
=====Cabecera IP=====
Servicio: Maximo Rendimiento
Checksum 6bf0
Direccion IP Origen: 148 204 57 203
Direccion IP Destino: 148 204 103 2
No tiene Opciones
=====Protocolo TCP=====
Puerto Origen 23
Puerto de Destino 53
El checksum es correcto :)
Sin Opciones
C:\Users\JomianTC\Desktop>
```

Investiga ejemplos de aplicaciones que utilicen el protocolo TCP

- Aplicaciones multimedia y vídeo en vivo: pueden tolerar cierta pérdida de datos, pero requieren retrasos cortos o que no haya retrasos, como VoIP y la transmisión de vídeo en vivo.
- Aplicaciones con solicitudes y respuestas simples: aplicaciones con transacciones simples en las que un host envía una solicitud y existe la posibilidad de que reciba una respuesta o no.
- Aplicaciones que manejan la confiabilidad por su cuenta: comunicaciones unidireccionales que no requieran control de flujo, detección de errores, reconocimientos ni recuperación de errores o que puedan ser manejados por la aplicación, como SNMP y TFTP.

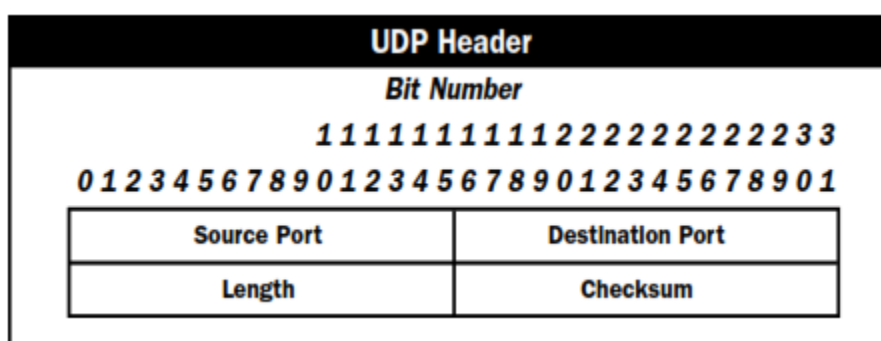


## Protocolo UDP

Incluir un párrafo de definición de siglas y principal funcionamiento del protocolo, definir en que capa trabaja y su función(es) principal(es)

Las siglas UDP significan **User Datagram Protocol** o en español **Protocolo de Datagramas de usuario** este protocolo al igual que TCP trabaja en la capa de transporte es para comunicaciones no orientadas a conexión este protocolo envía los datos al destino sin tener en cuenta si este esta listo o no, es muy rápido en el envío pero al contrario de TCP no tiene la misma fiabilidad

- Imagen de la Cabecera completa



- Descripción breve de cada campo

Los campos que conforman la cabecera UDP son los siguientes

1. Puerto origen – 2 bytes  
Indica el protocolo de la capa de aplicación de origen que se envía
2. Puerto Destino – 2 bytes  
Indica el protocolo de la capa de aplicación de destino
3. Longitud  
Numero en bytes en el datagrama incluyendo la cabecera con un valor mínimo de 8
4. Checksum  
Suma de comprobación para detectar errores en la transmisión de datos

- Mapa de memoria (como imagen)

Cabecera UDP	T[IHL+14]							Puerto Origen	(T[IHL+14]<<8) T[IHL+15])
	T[IHL+15]								
	T[IHL+16]							Puerto Destino	(T[IHL+16]<<8) T[IHL+17])
	T[IHL+17]								
	T[IHL+18]							Longitud	(T[IHL+18]<<8) T[IHL+19])
	T[IHL+19]								
T[IHL+20]							Checksum	(T[IHL+20]<<8) T[IHL+21])	
T[IHL+21]									

- Para la implementación de este analizador se imprimen solamente los campos
- Poner el código correspondiente y la salida para dos tramas ejemplo.

Código correspondiente a la impresión de datos del protocolo UDP

```
void UDP(unsigned char T[]){

    printf("=====Protocolo UDP=====\\n");

    unsigned char IHL = (T[14]&15)*4;

    printf("Puerto Origen %d\\n", (T[IHL+14]<<8 | T[IHL+15]));

    printf("Puerto de Destino %d\\n", (T[IHL+16]<<8 | T[IHL+17]));

    printf("Longitud %d\\n", (T[IHL+18]<<8 | T[IHL+19]));

    printf("Checksum %.2x\\n", (T[IHL+20]<<8 | T[IHL+21]));

}
```

## Captura de pantalla primera Trama

```

Administrador: C:\Windows\system32\cmd.exe
CP con opciones (4bytes) con checksum incorrecto
00 14 d1 c2 38 be 00 18 e7 33 3d c3 08 00 46 02
C:\Users\JomianTC\Desktop>gcc tramasFinal.c -o p.exe
00 30 94 71 40 00 80 06 f9 8c c0 a8 02 3c 4a 7d
C:\Users\JomianTC\Desktop>p.exe
00 50 04 38 00 00 00 00 70 02
Alumnos: Mora Ayala Jose Antonio 00 02 04 05 b4 01 01 04 02
Torres Carrillo Josehf Miguel Angel
TCP sin opciones con checksum correcto
-----Cabecera ETHERNET -----
MAC Destino: 00:1f:45:9d:1e:a2: 00 23 8b 46 e9 ad 08 00 45 08
80 42 04 55 34 11 80 06 6b f0 94 cc 39 cb 94 cc
MAC Origen: 00:23:8b:46:e9:ad: 00 2e 85 7c 00 00 00 00 50 02
40 00 1f 86 00 00
-----Cabecera IP-----
UDP ejemplo
Servicio: OTRO
Checksum 6bf0
Direccion IP Origen: 148.204.57.203 3 8b 46 e9 ad 08 00 46 00
Direccion IP Destino: 148.204.103.21 6b f0 94 cc 39 cb 94 cc
Tamano de Opciones 4 bytes
0xaa 0xbb 0xcc 0xdd 00 07 00 35 00 2e 85 7c e2 1a
01 00 00 01 00 00 00 00 00 00 03 77 77 77 03 69
-----Protocolo UDP-----
Puerto Origen 7 73 63 05 65 73 63 6f 6d 03 69 70 6e 02 6d 78 00
Puerto de Destino 53 00 1e 00 01
Longitud 46
Checksum 857c
ff ff ff ff ff ff 00 23 8b 46 e9 ad 08 00 45 00
08 00 06 04 00 01 00 11 8b 46 e9 ad 94 cc 39 cb
C:\Users\JomianTC\Desktop> 00 45 00 07 00 08 39 fe

```

## Captura de pantalla segunda Trama

```
C:\Windows\system32\cmd.exe
C:\Users\JomianTC\Desktop>gcc tramasFinal.c -o p.exe

C:\Users\JomianTC\Desktop>p.exe
Alumnos: Mora Ayala Jose Antonio
Torres Carrillo Josehf Miguel Angel

=====Cabecera ETHERNET =====
MAC Destino: ff:ff:ff:ff:ff:ff:
MAC Origen: 00:23:8b:46:e9:ad:

=====Cabecera IP=====
Servicio: OTRO
Checksum 8b46
Direccion IP Origen: 233 173 148 204
Direccion IP Destino: 57 203 0 0
No tiene Opciones

=====Protocolo UDP=====
Puerto Origen 69
Puerto de Destino 7
Longitud 8
Checksum 39fe

char T[] = {0x00, 0x50, 0xba, 0xb2, 0xf3, 0x7b, 0x00, 0x01, 0xf4, 0xc9, 0xc9, 0x00, 0x01, 0x0e, 0x94, 0xcc, 0x73, 0x02, 0x94, 0xcc, 0x00, 0x50, 0x02, 0x00, 0xf3, 0x01, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x6d, 0x6e, 0x6f, 0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69}
```

- Investiga ejemplos de aplicaciones que utilicen el protocolo UDP

Se encarga de todas las tareas asociadas con la división del flujo de datos en segmentos, lo que proporciona confiabilidad, control del flujo de datos y reordenamiento de segmentos, algunos ejemplos de las aplicaciones que utilizan TCP son: "HTTP", "FTP", "SMTP" y "Telnet"

## Conclusiones:

Durante todo el curso pudimos obtener una amplia variedad de conocimientos los cuales estoy seguro que me servirán de mucho en un futuro, tanto en otras unidades de aprendizaje como en mi vida laboral, pues siempre es útil tener este tipo de conocimientos, los cuales son variados y de suma importancia. Realmente durante todo el curso pude ser testigo de todo el gran alcance que los temas de redes pueden abarcar y cual interesantes pueden ser.

Me parece que es de sumo interés todo lo que, y tiene que ver con las tramas, la forma en que estas se clasifican y como se puede y se debe identificar cada uno de los campos, así como la importancia de saber considerar las operaciones a implementar, así como considerar los espacios de memoria y todo el almacenamiento que debemos considerar al momento de tener que realizar la implementación de las mismas. Debemos saber identificar y realizar el correcto manejo de las operaciones no solamente aritméticas sino también las operaciones a nivel binario, las cuales presentan un costo mucho menor, así como una complejidad temporal de menor costo, lo cual me parece increíble, saber administrar y realizar el manejo de todas estas operaciones, tanto como conocimiento general de programadores como al momento de realizar la implementación.

En conclusión, pude obtener una gran variedad de conocimientos gracias a la profesora y a esta Unidad de Aprendizaje, considero que aún hay muchos tópicos que deben ser abordados para poder terminar de conocer estos temas a un nivel de profundidad mayor.

Mora Ayala Jose Antonio

El poder de los bits me parece que es absolutamente increíble y el cual de manera general es de poco conocimiento y poca implementación al momento de tener que realizar la ejecución de un programa, o bien de transmitir los conocimientos al momento de realizar o aprender a programar, pues nos enseñan a tener que realizar siempre la implementación directa de un tipo de dato entero, cuando realmente no es para nada necesario ocupar tanta cantidad de memoria, la cual al final del camino solo esta siendo desaprovechada y siendo solicitada al compilador de forma innecesaria, así mismo retomando las operaciones binarias este es un tema que no es difícil de comprender sin embargo es poco enseñado y este ofrece una gran cantidad de ventajas que a simple vista no podemos observar por el tamaño de programas que hemos implementado, pero al momento de que estos programas se vean exponenciados a un tamaño mayor realmente podremos notar una diferencia y mejora si sabemos implementar los tipos de datos adecuados así como operaciones binarias cuando es necesario y de igual forma las aritméticas

Esta unidad de aprendizaje me nutrió de muchos conocimientos y estoy seguro que habrá muchos mas por venir, así mismo agradezco a la profesora por impartirnos de la forma en que nos impartió esta Unidad de Aprendizaje pues pudimos obtener mucho de ella.

Torres Carrillo Josehf Miguel Angel

Por equipo responder a las siguientes preguntas para generar las conclusiones de este analizador desarrollado a lo largo del semestre (solo debes tener las respuestas a las preguntas, no volver a escribir las preguntas)

1. La principal diferencia que podríamos encontrar es que aquellos que son orientados a byte se basan principalmente en la utilización de bytes completos para representar códigos de control establecidos (código ASCII), mientras que los protocolos orientados a bits confían en bits individuales para los códigos de control
2. La principal ventaja que podemos observar es que el protocolo LLC es capaz de proporcionar una transferencia de datos sin la necesidad de una conexión presente y al mismo tiempo orientada a conexión, este protocolo nos auxilia a la resolución de la mayoría de los problemas dado que no hay estados o sesiones a mantener a parte que los errores en una comunicación por LLC son raros.
3. Realmente antes de este curso, ambos miembros del equipo desconocíamos totalmente acerca de este proceso de encapsulamiento y el proceso de desencapsulamiento, posterior a la descripción que hemos proporcionado consideramos que hemos llegado a poder comprender este tópico de una forma mas clara, aunque no en su totalidad, pues con las lecturas que realizamos y la profundidad abarcada podemos concluir que este proceso realmente involucra mas temas que de igual forma tienen mucha importancia en estos procesos. De tal forma que podemos llegar a la conclusión de que este es un proceso importante para poder transmitir un mensaje de forma precisa y de la manera más segura posible.
- 4.

mensaje	Cantidad de caracteres	Bytes agregados en las cabeceras Ethernet, IP y TCP	Total de bytes para ese mensaje
Ola k ace	9	9	$14+20+20+9 = 63$
?	1	1	$14+20+20+1=55$
☺	1	1	$14+20+20+1=55$

Tal como ya hemos explicado cada capa realiza la inserción de una cabecera propia, la cual se la anexa a un todo conforme esta se va moviendo a lo largo de las distintas capas que forman parte del proceso de encapsulamiento.

5. Lo que sucede es que al momento de realizar la llamada a este tipo de operador hacemos que nuestro compilador trabaje de una forma extra, dado que estamos realizando una concatenación de varias sumas a nivel binario, provocando el inicio de un ciclo, lo cual provoca una complejidad temporal mayor (no la percibimos a simple vista, pero realizando una medición de tiempos lo podemos notar)
6. La forma en que se hace la interpretación de esta operatividad es mediante la modularidad de los valores de forma binaria, en los cuales simplemente realizara

desplazamiento de los valores binarios con los cuales está tratando actualmente en la cantidad indicada por la operatividad en cuestión, de tal forma que en los espacios desplazados se le agregaran 0 en su lugar

Espero que recuerdes que las operaciones

- $T[12] * 256 + T[13]$
- $T[12] \ll 8 \mid T[13]$

Devuelven exactamente el mismo resultado, ¿cuál es mejor y por qué?

Lo mejor sería la segunda opción dado que estamos haciendo que nuestro compilador trabaje directamente con las operatividades de forma binaria, no habiendo ninguna necesidad de hacer que este interprete alguno de los “caracteres” como suma, multiplicación etc. Es como trabajar desde la forma primitiva del compilador, ahorrándole un paso o bien gasto de memoria.

## 7. Tecnología

- Unicast (1 a 1)
- Multicast (1 a varios)
- Broadcast (1 a Todos)

### Extensión

- Según su área geográfica

### Topología

- Forma en como estan conectadas