INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

PROYECTO

"ANALIZADOR DE PROTOCOLOS"

REDES

INTEGRANTES:

- CORTES LÓPEZ JAIME ALEJANDRO
- GONZÁLEZ MORA ERIKA GISELLE
- MARTINEZ MARTINEZ FERNANDO
- OLIVARES MENÉZ GLORIA OLIVA
- VAZQUEZ PEREZ DENZEL OMAR

GRUPO: 2CV16

ÍNDICE

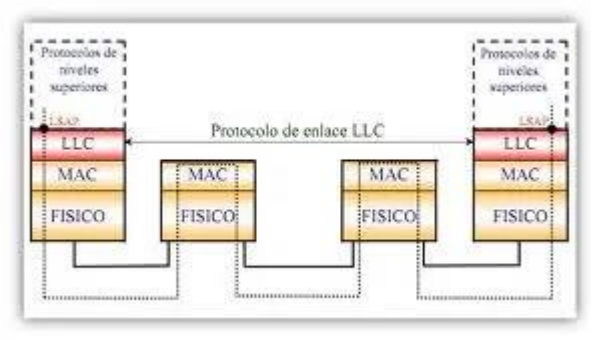
1. Introducción

Un analizador de protocolos es una herramienta que sirve para desarrollar y depurar protocolos y aplicaciones de red. Permite al ordenador capturar diversas tramas de red para analizarlas, ya sea en tiempo real o después de haberlas capturado.

Por analizar se entiende que el programa puede reconocer que la trama capturada pertenece a un protocolo concreto (TCP, ICMP...) y muestra al usuario la información decodificada. De esta forma, el usuario puede ver todo aquello que en un momento concreto está circulando por la red que se está analizando. También, gracias a estos analizadores, se puede ver la relación que hay entre diferentes protocolos, para así, comprender mejor su funcionamiento.

1.1. Protocolo LLC

Control de enlace lógico LLC ("Logical Link Control") define la forma en que los datos son transferidos sobre el medio físico, proporcionando servicio a las capas superiores. Es la más alta de las dos subcapas de enlace de datos definidas por el IEEE y la responsable del control de enlace lógico.



La subcapa LLC maneja el control de errores, control del flujo, entramado, control de diálogo y direccionamiento de la subcapa MAC. El protocolo LLC más generalizado es IEEE 802.2, que incluye variantes no orientado a conexión y orientadas a conexión.

1.2. Protocolo ARP

Para poder enviar paquetes de datos en redes TCP/IP, un servidor necesita, sobre todo, tres datos de dirección sobre el host al que se dirige: la **máscara de subred**, la **dirección IP** y la dirección MAC (también conocida como dirección de hardware o dirección física). Los dispositivos reciben la máscara de red y la dirección IP de manera automática y flexible cuando se establece la conexión con una red.

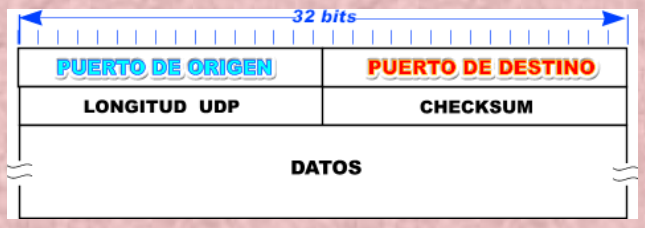
	Bits 0–7	Bits 8–15	Bits 16–23	Bits 24–31
0	Tipo de dirección de hardware (HTYPE)		Tipo de protocolo de red (PTYPE)	
32	Longitud de la dirección de hardware (HLEN)	Longitud de la dirección de protocolo (PLEN)	Operación	
64	Dirección MAC del remitente			
96				
112	Dirección IP del remitente			
144	Dirección MAC del destinatario			
176				
192	Dirección IP del destinatario			

Con este objetivo, los dispositivos de comunicación mediadores como routers o concentradores (hubs) recurren al protocolo DHCP. En las

redes locales se pueden introducir ambos datos manualmente. El **fabricante del dispositivo correspondiente** otorga la dirección de hardware, que queda vinculada a una dirección IP con ayuda del llamado Address Resolution Protocol (ARP).

1.3. Protocolo UDP

El protocolo de datagramas de usuario, abreviado como UDP, es un protocolo que permite la transmisión sin conexión de datagramas en redes basadas en IP. Para obtener los servicios deseados en los hosts de destino, se basa en los puertos que están listados como uno de los campos principales en la cabecera UDP. Como muchos otros protocolos de red, UDP pertenece a la familia de protocolos de Internet, por lo que debe clasificarse en el nivel de transporte y, en consecuencia, se encuentra en una capa intermedia entre la capa de red y la capa de aplicación.



1.4. Protocolo TCP

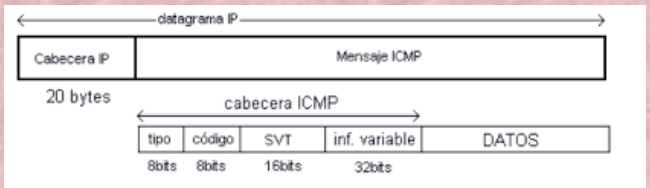
El protocolo TCP (Protocolo de Control de Transmisión) es uno de los protocolos fundamentales en Internet, nos permite que las aplicaciones puedan comunicarse con garantías independientemente de las capas inferiores del modelo TCP/IP. Esto significa que los routers (capa de red en el modelo TCP/IP) solamente tienen que enviar los segmentos (unidad de medida en TCP), sin preocuparse si van a llegar esos datos correctamente o no.

TCP da soporte a múltiples protocolos de la capa de aplicación, como, por ejemplo, HTTP (web), HTTPS (web segura), POP3 (correo entrante) y SMTP (correo saliente) así como sus versiones seguras utilizando TLS. También se utiliza TCP en protocolos tan importantes como FTP, FTPES y SFTP para transferir archivos desde un origen a un destino, e incluso el protocolo SSH para administrar equipos de forma local y remota de manera segura utiliza el protocolo TCP.

1.5. Protocolo ICMP

Para intercambiar datos de estado o mensajes de error, los nodos recurren al Internet Control Message Protocol (ICMP) en las redes TCP/IP. Por definición, ICMP es un protocolo

autónomo aun cuando los diferentes mensajes están incluidos en paquetes IP tradicionales. Para tal fin, el protocolo de Internet trata a la implementación opcional como un protocolo de capas superiores. Los diversos servicios de red que se suelen utilizar hoy en día, como traceroute o ping, se basan en el protocolo ICMP.

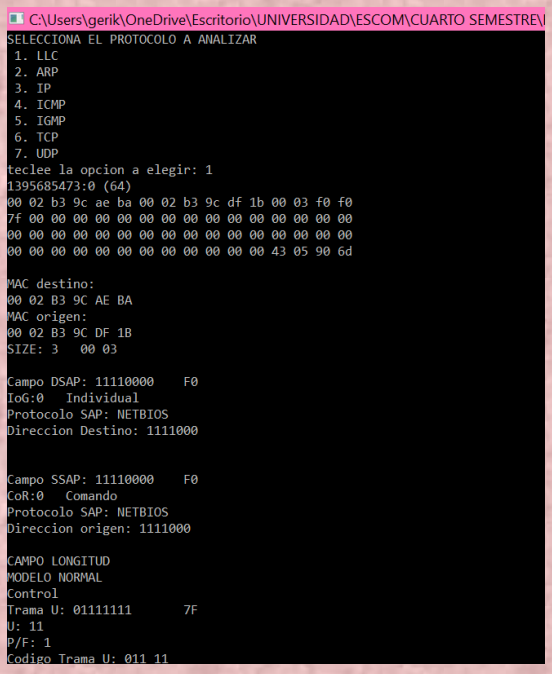


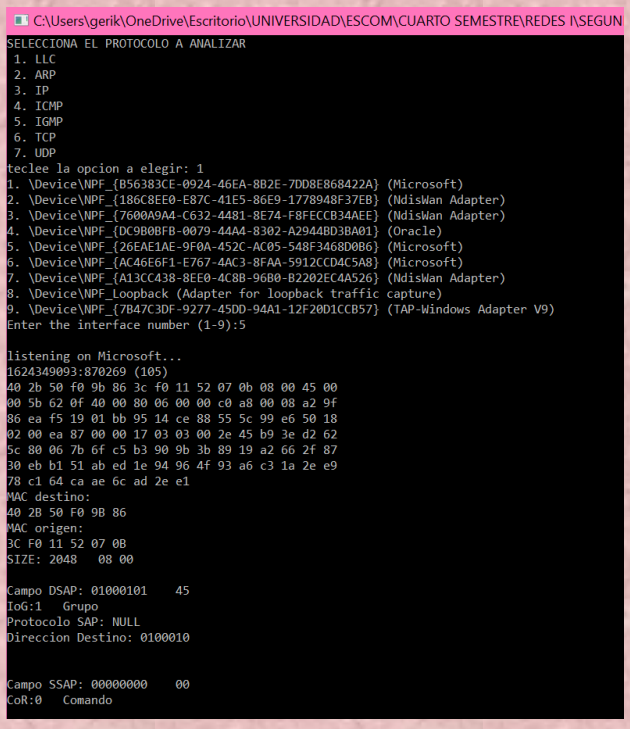
1.6. Protocolo IGMP

El Internet Group Management Protocol (IGMP) es utilizado por los host y el Routers en una red IP para crear las calidades de miembro de grupo de multidifusión. IGMP es un protocolo usado para notificar las membresías del grupo del host a un router vecino inmediato del Multicast por los host IP del dispositivo. IGMP se puede utilizar para los recursos de la red y de las aplicaciones de soporte como en línea fluir para los vídeos y los juegos. Los grupos de multidifusión son útiles especialmente como alternativa más eficiente de difundir. IGMP tiene versiones IGMP v1, v2 y v3. El v3 es la última versión del protocolo.

2. Pruebas y funcionamiento del analizador de protocolos

2.1. Tramas LLC





```
C:\Users\gerik\OneDrive\Escritorio\UNIVERSIDAD\ESCOM\CUARTO SEMESTRE\F

listening on Microsoft...
1624349093:870269 (105)
40 2b 50 f0 9b 86 3c f0 11 52 07 0b 08 00 45 00
00 5b 62 0f 40 00 80 06 00 00 c0 a8 00 08 a2 9f
86 ea f5 19 01 bb 95 14 ce 88 55 5c 99 e6 50 18
02 00 ea 87 00 00 17 03 03 00 2e 45 b9 3e d2 62
5c 80 06 7b 6f c5 b3 90 9b 3b 89 19 a2 66 2f 87
30 eb b1 51 ab ed 1e 94 96 4f 93 a6 c3 1a 2e e9
78 c1 64 ca ae 6c ad 2e e1
MAC destino:
40 2B 50 F0 9B 86
MAC origen:
3C F0 11 52 07 0B
SIZE: 2048   08 00

Campo DSAP: 01000101    45
IoG:1   Grupo
Protocolo SAP: NULL
Direccion Destino: 0100010


Campo SSAP: 00000000    00
CoR:0   Comando
Protocolo SAP: NULL SAP
Direccion origen: 0000000

CAMPO TIPO
```

2.2. Trama ARP

```
C:\Users\gerik\OneDrive\Escritorio\UNIVERSIDAD\ESCOM\CUARTO SEMESTRE\REDES I\SEGUNDO PARCIAL\Project_redes\Projec
8c c8 4b c1 26 ef 18 4a 6f 7b 48 68 08 06 00 01
08 00 06 04 00 01 18 4a 6f 7b 48 68 c0 a8 01 fe
00 00 00 00 00 00 c0 a8 01 47 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00
MAC destination: 8C:C8:4B:C1:26:EF:
MAC source: 18: 4A: 6F: 7B: 48: 68:

Tipo: 2054   08 06
Target Protocol Address

Hardware type:
        Ethernet(1)

Protocol type:
        IPv4

Hardware size: 6
Protocol size: 4
Opcode:
        ARP REQUEST (1)

Sender MAC address: 18 4A 6F 7B 48 68
Sender IP address: 192 168 1 254
Target MAC address: 00 00 00 00 00 00
Target IP address: 192.168.1.71

--------------------------------------------------------------------------------------------------------------


18 4a 6f 7b 48 68 8c c8 4b c1 26 ef 08 06 00 01
08 00 06 04 00 02 8c c8 4b c1 26 ef c0 a8 01 47
18 4a 6f 7b 48 68 c0 a8 01 fe
MAC destination: 18:4A:6F:7B:48:68:
MAC source: 8C: C8: 4B: C1: 26: EF:

Tipo: 2054   08 06
Target Protocol Address

Hardware type:
        Ethernet(1)

Protocol type:
        IPv4

Hardware size: 6
Protocol size: 4
Opcode:
        ARP REPLY (2)

Sender MAC address: 8C C8 4B C1 26 EF
Sender IP address: 192 168 1 71
Target MAC address: 18 4A 6F 7B 48 68
Target IP address: 192.168.1.254
```

2.3. Tramas UDP

```
b0 68 e6 03 77 c9 8c 61 a3 67 68 c8 08 00 45 00
00 39 00 00 40 00 76 11 7e 9e bd d8 07 91 c0 a8
00 04 01 bb ff 1f 00 25 7e 60 56 4e e3 3e 52 bf
78 37 ab bf 6f 00 d5 7b a4 ba c5 94 87 2f 5c 0f
9c dc 59 25 e8 01 da
MAC destination: B0:68:E6:03:77:C9:
MAC source: 8C:61:A3:67:68:C8:

Tipo: 2048   08 00
Paquete IP..
0100 .... = Version: 4
.... 0101 = Header Length : 20 bytes (5)
Longitud total: 57
Servicios Diferenciados: [precedence: 000 (Routine)] [ECN: 00 (Sin capacidad ECN)]
ID: 00 00
----Flags
Don't Fragment: 1 Encendido
More: 0 Apagado
Fragment offset: 00 00 (00)
TTL: 76 (118)
Protocolo: 11 UDP
Checksum: 7E 9E
Source IP Address: 189.216.7.145
Destination IP Address: 192.168.0.4
Protocol UDP
       Source port: 01 BB
       Secuense number: FF 1F
       Length: 00 25
       Checksum: 7E 60
```

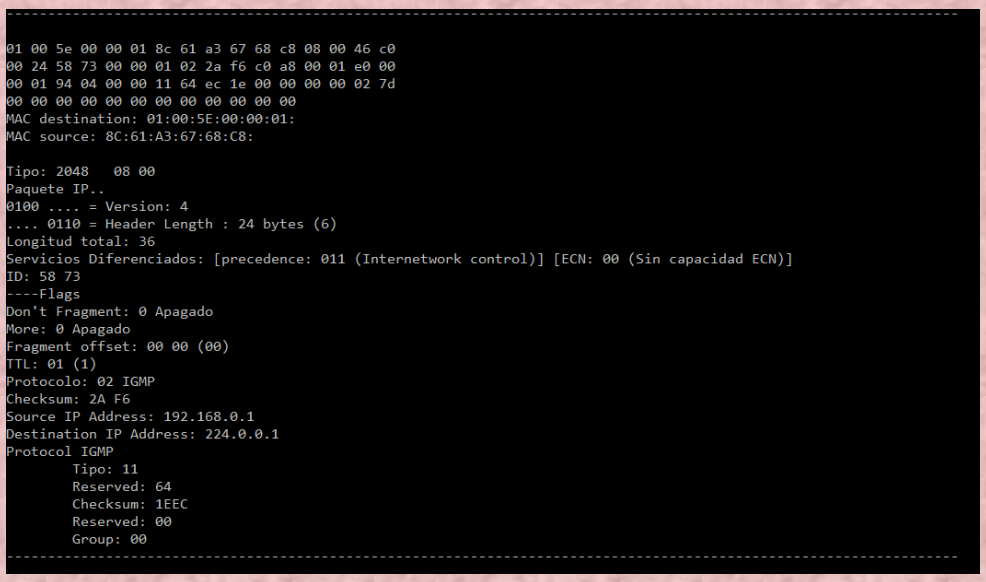
```
40 2b 50 f0 9b 86 3c f0 11 52 07 0b 08 00 45 00
00 4a aa 33 00 00 80 11 00 00 c0 a8 00 08 0a 93
70 02 f0 3f 00 35 00 36 3b 8d 71 2a 01 00 00 01
00 00 00 00 00 00 08 70 72 65 73 65 6e 63 65 05
74 65 61 6d 73 09 6d 69 63 72 6f 73 6f 66 74 03
63 6f 6d 00 00 01 00 01
MAC destination: 40:2B:50:F0:9B:86:
MAC source: 3C:F0:11:52:07:0B:

Tipo: 2048   08 00
Paquete IP..
0100 .... = Version: 4
.... 0101 = Header Length : 20 bytes (5)
Longitud total: 74
Servicios Diferenciados: [precedence: 000 (Routine)] [ECN: 00 (Sin capacidad ECN)]
ID: AA 33
----Flags
Don't Fragment: 0 Apagado
More: 0 Apagado
Fragment offset: 00 00 (00)
TTL: 80 (128)
Protocolo: 11 UDP
Checksum: 00 00
Source IP Address: 192.168.0.8
Destination IP Address: 10.147.112.2
Protocol UDP
       Source port: F0 3F
       Secuense number: 00 35
       Length: 00 36
       Checksum: 3B 8D
```

2.4. Tramas TCP

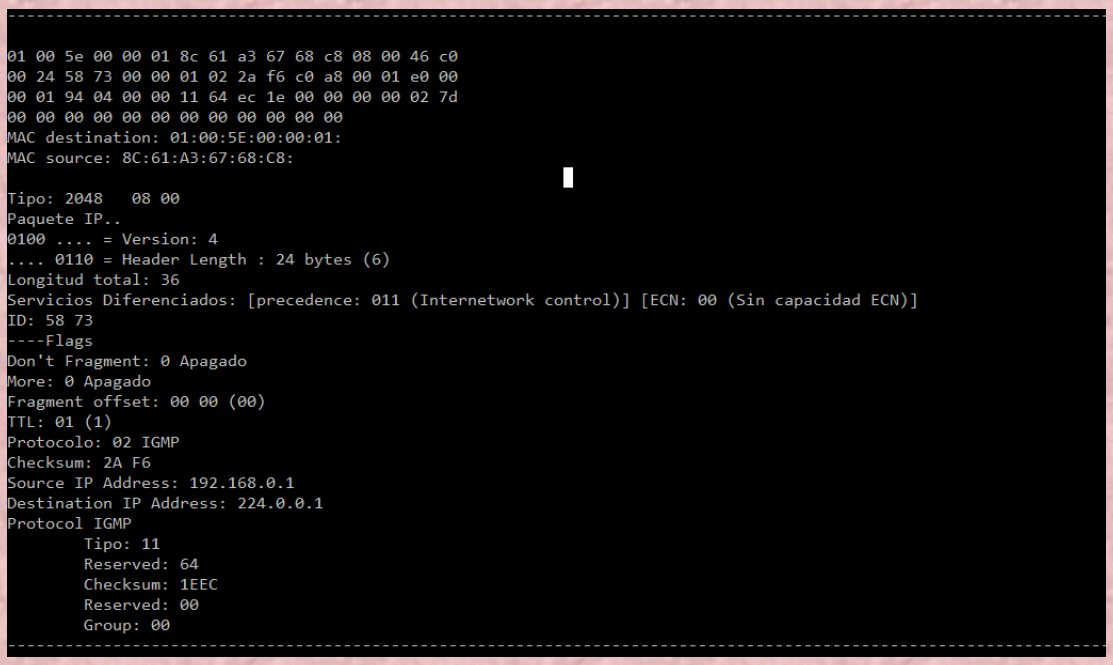
```
8c 61 a3 67 68 c8 b0 68 e6 03 77 c9 08 00 45 00
00 8d 01 b2 40 00 80 06 5c 7b c0 a8 00 04 34 b1
a6 e0 dc 3c 01 bb e3 cd 58 98 c7 28 54 56 50 18
02 03 6a e5 00 00 17 03 03 00 60 00 00 00 00 00
00 00 1b c4 b0 d8 d8 7c 5f 69 28 5a 6f 37 35 0d
04 20 9a 2b 30 fe 8e ce e8 81 66 2b 01 f9 a8 21
51 05 eb f6 99 98 67 79 53 48 61 2d 77 58 80 88
78 21 cf 24 8d 27 0b 16 0f 25 74 e2 a7 70 d2 7b
ac c4 8e 05 9e 5c bf 53 10 07 74 4d c5 19 f8 d2
a8 94 3a ae b5 3b 71 90 b3 b6 6e
MAC destination: 8C:61:A3:67:68:C8:
MAC source: B0:68:E6:03:77:C9:

Tipo: 2048   08 00
Paquete IP..
0100 .... = Version: 4
.... 0101 = Header Length : 20 bytes (5)
Longitud total: 141
Servicios Diferenciados: [precedence: 000 (Routine)] [ECN: 00 (Sin capacidad ECN)]
ID: 01 B2
----Flags
Don't Fragment: 1 Encendido
More: 0 Apagado
Fragment offset: 00 00 (00)
TTL: 80 (128)
Protocolo: 06 Protocol TCP
        Source port: DC 3C
        Destination port: 01 BB
        Secuense number: E3 CD 58 98
        Ack number: C7 28 54 56
        Checksum: 6A E5
        Data Offset: 5
        Flags: 00011000 ACK PSH
        Urgent Pointer: 00  00
```

```
b0 68 e6 03 77 c9 8c 61 a3 67 68 c8 08 00 45 00
00 a2 4d d9 40 00 6c 06 24 3f 34 b1 a6 e0 c0 a8
00 04 01 bb dc 3c c7 28 54 56 e3 cd 58 fd 50 18
1c 0e 32 d8 00 00 17 03 03 00 75 00 00 00 00 00
00 00 1c 8d da ba c2 d6 0a 45 db ad 70 a3 d4 1e
94 1a 07 80 9c a2 ef 15 53 9a 34 ea 35 99 d1 81
93 35 c7 67 8c db 28 43 d0 7e 00 6d 9a 22 54 27
de 0f 39 b2 ea 73 14 9e 7d 8d 0c d2 e3 18 8d b8
b2 71 ca aa a8 88 1f a5 b3 d6 b3 ec e4 ec f7 6c
45 fd c1 c8 1b 67 cf f8 36 9d 02 8f 07 3d 87 68
75 14 00 9f 4a 8f 84 d3 83 45 41 e9 d7 cc 5b ba

MAC destination: B0:68:E6:03:77:C9:
MAC source: 8C:61:A3:67:68:C8:

Tipo: 2048   08 00
Paquete IP..
0100 .... = Version: 4
.... 0101 = Header Length : 20 bytes (5)
Longitud total: 162
Servicios Diferenciados: [precedence: 000 (Routine)] [ECN: 00 (Sin capacidad ECN)]
ID: 4D D9
----Flags
Don't Fragment: 1 Encendido
More: 0 Apagado
Fragment offset: 00 00 (00)
TTL: 6C (108)
Protocolo: 06 Protocol TCP
        Source port: 01 BB
        Destination port: DC 3C
        Secuense number: C7 28 54 56
        Ack number: E3 CD 58 FD
        Checksum: 32 D8
        Data Offset: 5
        Flags: 00011000 ACK PSH
        Urgent Pointer: 00  00
```

2.5. Tramas IGMP



```
01 00 5e 00 00 01 8c 61 a3 67 68 c8 08 00 46 c0
00 24 58 73 00 00 01 02 2a f6 c0 a8 00 01 e0 00
00 01 94 04 00 00 11 64 ec 1e 00 00 00 00 02 7d
00 00 00 00 00 00 00 00 00 00 00 00
MAC destination: 01:00:5E:00:00:01:
MAC source: 8C:61:A3:67:68:C8:

Tipo: 2048   08 00
Paquete IP..
0100 .... = Version: 4
.... 0110 = Header Length : 24 bytes (6)
Longitud total: 36
Servicios Diferenciados: [precedence: 011 (Internetwork control)] [ECN: 00 (Sin capacidad ECN)]
ID: 58 73
----Flags
Don't Fragment: 0 Apagado
More: 0 Apagado
Fragment offset: 00 00 (00)
TTL: 01 (1)
Protocolo: 02 IGMP
Checksum: 2A F6
Source IP Address: 192.168.0.1
Destination IP Address: 224.0.0.1
Protocol IGMP
       Tipo: 11
       Reserved: 64
       Checksum: 1EEC
       Reserved: 00
       Group: 00
```




```
01 00 5e 00 00 01 8c 61 a3 67 68 c8 08 00 46 c0
00 24 58 73 00 00 01 02 2a f6 c0 a8 00 01 e0 00
00 01 94 04 00 00 11 64 ec 1e 00 00 00 00 02 7d
00 00 00 00 00 00 00 00 00 00 00 00
MAC destination: 01:00:5E:00:00:01:
MAC source: 8C:61:A3:67:68:C8:

Tipo: 2048   08 00
Paquete IP..
0100 .... = Version: 4
.... 0110 = Header Length : 24 bytes (6)
Longitud total: 36
Servicios Diferenciados: [precedence: 011 (Internetwork control)] [ECN: 00 (Sin capacidad ECN)]
ID: 58 73
----Flags
Don't Fragment: 0 Apagado
More: 0 Apagado
Fragment offset: 00 00 (00)
TTL: 01 (1)
Protocolo: 02 IGMP
Checksum: 2A F6
Source IP Address: 192.168.0.1
Destination IP Address: 224.0.0.1
Protocol IGMP
       Tipo: 11
       Reserved: 64
       Checksum: 1EEC
       Reserved: 00
       Group: 00
```

2.6. Tramas ICMP

```
b0 68 e6 03 77 c9 8c 61 a3 67 68 c8 08 00 45 00
00 78 4a 60 00 00 2c 01 61 41 2e 34 f4 03 c0 a8
00 04 0b 00 21 78 00 00 00 00 45 70 00 5c 93 e3
00 00 00 01 e7 89 c0 a8 00 04 5e 1c 1e fc 08 00
cb f0 00 01 00 96 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00
MAC destination: B0:68:E6:03:77:C9:
MAC source: 8C:61:A3:67:68:C8:

Tipo: 2048   08 00
Paquete IP..
0100 .... = Version: 4
.... 0101 = Header Length : 20 bytes (5)
Longitud total: 120
Servicios Diferenciados: [precedence: 000 (Routine)] [ECN: 00 (Sin capacidad ECN)]
ID: 4A 60
----Flags
Don't Fragment: 0 Apagado
More: 0 Apagado
Fragment offset: 00 00 (00)
TTL: 2C (44)
Protocolo: 01 ICMP
Checksum: 61 41
Source IP Address: 46.52.244.3
Destination IP Address: 192.168.0.4
Protocol ICMP
        Tipo: 11
        Codigo:0
        Description:    Time Exceeded ---> Time to live exceeded in transit
        Checksum: 21 78
```

```
8c 61 a3 67 68 c8 b0 68 e6 03 77 c9 08 00 45 00
00 5c 93 e3 00 00 11 01 d7 f9 c0 a8 00 04 5e 1c
1e fc 08 00 f7 68 00 01 00 96 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00
MAC destination: 8C:61:A3:67:68:C8:
MAC source: B0:68:E6:03:77:C9:

Tipo: 2048   08 00
Paquete IP..
0100 .... = Version: 4
.... 0101 = Header Length : 20 bytes (5)
Longitud total: 92
Servicios Diferenciados: [precedence: 000 (Routine)] [ECN: 00 (Sin capacidad ECN)]
ID: 93 E3
----Flags
Don't Fragment: 0 Apagado
More: 0 Apagado
Fragment offset: 00 00 (00)
TTL: 11 (17)
Protocolo: 01 ICMP
Checksum: D7 F9
Source IP Address: 192.168.0.4
Destination IP Address: 94.28.30.252
Protocol ICMP
        Tipo: 8
        Codigo:0
        Description:    Echo Request ---> Echo request
        Checksum: F7 68
```

3. Conclusiones

CORTES LÓPEZ JAIME ALEJANDRO

Durante la realización de este proyecto se pudieron obtener conocimientos respecto al análisis de protocolos, y se reforzaron los conocimientos acerca de los protocolos más importantes de internet.

Se pudo llegar a la conclusión de que los protocolos son fundamentales para el internet y por lo tanto en la vida actual. Se pudo llegar a comprender de mejor manera el modelo de referencia OSI, ya que se trabajó con protocolos de distintas capas.

Además, se pudo comprender de mejor forma cómo funcionan las redes de computadoras, y se llegó a la conclusión de que internet está compuesto de distintos protocolos que son los que rigen la forma del intercambio de información para hacerlo de forma ordenada.

GONZÁLEZ MORA ERIKA GISELLE

Haber realizado este analizador de protocolos en equipo, fue muy interesante y tuvo su nivel de complejidad respecto a cómo programar cada uno de los protocolos vistos en clase, sin embargo pude comprender de una manera muy completa, como se conforma un analizador de protocolos, y cada una de sus características. El haber concluido satisfactoriamente con el proyecto, me hizo reaccionar acerca de lo que significan las redes de computadoras, hoy en día; básicamente en la actualidad es muy raro que nadie tenga acceso a internet, y este trabajo nos introdujo precisamente a estudiar y programar los protocolos más importantes que se manejan dentro de él.

MARTÍNEZ MARTÍNEZ FERNANDO

Como en parte de prácticas anteriores, en donde tuvimos la oportunidad de analizar individualmente los protocolos comprendidos en la unidad de aprendizaje, pudimos notar la importancia de conocer adecuadamente los campos de información que están comprendidos al interior de las tramas que los encapsulan, pues de otra forma, podría cometerse el error de no identificarlos adecuadamente y por tanto confundir la información que contienen. Por otro lado, nos permitió notar la forma en que el analizador, diseñado para este proyecto, puede mejorarse, pues al menos en el paradigma orientado a objetos, podemos modelar comportamientos similares entre protocolos, lo cual

nos permitiría reducir líneas de código, pero además, poder escalar para el análisis de otros protocolos. Finalmente, pudimos notar la aplicación en la realidad, pues gran parte de los protocolos analizados, son utilizados por los dispositivos conectados a la red en casa, aunque a priori no los notemos, son estos los responsables de conectarnos a internet y permitirnos acceder a los servicios que es capaz de ofrecer.

OLIVARES MENÉZ GLORIA OLIVA

Gracias a la realización de este proyecto se pudo observar a mejor detalle el funcionamiento de algunos de los protocolos más utilizados, así como conocer algunas de sus características más importantes y sobre todo, se aprendió cómo es que se puede identificar cada uno de estos protocolos.

Además de ello, también se pudo observar el comportamiento de la información que fluye a través de nuestras propias redes, y se pudo observar también la frecuencia con la que se utilizan algunos protocolos, esto gracias a la opción que brinda el proyecto en la cual se muestran estadísticas sobre la cantidad de tramas que se analizaron dependiendo de los protocolos que tenían implementados.

Finalmente, se pudo comprobar la gran importancia que tienen los protocolos para el correcto funcionamiento del internet en general, ya que, sin los protocolos, sería prácticamente imposible analizar o decodificar las tramas y la información que viaja a través de internet.

VAZQUEZ PEREZ DENZEL OMAR

Es muy importante el conocer cómo es que viaja la información a través de la red, así como debemos de saber las ventajas y desventajas de los protocolos de internet para poder conocer más allá de los errores que se pueden presentar en la conexión de dos o más equipos a través de la red, así como el origen y las causas de los ataques cibernéticos que se presentan en la actualidad. Al comprender la estructura de cada protocolo de internet se nos facilitará el análisis e identificación de las tramas Ethernet que viajan por nuestra computadora y nuestra red LAN que tenemos en nuestro hogar.

El realizar este proyecto fue una gran y grata experiencia; pudimos aprender mucho del análisis de los protocolos de red que existen en la actualidad, y el trabajo cooperativo fue parte fundamental para

el trabajo final que se desarrolló. Cada integrante tiene una forma distinta de entender las cosas y de plantearlas en un programa computacional, pero con ayuda de las librerías previamente cargadas, así como las clases teóricas de la unidad de aprendizaje de Redes de Computadoras, pudimos llegar a un acuerdo y unificar los conocimientos para plantearlos en este programa analizador de protocolos. Me gustó mucho el trabajo final que desarrollamos en conjunto.

4. Referencias

- "Redes de Comunicación". 1ª edición. Alberto León-García, Indra Widjaja. Ed. Mc Graw Hill. 2001.
- "Internetworking with TCP/IP Vol. I, Principles, Protocols, and Architecture)". 4th edition. Douglas E. Comer. Ed. Prentice Hall, 2000.
- "Computer Networks and Internets". 4th edition. Douglas E. Comer, Ralph E. Droms. Ed. Prentice Hall, 2003..
- "Tecnologías de Interconectividad de Redes". Merilee Ford. Ed. Prentice-Hall. 1998.
- "Redes de computadores: un enfoque descendente basado en Internet". James F. Kurose, Keith W. Ross. Ed. Addison-Wesley, 2004.
- "Inside TCP/IP". 3rd edition. Karanjit S. Siyan. Ed. New Riders. 1997.

5. Anexos

a) Códigos

- PROJECT_REDES_MAIN.C

```
#include <stdio.h>
#include <stdlib.h>
#include <winsock2.h>
#include<windows.h>
#include <math.h>
#include "C:\\Users\\gerik\\OneDrive\\Escritorio\\Include\\pcap.h"
#ifdef _MSC_VER
#define _CRT_SECURE_NO_WARNINGS
#endif
#define LINE_LEN 16
//#include <pcap.h>
#include <string.h>
#include "ARPTrama.h"
#include "LLCTrama.h"
#include "IPTrama.h"
```

```
#include "ICMPTrama.h"
#include "IGMPTrama.h"
#include "TCPTrama.h"
#include "UDPTrama.h"
#define RUTA_IEEE
"C:\\Users\\gerik\\OneDrive\\Escritorio\\UNIVERSIDAD\\ESCOM\\CUARTO
SEMESTRE\\REDES I\\SEGUNDO PARCIAL\\Project_redes\\LLC.pcap"
#define RUTA_IP
"C:\\Users\\gerik\\OneDrive\\Escritorio\\UNIVERSIDAD\\ESCOM\\CUARTO
SEMESTRE\\REDES I\\SEGUNDO PARCIAL\\Project_redes\\IP.pcap"
#define RUTA_ARP
"C:\\Users\\gerik\\OneDrive\\Escritorio\\UNIVERSIDAD\\ESCOM\\CUARTO
SEMESTRE\\REDES I\\SEGUNDO PARCIAL\\Project_redes\\ARP.pcap"
#define RUTA_ICMP
"C:\\Users\\gerik\\OneDrive\\Escritorio\\UNIVERSIDAD\\ESCOM\\CUARTO
SEMESTRE\\REDES I\\SEGUNDO PARCIAL\\Project_redes\\ICMP.pcap"
#define RUTA_IGMP
"C:\\Users\\gerik\\OneDrive\\Escritorio\\UNIVERSIDAD\\ESCOM\\CUARTO
SEMESTRE\\REDES I\\SEGUNDO PARCIAL\\Project_redes\\IGMP2.pcap"
#define RUTA_TCP
"C:\\Users\\gerik\\OneDrive\\Escritorio\\UNIVERSIDAD\\ESCOM\\CUARTO
SEMESTRE\\REDES I\\SEGUNDO PARCIAL\\Project_redes\\TCP.pcap"
#define RUTA_UDP
"C:\\Users\\gerik\\OneDrive\\Escritorio\\UNIVERSIDAD\\ESCOM\\CUARTO
SEMESTRE\\REDES I\\SEGUNDO PARCIAL\\Project_redes\\UDP.pcap"
#define      PCAP_SRC_FILE   2
#define      PCAP_BUF_SIZE   1024


int main(){
      char *Resp = malloc(20);
      do{
            pcap_if_t *alldevs;
            pcap_if_t *d;
            pcap_t *fp;
            char errbuf[PCAP_ERRBUF_SIZE];
            char source[PCAP_BUF_SIZE];
            int inum=0,i=0;
            int captura_tramas;
            int Tipo_Protocolo;
            pcap_t *adhandle;

            system("cls");
            printf("\t-*-*-*-*-*-*-*P R O T O C O L---A N A L Y Z E R*-*-*-*-*-
*-*-\nElija una opcion para capturar las tramas\n 1. Tramas al vuelo\n 2. Por
archivo\nTeclee la opcion a elegir (1-2): ");
            scanf("%d",&captura_tramas);
            while(captura_tramas!=1 && captura_tramas!=2){
                  printf("Seleccione una opcion existente.\n Opcion: ");
                  scanf("%d",&captura_tramas);
            }
            system("cls");
            printf("SELECCIONA EL PROTOCOLO A ANALIZAR\n 1. LLC\n 2. ARP\n 3.
IP\n 4. ICMP\n 5. IGMP\n 6. TCP\n 7. UDP\n");
            printf("teclee la opcion a elegir: ");
```

```
		scanf("%d",&Tipo_Protocolo);
		while(Tipo_Protocolo<1 && Tipo_Protocolo>8){
			printf("Seleccione una opción existente.\n");
			scanf("%d",&Tipo_Protocolo);
		}

		switch(captura_tramas){

			case 1:{
				if(pcap_findalldevs(&alldevs, errbuf) == -1)
				{
					fprintf(stderr,"Error in pcap_findalldevs:
%s\n", errbuf);
					exit(1);
				}

				/* Print the list */
				for(d=alldevs; d; d=d->next)
				{
					printf("%d. %s", ++i, d->name);
					if (d->description)
						printf(" (%s)\n", d->description);
					else
						printf(" (No description available)\n");
				}

				if(i==0)
				{
					printf("\nNo interfaces found! Make sure WinPcap
is installed.\n");
					return -1;
				}

				printf("Enter the interface number (1-%d):",i);
				scanf("%d", &inum);

				if(inum < 1 || inum > i)
				{
					printf("\nInterface number out of range.\n");
					/* Free the device list */
					pcap_freealldevs(alldevs);
					return -1;
				}

				/* Jump to the selected adapter */
				for(d=alldevs, i=0; i< inum-1 ;d=d->next, i++);

				/* Open the device */
				/* Open the adapter */
				if ((adhandle= pcap_open_live(d->name, // name of the
device
							65536,
		// portion of the packet to capture.
```

```
            // 65536 grants that the whole packet will be captured on all the
MACs.
                                                                  1,
            // promiscuous mode (nonzero means promiscuous)
                                                                  5000,
      // read timeout
                                                                  errbuf
            // error buffer
                                                                  )) == NULL)
                        {
                              fprintf(stderr,"\nUnable to open the adapter. %s
is not supported by WinPcap\n", d->name);
                              /* Free the device list */
                              pcap_freealldevs(alldevs);
                              return -1;
                        }

                        printf("\nlistening on %s...\n", d->description);

                        /* At this point, we don't need any more the device
list. Free it */
                        pcap_freealldevs(alldevs);

                        break;
                  }
                  case 2:{
                        switch(Tipo_Protocolo){
                              case 1:{
                                    if ( pcap_createsrcstr( source,
// variable that will keep the source string
                                                PCAP_SRC_FILE,  // we want to
open a file
                                                NULL,           // remote host
                                                NULL,           // port on the
remote host
                                                RUTA_IEEE, //argv[1],        //
name of the file we want to open
                                                errbuf          // error buffer
                                                ) != 0)
                        {
                            fprintf(stderr,"\nError creating a source string\n");
                            return -1;
                        }
                                    break;
                              }
                              case 2:{
                                    if ( pcap_createsrcstr( source,
// variable that will keep the source string
                                                PCAP_SRC_FILE,  // we want to
open a file
                                                NULL,           // remote host
                                                NULL,           // port on the
remote host
```

```
                                                RUTA_ARP, //argv[1],        //
name of the file we want to open
                                                errbuf          // error buffer
                                                ) != 0)
                        {
                            fprintf(stderr,"\nError creating a source string\n");
                            return -1;
                        }
                                    break;
                                }
                                case 3:{
                                    if ( pcap_createsrcstr( source,
// variable that will keep the source string
                                                PCAP_SRC_FILE,  // we want to
open a file
                                                NULL,           // remote host
                                                NULL,           // port on the
remote host
                                                RUTA_IP, //argv[1],        //
name of the file we want to open
                                                errbuf          // error buffer
                                                ) != 0)
                        {
                            fprintf(stderr,"\nError creating a source string\n");
                            return -1;
                        }
                                    break;
                                }
                                case 4:{
                                    if ( pcap_createsrcstr( source,
// variable that will keep the source string
                                                PCAP_SRC_FILE,  // we want to
open a file
                                                NULL,           // remote host
                                                NULL,           // port on the
remote host
                                                RUTA_ICMP, //argv[1],        //
name of the file we want to open
                                                errbuf          // error buffer
                                                ) != 0)
                        {
                            fprintf(stderr,"\nError creating a source string\n");
                            return -1;
                        }
                                    break;
                                }
                                case 5:{
                                    if ( pcap_createsrcstr( source,
// variable that will keep the source string
                                                PCAP_SRC_FILE,  // we want to
open a file
                                                NULL,           // remote host
                                                NULL,           // port on the
remote host
```

```
                                                RUTA_IGMP, //argv[1],        //
name of the file we want to open
                                                errbuf          // error buffer
                                                ) != 0)
                                            {
                                                fprintf(stderr,"\nError creating
a source string\n");
                                                return -1;
                                            }
                                        break;
                                    }
                                    case 6:{
                                        if ( pcap_createsrcstr( source,
// variable that will keep the source string
                                                PCAP_SRC_FILE,  // we want to
open a file
                                                NULL,           // remote host
                                                NULL,           // port on the
remote host
                                                RUTA_TCP, //argv[1],        //
name of the file we want to open
                                                errbuf          // error buffer
                                                ) != 0)
                                        {
                                            fprintf(stderr,"\nError creating a
source string\n");
                                            return -1;
                                        }
                                        break;
                                    }
                                    case 7:{
                                        if ( pcap_createsrcstr( source,
// variable that will keep the source string
                                                PCAP_SRC_FILE,  // we want to
open a file
                                                NULL,           // remote host
                                                NULL,           // port on the
remote host
                                                RUTA_UDP, //argv[1],        //
name of the file we want to open
                                                errbuf          // error buffer
                                                ) != 0)
                                {
                                    fprintf(stderr,"\nError creating a source string\n");
                                    return -1;
                                }
                                        break;
                                    }
                                }
                                if ( (adhandle= (pcap_t *)pcap_open(source,         //
name of the device
                                    65536,          // portion of the packet to
capture
                                                // 65536 guarantees that the
whole packet will be captured on all the link layers
```

```
                                PCAP_OPENFLAG_PROMISCUOUS,     // promiscuous
mode
                                1000,              // read timeout
                                NULL,              // authentication on the
remote machine
                                errbuf         // error buffer
                                ) ) == NULL)
                        {
                            fprintf(stderr,"\nUnable to open the file %s\n",
source);
                            return -1;
                        }
                             break;
                    }

            }

            if(Tipo_Protocolo==1){
                  pcap_loop(adhandle, 15, IEEE_Trama, NULL);
                  pcap_close(adhandle);
            }
            if(Tipo_Protocolo==2){
                  pcap_loop(adhandle, 15, ARP_Trama, NULL);
                  pcap_close(adhandle);
            }
            if(Tipo_Protocolo==3){
                  pcap_loop(adhandle, 15, IP_Trama, NULL);
                  pcap_close(adhandle);
            }
            if(Tipo_Protocolo==4){
                  pcap_loop(adhandle, 15, ICMP_Trama, NULL);
                  pcap_close(adhandle);
            }
            if(Tipo_Protocolo==5){
                  pcap_loop(adhandle,41, IGMP_Trama, NULL);
                  pcap_close(adhandle);
            }
            if(Tipo_Protocolo==6){
                  pcap_loop(adhandle, 15, TCP_Trama, NULL);
                  pcap_close(adhandle);
            }
            if(Tipo_Protocolo==7){
                  pcap_loop(adhandle, 15, UDP_Trama, NULL);
                  pcap_close(adhandle);
            }
            printf("\n");
      printf("-----------------------------------------------------------------
-------------------------------------------------\n\n");
            puts("Desea hacer otra consulta: ");
            scanf("%s",Resp);
      }while(strcmp(Resp,"si")==0 || strcmp(Resp,"")==0 ||
strcmp(Resp,"si")==0);

      puts("Vuelva pronto");
```

```
	return 0;
}
```

- LLCTRAMA.H

```
const char *byte_to_binary(unsigned short i){
	static char  b[9];
	b[0]= '\0';
	int z;
	for(z = 128; z>0; z>>=1){
		strcat(b,((i&z) == z)? "1" : "0");
	}
	return b;
	free(b);
}

const char *Trama7(unsigned short i){
	static char  b[9];
	b[0]= '\0';
	int z;
	for(z = 64; z>0; z>>=1){
		strcat(b,((i&z) == z)? "1" : "0");
	}
	return b;
	free(b);
}

const char *Trama2(unsigned short i){
	static char  b[3];
	b[0]= '\0';
	int z;
	for(z = 3; z>0; z>>=1){
		strcat(b,((i&z) == z)? "1" : "0");
	}
	return b;
	free(b);
}

const char *Trama3(unsigned short i){
	static char  b[4];
	b[0]= '\0';
	int z;
	for(z = 4; z>0; z>>=1){
		strcat(b,((i&z) == z)? "1" : "0");
	}
	return b;
	free(b);
}

const char *Trama1(unsigned short i){
	static char  b[2];
	b[0]= '\0';
	int z;
	for(z = 1; z>0; z>>=1){
```

```
            strcat(b,((i&z) == z)? "1" : "0");
      }
      return b;
      free(b);
}

void Protocolo(unsigned short i){
      printf("Protocolo SAP: ");
      if(i==0)
            printf("NULL SAP\n");
      else if(i==4)
            printf("SNA\n");
      else if(i==5)
            printf("SNA\n");
      else if(i==6)
            printf("TCP\n");
      else if(i==8)
            printf("SNA\n");
      else if(i==12)
            printf("SNA\n");
      else if(i==66)
            printf("spanning tree\n");
      else if(i==127)
            printf("ISO IEEE 802.2\n");
      else if(i==128)
            printf("XNS\n");
      else if(i==170)
            printf("SNAP\n");
      else if(i==224)
            printf("IPX\n");
      else if(i==240)
            printf("NETBIOS\n");
      else if(i==248)
            printf("RPL\n");
      else if(i==252)
            printf("RPL\n");
      else if(i==254)
            printf("OSI\n");
      else if(i==255)
            printf("Global SAP\n");
      else
            printf("NULL\n");
}

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

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

    /* print pkt timestamp and pkt len */
```

```
    printf("%ld:%ld (%ld)\n", header->ts.tv_sec, header->ts.tv_usec, header-
>len);

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

    printf("\nMAC destino:\n");
      for(j=0;j<6;j++){
         printf("%02X ",pkt_data[j]);
      }
      printf("\nMAC origen:\n");
      for(k=6;k<12;k++){
         printf("%02X ",pkt_data[k]);
      }

    unsigned short tipo = (pkt_data[12]*256)+pkt_data[13];
    printf("\nSIZE: %d   %02X %02X \n",tipo,pkt_data[12],pkt_data[13]);

    unsigned short DSAP= pkt_data[14];
    printf("\nCampo DSAP: %s\t%02X\nIoG:%d\t",
byte_to_binary(DSAP),DSAP,DSAP&0x01);
    if(strcmp(Trama1(DSAP&0x01), "0")== 0){
      puts("Individual");
      }
      else{
            puts("Grupo");
      }
      Protocolo(DSAP);
      DSAP = DSAP >> 1;
      printf("Direccion Destino: %s\n\n",Trama7(DSAP&0x7f));

    unsigned short SSAP= pkt_data[15];
    printf("\nCampo SSAP: %s\t%02X\nCoR:%d\t",
byte_to_binary(SSAP),SSAP,SSAP&0x01);
    if(strcmp(Trama1(SSAP&0x01), "0")== 0){
      puts("Comando");
      }
      else{
            puts("Respuesta");
      }
    Protocolo(SSAP);
      SSAP = SSAP >> 1;
      printf("Direccion origen: %s\n\n",Trama7(SSAP&0x7f));

    if(tipo<1500){
            printf("CAMPO LONGITUD\n");
      if(tipo<=3){
            printf("MODELO NORMAL\n");
            unsigned short Cp = pkt_data[16]>>1;
            unsigned short C = pkt_data[16]&0x01;
            if(C==0){
```

```
                        printf("Trama I\n");
                  unsigned short TI= pkt_data[16];
                  unsigned short TI_copia= TI;
                  unsigned short TI_copia_1= TI;
                  printf("Trama I: %s\t%02X\n", byte_to_binary(TI),TI);
                  TI = TI >> 1;
                        printf("Numero de secuencia solicitada:
%s\n",Trama3(TI&0x07));
                        TI_copia = TI_copia >> 4;
                  printf("P/F: %d\n", TI_copia&0x01);
                  TI_copia_1 = TI_copia_1 >>5;
                        printf("Numero de secuencia esperada:
%s\n",Trama7(TI&0x7f));
            }
                  else{
                        Cp=Cp&0x01;
                        if(Cp==0){
                              unsigned short TS= pkt_data[16];
                        unsigned short TS_copia= TS;
                        unsigned short TS_copia_1= TS;
                        unsigned short TS_copia_2= TS;
                        printf("Trama S: %s\t%02X\n", byte_to_binary(TS),TS);
                        printf("S: %d\n", TS&0x03);
                              TS_copia = TS_copia>>2;
                              printf("Codigo Trama S: %s\n",Trama3(TS&0x03));
                              if(strcmp(Trama2(TS&0x03), "00")== 0)
                                    printf("Listo para recibir(RR)\n");
                              else if(strcmp(Trama2(TS&0x03), "01")== 0 )
                                    printf("Receptor no listo para
recibir(RNR)");
                              else if(strcmp(Trama2(TS&0x03), "10")== 0 )
                                    printf("Rechazo(REJ)");
                              else if(strcmp(Trama2(TS&0x03), "11")== 0 )
                                    printf("Rechazo selectivo(SREJ)");
                              TS_copia_1 = TS_copia_1>>4;
                        printf("P/F: %d\n", TS_copia&0x01);
                        TS_copia_2 = TS_copia_2>>5;
                              printf("Numero de secuencia esperada:
%s\n",Trama7(TS_copia_2&0x7f));

                        }
                        else{
                              printf("Control\n");
                              unsigned short TU = pkt_data[16];
                              unsigned short TU_copia= TU;
                              unsigned short TU_copia_1 = TU;
                              unsigned short TU_copia_2 = TU;
                              unsigned short CR= pkt_data[15];
                              printf("Trama U:
%s\t%02X\n",byte_to_binary(TU),TU);
                              printf("U: %s\n",Trama2(TU&0x03));
                              TU_copia = TU_copia >> 2;
                              TU_copia_1 = TU_copia_1 >> 5;
                              TU_copia_2 = TU_copia_2 >> 4;
                              printf("P/F: %d\n", TU_copia_2&0x01);
```

```
                                printf("Codigo Trama U: %s
",Trama3(TU_copia_1&0x07));
                                printf("%s\n",Trama2(TU_copia&0x03));

                                if(strcmp(Trama1(CR&0x01), "0")== 0 &&
strcmp(Trama1(TU_copia_2&0x01), "1") == 0){
                                printf("Comando con Protocolo: ");
                                if(strcmp(Trama3(TU_copia_1&0x07), "100")== 0 &&
strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                                printf("SNRM");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"110")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                                printf("SNRME");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"001")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                                printf("SABM");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"011")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                                printf("SABME");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"000")== 0 && strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                                printf("UI");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"011")== 0 && strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                                printf("NULL");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"010")== 0 && strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                                printf("DISC");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"000")== 0 && strcmp(Trama2(TU_copia&0x03), "01")== 0 )
                                                printf("SIM");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"001")== 0 && strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                                printf("UP");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"100")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                                printf("RSET");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"101")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                                printf("XID");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"100")== 0 && strcmp(Trama2(TU_copia&0x03), "01")== 0 )
                                                printf("FRMR");
                                }
                                else if(strcmp(Trama1(CR&0x01), "1")== 0 &&
strcmp(Trama1(TU_copia_2&0x01), "1") == 0){
                                        printf("Respuesta con Protocolo: ");
                                if(strcmp(Trama3(TU_copia_1&0x07), "100")== 0 &&
strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                                printf("NULL");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"110")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                                printf("NULL");
```

```
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"001")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                              printf("DM");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"011")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                              printf("NULL");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"000")== 0 && strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                              printf("UI");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"011")== 0 && strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                              printf("UA");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"010")== 0 && strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                              printf("RD");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"000")== 0 && strcmp(Trama2(TU_copia&0x03), "01")== 0 )
                                              printf("RIM");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"001")== 0 && strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                              printf("NULL");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"100")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                              printf("NULL");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"101")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                              printf("XID");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"100")== 0 && strcmp(Trama2(TU_copia&0x03), "01")== 0 )
                                              printf("FRMR");
                                  }
                                  else
                                        puts("El bit P/F esta apagado");

                            }
                      }
                }
      else{
            printf("MODELO EXTENDIDO\n");
           unsigned short Ccp = pkt_data[16]>>1;
            unsigned short C1 = pkt_data[16]&0x01;
                  if(C1==0){
                        printf("Trama I\n");
                  unsigned short TI_p1 = pkt_data[16];
                  unsigned short TI_p2 = pkt_data[17];
                  unsigned short TI_copia_p1 = TI_p1&0x01;
                  unsigned short TI_copia_p2 = TI_p2;
                        printf("Trama I: %s  ", byte_to_binary(TI_p1));
                  printf("%s\t%02X %02X\n", byte_to_binary(TI_p2), TI_p1,
TI_p2);
                  printf("I: %d\n",TI_p1&0x01);
                  TI_p1 = TI_p1 >> 1;
                        printf("Numero de secuencia solicitada:
%s\n",Trama7(TI_p1&0x7f));
                  printf("P/F: %d\n", TI_p2&0x01);
```

```
                  TI_copia_p2 = TI_copia_p2 >> 1;
                        printf("Numero de secuencia esperada:
%s\n",Trama7(TI_copia_p2&0x7f));
                  }
                  else{
                        Ccp=Ccp&0x01;
                        if(Ccp==0){
                              printf("Trama S\n");
                              unsigned short TS_p1 = pkt_data[16];
                              unsigned short TS_p2 = pkt_data[17];
                              unsigned short TS_copia_p1 = TS_p1;
                              unsigned short TS_copia_p2 = TS_p2;
                              unsigned short TS_copia_p3 = TS_p2;
                              unsigned short TS_copia_p4 = TS_p1;
                              printf("Trama S: %s  ",byte_to_binary(TS_p1));
                              printf("%s\t%02X %02X\n", byte_to_binary(TS_p2),
TS_p1, TS_p2);
                              printf("S: %s\n",Trama2(TS_p1&0x03));
                              TS_copia_p1 = TS_copia_p1 >> 2;
                              printf("Codigo Trama S:
%s\t",Trama2(TS_copia_p1&0x03));

                              if(strcmp(Trama2(TS_copia_p1&0x03), "00")== 0)
                                    printf("Listo para recibir(RR)\n");
                              else if(strcmp(Trama2(TS_copia_p1&0x03), "01")==
0 )
                                    printf("Receptor no listo para
recibir(RNR)\n");
                              else if(strcmp(Trama2(TS_copia_p1&0x03), "10")==
0 )
                                    printf("Rechazo(REJ)\n");
                              else if(strcmp(Trama2(TS_copia_p1&0x03), "11")==
0 )
                                    printf("Rechazo selectivo(SREJ)\n");

                              printf("P/F: %d\n", TS_p2&0x01);
                        TS_copia_p2 = TS_copia_p2 >> 1;
                              printf("Numero de secuencia esperada:
%s\n",Trama7(TS_copia_p2&0x7f));
                        }
                        else{
                              unsigned short TU = pkt_data[16];
                              unsigned short TU_copia= TU;
                              unsigned short TU_copia_1 = TU;
                              unsigned short TU_copia_2 = TU;
                              printf("Trama U: %s\t%02X\n",byte_to_binary(TU),
TU);
                              printf("U: %s\n",Trama2(TU&0x03));
                              TU_copia = TU_copia >> 2;
                              TU_copia_1 = TU_copia_1 >> 5;
                              TU_copia_2 = TU_copia_2 >> 4;
                              printf("P/F: %d\n", TU_copia_2&0x01);
                              printf("Codigo Trama U: %s
",Trama3(TU_copia_1&0x07));
                              printf("%s\n",Trama2(TU_copia&0x03));
```

```
                                if((strcmp(Trama1(SSAP&0x01), "0")== 0 &&
strcmp(Trama1(TU_copia_2&0x01), "1") == 0)){
                                printf("Comando con Protocolo: ");
                                if(strcmp(Trama3(TU_copia_1&0x07), "100")== 0 &&
strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                                printf("SNRM");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"110")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                                printf("SNRME");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"001")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                                printf("SABM");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"011")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                                printf("SABME");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"000")== 0 && strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                                printf("UI");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"011")== 0 && strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                                printf("NULL");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"010")== 0 && strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                                printf("DISC");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"000")== 0 && strcmp(Trama2(TU_copia&0x03), "01")== 0 )
                                                printf("SIM");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"001")== 0 && strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                                printf("UP");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"100")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                                printf("RSET");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"101")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                                printf("XID");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"100")== 0 && strcmp(Trama2(TU_copia&0x03), "01")== 0 )
                                                printf("FRMR");
                                }
                                else if(strcmp(Trama1(SSAP&0x01), "1")== 0 &&
strcmp(Trama1(TU_copia_2&0x01), "1") == 0){
                                        printf("Respuesta con Protocolo: ");
                                if(strcmp(Trama3(TU_copia_1&0x07), "100")== 0 &&
strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                                printf("NULL");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"110")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                                printf("NULL");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"001")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                                printf("DM");
                                        else if(strcmp(Trama3(TU_copia_1&0x07),
"011")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
```

```
                                                printf("NULL");
                                          else if(strcmp(Trama3(TU_copia_1&0x07),
"000")== 0 && strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                                printf("UI");
                                          else if(strcmp(Trama3(TU_copia_1&0x07),
"011")== 0 && strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                                printf("UA");
                                          else if(strcmp(Trama3(TU_copia_1&0x07),
"010")== 0 && strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                                printf("RD");
                                          else if(strcmp(Trama3(TU_copia_1&0x07),
"000")== 0 && strcmp(Trama2(TU_copia&0x03), "01")== 0 )
                                                printf("RIM");
                                          else if(strcmp(Trama3(TU_copia_1&0x07),
"001")== 0 && strcmp(Trama2(TU_copia&0x03), "00")== 0 )
                                                printf("NULL");
                                          else if(strcmp(Trama3(TU_copia_1&0x07),
"100")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                                printf("NULL");
                                          else if(strcmp(Trama3(TU_copia_1&0x07),
"101")== 0 && strcmp(Trama2(TU_copia&0x03), "11")== 0 )
                                                printf("XID");
                                          else if(strcmp(Trama3(TU_copia_1&0x07),
"100")== 0 && strcmp(Trama2(TU_copia&0x03), "01")== 0 )
                                                printf("FRMR");
                                    }
                                    else
                                          puts("El bit P/F esta apagado\n");

                              }
                        }
                  }
      }
    else
      printf("CAMPO TIPO\n");
}
```

- ARPTRAMA.H

```
void ARP_Trama(u_char *param, const struct pcap_pkthdr *header, const u_char
*pkt_data)
{
      struct tm *ltime;
      char timestr[16];
      time_t local_tv_sec;

      /*
       * unused parameters
       */
      (VOID)(param);
      (VOID)(pkt_data);

      /* convert the timestamp to readable format */
      local_tv_sec = header->ts.tv_sec;
```

```
      //ltime=localtime(&local_tv_sec);
      //strftime( timestr, sizeof timestr, "%H:%M:%S", ltime);
          /* Print the packet */
      int i;
      printf("\n");
      printf("------------------------------------------------------------------
----------------------------------------------------\n\n");
    for (i=1; (i < header->caplen + 1 ) ; i++)
    {
        printf("%.2x ", pkt_data[i-1]);
        if ( (i % 16) == 0) printf("\n");
    }

      //printf("%s,%.6d len:%d\n", timestr, header->ts.tv_usec, header->len);
      int j=0,k=0, m=0, n=0, l=0, p=0;
      printf("\nMAC destination: ");
      for(j=0;j<6;j++){
         printf("%02X:",pkt_data[j]);
      }
      printf("\nMAC source: ");
      for(k=6;k<12;k++){
         printf("%02X: ",pkt_data[k]);
      }

      printf(" \n\n");
    unsigned short tipo = (pkt_data[12]*256)+pkt_data[13];
    printf("Tipo: %d   %02X %02X \n",tipo,pkt_data[12],pkt_data[13]);

    if(tipo==2054){
            printf("Target Protocol Address\n");
            //Tipo de hardware
            printf("\nHardware type: ");
            unsigned short type_Hardware = (pkt_data[14]*256) + pkt_data[15];
            if(type_Hardware==1)
                  printf("\n\tEthernet(1)");
            else if(type_Hardware==6)
                  printf("\n\tIEEE 802 Networks (6)");
            else if(type_Hardware==7)
                  printf("\n\tARC NET");
            else if(type_Hardware==15)
                  printf("\n\tFrame Relay (15)");
            else if(type_Hardware==16)
                  printf("\n\tATM (Asynchronous Transfer Mode)");
            else if(type_Hardware==17)
                  printf("\n\tHDLC");
            else if(type_Hardware==18)
                  printf("\n\tFibre Channel");
            else if(type_Hardware==19)
                  printf("\n\tATM (Asynchronous Transfer Mode)");
            else if(type_Hardware==20)
                  printf("\n\tSerial Line");
            else
                  printf("\n\tUnknown");
            printf("\n\n");
```

```
//Tipo de protocolo
printf("Protocol type: ");
unsigned short type_Protocol = (pkt_data[16]*256) + pkt_data[17];
if(type_Protocol==2048)
	printf("\n\tIPv4\n");
else if(type_Protocol==2054)
	printf("\n\tARP\n");
else
	printf("\n\tUnknown\n");

//Tamaño de hardware
printf("\nHardware size: ");
printf("%d", pkt_data[18]);

//Tamaño de protocolo
printf("\nProtocol size: ");
printf("%d", pkt_data[19]);

//Opcode
printf("\nOpcode: ");
unsigned short Opcode =(pkt_data[20] *256) + pkt_data[21];
if(Opcode==1)
	printf("\n\tARP REQUEST (1)\n");
else if(Opcode==2)
	printf("\n\tARP REPLY (2)\n");
else if(Opcode==3)
	printf("\n\tRARP REQUEST (3)\n");
else if(Opcode==4)
	printf("\n\tRARP REPLY (4)\n");
else
	printf("\n\tUnknown\n");

//MAC address del remitente
printf("\nSender MAC address: ");
for(m=22;m<28;m++){
	printf("%02X ",pkt_data[m]);
}

//IP address del remitente
printf("\nSender IP address: ");
for(n=28;n<31;n++){
	printf("%d ",pkt_data[n]);
}
printf("%d",pkt_data[31]);

//MAC address del objetivo
printf("\nTarget MAC address: ");
for(l=32;l<38;l++){
	printf("%02X ",pkt_data[l]);
}

printf("\nTarget IP address: ");
for(p=38; p<41;p++){
	printf("%d.",pkt_data[p]);
}
```

```
            printf("%d\n",pkt_data[41]);
      }

}
```

- IPTRAMA.H

```
/* 4 bytes IP address */
typedef struct ip_address{
      u_char byte1;
      u_char byte2;
      u_char byte3;
      u_char byte4;
}ip_address;

/* IPv4 header */
typedef struct ip_header{
      u_char ver_ihl; // Version (4 bits) + IP header length (4 bits)
      u_char tos; // Type of service
      u_short tlen; // Total length
      u_short identification; // Identification
      u_short flags_fo; // Flags (3 bits) + Fragment offset (13 bits)
      u_char ttl; // Time to live
      u_char proto; // Protocol
      u_short crc; // Header checksum
      ip_address saddr; // Source address
      ip_address daddr; // Destination address
      u_int op_pad; // Option + Padding
}ip_header;

void IP_Trama(u_char *param, const struct pcap_pkthdr *header, const u_char
*pkt_data)
{
      struct tm *ltime;
      char timestr[16];
      time_t local_tv_sec;

      /*unused parameters*/
      (VOID)(param);
      (VOID)(pkt_data);

      /* convert the timestamp to readable format */
      local_tv_sec = header->ts.tv_sec;
      //ltime=localtime(&local_tv_sec);
      //strftime( timestr, sizeof timestr, "%H:%M:%S", ltime);
      printf("\n");
      printf("------------------------------------------------------------------
-------------------------------------------------\n\n");

      int i;
    for (i=1; (i < header->caplen + 1 ) ; i++){
        printf("%.2x ", pkt_data[i-1]);
        if ( (i % 16) == 0) printf("\n");
    }
```

```
    printf("\nMAC destination: ");
      for(i=0;i<6;i++){
         printf("%02X:",pkt_data[i]);
      }
      printf("\nMAC source: ");
      for(i=6;i<12;i++){
         printf("%02X:",pkt_data[i]);
      }

      //printf("%s,%.6d len:%d\n", timestr, header->ts.tv_usec, header->len);

      printf(" \n\n");
    unsigned short tipo = (pkt_data[12]*256)+pkt_data[13];
    printf("Tipo: %d   %02X %02X \n",tipo,pkt_data[12],pkt_data[13]);
    if (tipo==2048){
            printf("Paquete IP..\n");
            ip_header *ih;
            u_int ip_len;
            /* retireve the position of the ip header */
            ih = (ip_header *) (pkt_data + 14); //length of ethernet header
            /* print ip addresses and udp ports */
            int j;
            for(j=5;j<9;j++){
                  printf("%d",(ih->ver_ihl>>j)&0x01);
            }
            printf(" .... = Version: ");
            printf("%d",(ih->ver_ihl>>4)&0x0f);

            char le[4];
            printf("\n.... ");
            for(j=0;j<4;j++){
                  le[j] = (ih->ver_ihl>>j)&0x01;
            }
            for(j=3;j>=0;j--)
                  printf("%d",(int)le[j]);

            u_short length = (u_short)((ih->ver_ihl)&0x0f)* (u_short)((ih-
>ver_ihl>>4)&0x0f);
            printf(" = Header Length : %d bytes ", length);
            printf("(%d)\n",(ih->ver_ihl)&0x0f);

            u_short lentotal = ((u_short)((ih->tlen)&0xff)*256) +
(u_short)((ih->tlen>>8)&0xff);
            printf("Longitud total: %d\n", lentotal);

            printf("Servicios Diferenciados: [precedence: ");
            for(j=5; j<8; j++)
                  printf("%d",(ih->tos>>j)&0x01);
            u_short CSI = (ih->tos)&0xe0;

            if(CSI==224)//111
                  printf(" (Network Control)]");
            else if(CSI==192)//110
                  printf(" (Internetwork control)]");
```

```
            else if(CSI==160)//101
                  printf(" (CRITIC/ECP)]");
            else if(CSI==128)//100
                  printf(" (Flash overrite)]");
            else if(CSI==96)//011
                  printf(" (Flash)]");
            else if(CSI==64)//010
                  printf(" (Immediate)]");
            else if(CSI==32)//001
                  printf(" (Priority)]");
            else if(CSI==0)//000
                  printf(" (Routine)]");
            else
                  printf(" (Unknown)]");

            printf(" [ECN: ");
            for(j=0; j<2; j++)
                  printf("%d",(ih->tos>>j)&0x01);
            u_short ECN = (ih->tos)&0x03;

            if(ECN==0)//111
                  printf(" (Sin capacidad ECN)]\n");
            else if(ECN==1)//001
                  printf(" (Capacidad de transporte ECN (0))]\n");
            else if(ECN==2)//010
                  printf(" (Capacidad de transporte ECN (1))]\n");
            else if(ECN==3)//011
                  printf(" (Congestion encontrada)]\n");
            else
                  printf(" (Unknown)]\n");

            printf("ID: %02X %02X", (ih->identification)&0xff, (ih-
>identification>>8)&0xff);

            printf("\n----Flags\nDon't Fragment: %d", (ih->flags_fo>>6)&0x01);
            (ih->flags_fo>>6)&0x01==1? puts(" Encendido"):puts(" Apagado");
            printf("More: %d", (ih->flags_fo>>5)&0x01);
            (ih->flags_fo>>5)&0x01==1? puts(" Encendido"):puts(" Apagado");
            u_short Foffset = ((u_short)((ih->flags_fo<<3)&0x1f)*256) +
(u_short)((ih->flags_fo>>8)&0xff);
            printf("Fragment offset: %02X %02X (%02X)", ((ih-
>flags_fo<<3))&0x1f, (ih->flags_fo>>8)&0xff, Foffset);

            printf("\nTTL: %02X (%d)", ih->ttl,ih->ttl);

            printf("\nProtocolo: %02X ", ih->proto);

            if(ih->proto==0)
                  printf("Reserved");
            else if(ih->proto==1)
                  printf("ICMP");
            else if(ih->proto==2)
                  printf("IGMP");
            else if(ih->proto==6)
                  printf("TCP");
```

```
            else if(ih->proto==17)
                  printf("UDP");
            else
                  printf("Other");

            printf("\nChecksum: %02X %02X", (ih->crc)&0xff, (ih->crc>>8)&0xff);

            printf("\nSource IP Address: %d.%d.%d.%d\nDestination IP Address:
%d.%d.%d.%d\n",ih->saddr.byte1,ih->saddr.byte2,ih->saddr.byte3,ih-
>saddr.byte4,ih->daddr.byte1,ih->daddr.byte2,ih->daddr.byte3,ih->daddr.byte4);
      }
      else
            printf("No es IP...\n");
}
```

- ICMPTRAMA.H

```
/* ICMP header*/
typedef struct icmp_header{
      u_char type; // ICMP type
      u_char code; // ICMP code
      u_short crc; // Checksum
}icmp_header;


void ICMP_Trama(u_char *param, const struct pcap_pkthdr *header, const u_char
*pkt_data)
{
      struct tm *ltime;
      char timestr[16];
      time_t local_tv_sec;

      /*unused parameters*/
      (VOID)(param);
      (VOID)(pkt_data);

      /* convert the timestamp to readable format */
      local_tv_sec = header->ts.tv_sec;
      //ltime=localtime(&local_tv_sec);
      //strftime( timestr, sizeof timestr, "%H:%M:%S", ltime);
      printf("\n");
      printf("------------------------------------------------------------------
--------------------------------------------------\n\n");

      int i;
    for (i=1; (i < header->caplen + 1 ) ; i++){
        printf("%.2x ", pkt_data[i-1]);
        if ( (i % 16) == 0) printf("\n");
    }

    printf("\nMAC destination: ");
      for(i=0;i<6;i++){
         printf("%02X:",pkt_data[i]);
      }
      printf("\nMAC source: ");
```

```
      for(i=6;i<12;i++){
         printf("%02X:",pkt_data[i]);
      }

      //printf("%s,%.6d len:%d\n", timestr, header->ts.tv_usec, header->len);

      printf(" \n\n");
    unsigned short tipo = (pkt_data[12]*256)+pkt_data[13];
    printf("Tipo: %d   %02X %02X \n",tipo,pkt_data[12],pkt_data[13]);
    if (tipo==2048){
            printf("Paquete IP..\n");
            ip_header *ih;
            u_int ip_len;
            /* retireve the position of the ip header */
            ih = (ip_header *) (pkt_data + 14); //length of ethernet header
            /* print ip addresses and udp ports */
            int j;
            for(j=5;j<9;j++){
                  printf("%d",(ih->ver_ihl>>j)&0x01);
            }
            printf(" .... = Version: ");
            printf("%d",(ih->ver_ihl>>4)&0x0f);

            char le[4];
            printf("\n.... ");
            for(j=0;j<4;j++){
                  le[j] = (ih->ver_ihl>>j)&0x01;
            }
            for(j=3;j>=0;j--)
                  printf("%d",(int)le[j]);

            u_short length = (u_short)((ih->ver_ihl)&0x0f)* (u_short)((ih-
>ver_ihl>>4)&0x0f);
            printf(" = Header Length : %d bytes ", length);
            printf("(%d)\n",(ih->ver_ihl)&0x0f);

            u_short lentotal = ((u_short)((ih->tlen)&0xff)*256) +
(u_short)((ih->tlen>>8)&0xff);
            printf("Longitud total: %d\n", lentotal);

            printf("Servicios Diferenciados: [precedence: ");
            for(j=5; j<8; j++)
                  printf("%d",(ih->tos>>j)&0x01);
            u_short CSI = (ih->tos)&0xe0;

            if(CSI==224)//111
                  printf(" (Network Control)]");
            else if(CSI==192)//110
                  printf(" (Internetwork control)]");
            else if(CSI==160)//101
                  printf(" (CRITIC/ECP)]");
            else if(CSI==128)//100
                  printf(" (Flash overrite)]");
            else if(CSI==96)//011
                  printf(" (Flash)]");
```

```
            else if(CSI==64)//010
                  printf(" (Immediate)]");
            else if(CSI==32)//001
                  printf(" (Priority)]");
            else if(CSI==0)//000
                  printf(" (Routine)]");
            else
                  printf(" (Unknown)]");

            printf(" [ECN: ");
            for(j=0; j<2; j++)
                  printf("%d",(ih->tos>>j)&0x01);
            u_short ECN = (ih->tos)&0x03;

            if(ECN==0)//111
                  printf(" (Sin capacidad ECN)]\n");
            else if(ECN==1)//001
                  printf(" (Capacidad de transporte ECN (0))]\n");
            else if(ECN==2)//010
                  printf(" (Capacidad de transporte ECN (1))]\n");
            else if(ECN==3)//011
                  printf(" (Congestion encontrada)]\n");
            else
                  printf(" (Unknown)]\n");

            printf("ID: %02X %02X", (ih->identification)&0xff, (ih-
>identification>>8)&0xff);

            printf("\n----Flags\nDon't Fragment: %d", (ih->flags_fo>>6)&0x01);
            (ih->flags_fo>>6)&0x01==1? puts(" Encendido"):puts(" Apagado");
            printf("More: %d", (ih->flags_fo>>5)&0x01);
            (ih->flags_fo>>5)&0x01==1? puts(" Encendido"):puts(" Apagado");
            u_short Foffset = ((u_short)((ih->flags_fo<<3)&0x1f)*256) +
(u_short)((ih->flags_fo>>8)&0xff);
            printf("Fragment offset: %02X %02X (%02X)", ((ih-
>flags_fo<<3))&0x1f, (ih->flags_fo>>8)&0xff, Foffset);

            printf("\nTTL: %02X (%d)", ih->ttl,ih->ttl);

            printf("\nProtocolo: %02X ", ih->proto);

            if(ih->proto==0)
                  printf("Reserved");
            else if(ih->proto==1)
                  printf("ICMP");
            else if(ih->proto==2)
                  printf("IGMP");
            else if(ih->proto==6)
                  printf("TCP");
            else if(ih->proto==17)
                  printf("UDP");
            else
                  printf("Other");

            printf("\nChecksum: %02X %02X", (ih->crc)&0xff, (ih->crc>>8)&0xff);
```

```
            printf("\nSource IP Address: %d.%d.%d.%d\nDestination IP Address:
%d.%d.%d.%d\n",ih->saddr.byte1,ih->saddr.byte2,ih->saddr.byte3,ih-
>saddr.byte4,ih->daddr.byte1,ih->daddr.byte2,ih->daddr.byte3,ih->daddr.byte4);

            if(ih->proto==0)
                  printf("Protocol Reserved");
            else if(ih->proto==1){
                  printf("Protocol ICMP");
                  icmp_header *icmp;
                  u_char ihl = ((ih->ver_ihl)&0x0f)*4;
                  icmp = (icmp_header *) (pkt_data + 14+(ihl));
                  printf("\n\tTipo: %d\n",icmp->type);
                  printf("\tCodigo:%d\n", icmp->code);

                  printf("\tDescription: ");
                  if(icmp->type==0){
                        printf("Echo Reply ---> ");
                        if(icmp->code==0)
                              printf("Echo Reply");
                  }
                  else if(icmp->type==3){
                        printf("\tDestination unreachable ---> ");
                        if(icmp->code==0)
                              printf("Destination network unreachable");
                        else if(icmp->code==1)
                              printf("Destination host unreachable");
                        else if(icmp->code==2)
                              printf("Destination protocol unreachable");
                        else if(icmp->code==3)
                              printf("Destination port unreachable");
                        else if(icmp->code==4)
                              printf("Fragmetation needed and DF flag set");
                        else
                              printf("Source route failed");
                  }
                  else if(icmp->type==5){
                        printf("\tRedirect Message ---> ");//4
                        if(icmp->code==0)
                              printf("Redirect datagram for the Network");
                        else if(icmp->code==1)
                              printf("Redirect datagram for the host");
                        else if(icmp->code==2)
                              printf("Redirect datagram for the Type of
Service and Network");
                        else
                              printf("Redirect datagram for the Service and
Host");
                  }
                  else if(icmp->type==8){
                        printf("\tEcho Request ---> ");
                        if(icmp->code==0)
                              printf("Echo request");
                  }
                  else if(icmp->type==9){
```

```
                        printf("\tRouter Advertisement ---> ");
                        if(icmp->code==0)
                              printf("Use to discover the addresses of
operational routers");
                  }
                  else if(icmp->type==10){
                        printf("\tRouter Solicitation ---> ");
                        if(icmp->code==0)
                              printf("Use to discover the addresses of
operational routers");
                  }
                  else if(icmp->type==11){
                        printf("\tTime Exceeded ---> ");//2
                        if(icmp->code==0)
                              printf("Time to live exceeded in transit");
                        else
                              printf("Fragment reassembly time exceeded");
                  }
                  else if(icmp->type==12){
                        printf("\tParameter Problem ---> ");//3
                        if(icmp->code==0)
                              printf("Pointer indicates error");
                        else if(icmp->code==1)
                              printf("Missing required option");
                        else
                              printf("Bad length");
                  }
                  else if(icmp->type==13){
                        printf("\tTimestamp ---> ");
                        if(icmp->code==0)
                              printf("Used for time synchronization");
                  }
                  else if(icmp->type==14){
                        printf("\tTimestamp Reply ---> ");
                        if(icmp->code==0)
                              printf("Reply to Timestamp message");
                  }
                  else
                        printf("\tUnknown");

                  printf("\n\tChecksum: %02X %02X\n", (icmp->crc)&0xff, (icmp-
>crc>>8)&0xff);
            }
            else
                  printf("Other Protocol\n");
      }
      else
            printf("No es IP...\n");
}
```

- IGMP.H

```
      /* convert the timestamp to readable format */
      local_tv_sec = header->ts.tv_sec;
```

```
      //ltime=localtime(&local_tv_sec);
      //strftime( timestr, sizeof timestr, "%H:%M:%S", ltime);
      printf("\n");
      printf("------------------------------------------------------------------
-------------------------------------------------\n\n");

      int i;
    for (i=1; (i < header->caplen + 1 ) ; i++){
        printf("%.2x ", pkt_data[i-1]);
        if ( (i % 16) == 0) printf("\n");
    }

    printf("\nMAC destination: ");
      for(i=0;i<6;i++){
         printf("%02X:",pkt_data[i]);
      }
      printf("\nMAC source: ");
      for(i=6;i<12;i++){
         printf("%02X:",pkt_data[i]);
      }

      //printf("%s,%.6d len:%d\n", timestr, header->ts.tv_usec, header->len);

      printf(" \n\n");
    unsigned short tipo = (pkt_data[12]*256)+pkt_data[13];
    printf("Tipo: %d   %02X %02X \n",tipo,pkt_data[12],pkt_data[13]);
    if (tipo==2048){
            printf("Paquete IP..\n");
            ip_header *ih;
            u_int ip_len;
            /* retireve the position of the ip header */
            ih = (ip_header *) (pkt_data + 14); //length of ethernet header
            /* print ip addresses and udp ports */
            int j;
            for(j=5;j<9;j++){
                  printf("%d",(ih->ver_ihl>>j)&0x01);
            }
            printf(" .... = Version: ");
            printf("%d",(ih->ver_ihl>>4)&0x0f);

            char le[4];
            printf("\n.... ");
            for(j=0;j<4;j++){
                  le[j] = (ih->ver_ihl>>j)&0x01;
            }
            for(j=3;j>=0;j--)
                  printf("%d",(int)le[j]);

            u_short length = (u_short)((ih->ver_ihl)&0x0f)* (u_short)((ih-
>ver_ihl>>4)&0x0f);
            printf(" = Header Length : %d bytes ", length);
            printf("(%d)\n",(ih->ver_ihl)&0x0f);

            u_short lentotal = ((u_short)((ih->tlen)&0xff)*256) +
(u_short)((ih->tlen>>8)&0xff);
```

```
            printf("Longitud total: %d\n", lentotal);

            printf("Servicios Diferenciados: [precedence: ");
            for(j=5; j<8; j++)
                  printf("%d",(ih->tos>>j)&0x01);
            u_short CSI = (ih->tos)&0xe0;

            if(CSI==224)//111
                  printf(" (Network Control)]");
            else if(CSI==192)//110
                  printf(" (Internetwork control)]");
            else if(CSI==160)//101
                  printf(" (CRITIC/ECP)]");
            else if(CSI==128)//100
                  printf(" (Flash overrite)]");
            else if(CSI==96)//011
                  printf(" (Flash)]");
            else if(CSI==64)//010
                  printf(" (Immediate)]");
            else if(CSI==32)//001
                  printf(" (Priority)]");
            else if(CSI==0)//000
                  printf(" (Routine)]");
            else
                  printf(" (Unknown)]");

            printf(" [ECN: ");
            for(j=0; j<2; j++)
                  printf("%d",(ih->tos>>j)&0x01);
            u_short ECN = (ih->tos)&0x03;

            if(ECN==0)//111
                  printf(" (Sin capacidad ECN)]\n");
            else if(ECN==1)//001
                  printf(" (Capacidad de transporte ECN (0))]\n");
            else if(ECN==2)//010
                  printf(" (Capacidad de transporte ECN (1))]\n");
            else if(ECN==3)//011
                  printf(" (Congestion encontrada)]\n");
            else
                  printf(" (Unknown)]\n");

            printf("ID: %02X %02X", (ih->identification)&0xff, (ih-
>identification>>8)&0xff);

            printf("\n----Flags\nDon't Fragment: %d", (ih->flags_fo>>6)&0x01);
            (ih->flags_fo>>6)&0x01==1? puts(" Encendido"):puts(" Apagado");
            printf("More: %d", (ih->flags_fo>>5)&0x01);
            (ih->flags_fo>>5)&0x01==1? puts(" Encendido"):puts(" Apagado");
            u_short Foffset = ((u_short)((ih->flags_fo<<3)&0x1f)*256) +
(u_short)((ih->flags_fo>>8)&0xff);
            printf("Fragment offset: %02X %02X (%02X)", ((ih-
>flags_fo<<3))&0x1f, (ih->flags_fo>>8)&0xff, Foffset);

            printf("\nTTL: %02X (%d)", ih->ttl,ih->ttl);
```

```
            printf("\nProtocolo: %02X ", ih->proto);

            if(ih->proto==0)
                  printf("Reserved");
            else if(ih->proto==1)
                  printf("ICMP");
            else if(ih->proto==2)
                  printf("IGMP");
            else if(ih->proto==6)
                  printf("TCP");
            else if(ih->proto==17)
                  printf("UDP");
            else
                  printf("Other");

            printf("\nChecksum: %02X %02X", (ih->crc)&0xff, (ih->crc>>8)&0xff);

            printf("\nSource IP Address: %d.%d.%d.%d\nDestination IP Address:
%d.%d.%d.%d\n",ih->saddr.byte1,ih->saddr.byte2,ih->saddr.byte3,ih-
>saddr.byte4,ih->daddr.byte1,ih->daddr.byte2,ih->daddr.byte3,ih->daddr.byte4);

            if(ih->proto==0)
                  printf("Protocol Reserved");
            else if(ih->proto==2){
                  igmp_header *igmp;
                  u_char ihl = ((ih->ver_ihl)&0x0f)*4;
                  igmp = (igmp_header *) (pkt_data + 14+(ihl));
                  printf("Protocol IGMP");
                  printf("\n\tTipo: %02X", igmp->type);
                  printf("\n\tReserved: %02X", igmp->rsv1);
                  printf("\n\tChecksum: %02X", igmp->crc);
                  printf("\n\tReserved: %02X", igmp->rsv2);
                  printf("\n\tGroup: %02X", igmp->ngr);
            }
            else
                  printf("\nOther Protocol");
      }
      else
            printf("No es IP...\n");
}
```

- TCPTRAMA.H

```
/* TCP header*/
typedef struct tcp_header{
      u_short sport; // Source port
      u_short dport; // Destination port
      u_int sec_num; // secuence number
      u_int ack_num; // ack number
      u_char d_offset_rsv; // 4bit data offset +4bit reserved
      u_char flags; // TCP flags
      u_short window; // window
      u_short crc; // Checksum
      u_short upointer; // urgent pointer
```

```
}tcp_header;

void TCP_Trama(u_char *param, const struct pcap_pkthdr *header, const u_char
*pkt_data){
      struct tm *ltime;
      char timestr[16];
      time_t local_tv_sec;

      /*unused parameters*/
      (VOID)(param);
      (VOID)(pkt_data);

      /* convert the timestamp to readable format */
      local_tv_sec = header->ts.tv_sec;
      //ltime=localtime(&local_tv_sec);
      //strftime( timestr, sizeof timestr, "%H:%M:%S", ltime);

      printf("\n");
      printf("------------------------------------------------------------------
----------------------------------------------------\n\n");

      int i;
    for (i=1; (i < header->caplen + 1 ) ; i++){
        printf("%.2x ", pkt_data[i-1]);
        if ( (i % 16) == 0) printf("\n");
    }

    printf("\nMAC destination: ");
      for(i=0;i<6;i++){
         printf("%02X:",pkt_data[i]);
      }
      printf("\nMAC source: ");
      for(i=6;i<12;i++){
         printf("%02X:",pkt_data[i]);
      }

      //printf("%s,%.6d len:%d\n", timestr, header->ts.tv_usec, header->len);

      printf(" \n\n");
    unsigned short tipo = (pkt_data[12]*256)+pkt_data[13];
    printf("Tipo: %d   %02X %02X \n",tipo,pkt_data[12],pkt_data[13]);
    if (tipo==2048){
            printf("Paquete IP..\n");
            ip_header *ih;
            u_int ip_len;
            /* retireve the position of the ip header */
            ih = (ip_header *) (pkt_data + 14); //length of ethernet header
            /* print ip addresses and udp ports */
            int j;
            for(j=5;j<9;j++){
                  printf("%d",(ih->ver_ihl>>j)&0x01);
            }
            printf(" .... = Version: ");
            printf("%d",(ih->ver_ihl>>4)&0x0f);
```

```
            char le[4];
            printf("\n.... ");
            for(j=0;j<4;j++){
                  le[j] = (ih->ver_ihl>>j)&0x01;
            }
            for(j=3;j>=0;j--)
                  printf("%d",(int)le[j]);

            u_short length = (u_short)((ih->ver_ihl)&0x0f)* (u_short)((ih-
>ver_ihl>>4)&0x0f);
            printf(" = Header Length : %d bytes ", length);
            printf("(%d)\n",(ih->ver_ihl)&0x0f);

            u_short lentotal = ((u_short)((ih->tlen)&0xff)*256) +
(u_short)((ih->tlen>>8)&0xff);
            printf("Longitud total: %d\n", lentotal);

            printf("Servicios Diferenciados: [precedence: ");
            for(j=5; j<8; j++)
                  printf("%d",(ih->tos>>j)&0x01);
            u_short CSI = (ih->tos)&0xe0;

            if(CSI==224)//111
                  printf(" (Network Control)]");
            else if(CSI==192)//110
                  printf(" (Internetwork control)]");
            else if(CSI==160)//101
                  printf(" (CRITIC/ECP)]");
            else if(CSI==128)//100
                  printf(" (Flash overrite)]");
            else if(CSI==96)//011
                  printf(" (Flash)]");
            else if(CSI==64)//010
                  printf(" (Immediate)]");
            else if(CSI==32)//001
                  printf(" (Priority)]");
            else if(CSI==0)//000
                  printf(" (Routine)]");
            else
                  printf(" (Unknown)]");

            printf(" [ECN: ");
            for(j=0; j<2; j++)
                  printf("%d",(ih->tos>>j)&0x01);
            u_short ECN = (ih->tos)&0x03;

            if(ECN==0)//111
                  printf(" (Sin capacidad ECN)]\n");
            else if(ECN==1)//001
                  printf(" (Capacidad de transporte ECN (0))]\n");
            else if(ECN==2)//010
                  printf(" (Capacidad de transporte ECN (1))]\n");
            else if(ECN==3)//011
                  printf(" (Congestion encontrada)]\n");
            else
```

```
                  printf(" (Unknown)]\n");

            printf("ID: %02X %02X", (ih->identification)&0xff, (ih-
>identification>>8)&0xff);

            printf("\n----Flags\nDon't Fragment: %d", (ih->flags_fo>>6)&0x01);
            (ih->flags_fo>>6)&0x01==1? puts(" Encendido"):puts(" Apagado");
            printf("More: %d", (ih->flags_fo>>5)&0x01);
            (ih->flags_fo>>5)&0x01==1? puts(" Encendido"):puts(" Apagado");
            u_short Foffset = ((u_short)((ih->flags_fo<<3)&0x1f)*256) +
(u_short)((ih->flags_fo>>8)&0xff);
            printf("Fragment offset: %02X %02X (%02X)", ((ih-
>flags_fo<<3))&0x1f, (ih->flags_fo>>8)&0xff, Foffset);

            printf("\nTTL: %02X (%d)", ih->ttl,ih->ttl);

            printf("\nProtocolo: %02X ", ih->proto);

            if(ih->proto==0)
                  printf("Reserved");
            else if(ih->proto==6){
                  printf("Protocol TCP");
                  tcp_header *tcp;
                  u_char ihl = ((ih->ver_ihl)&0x0f)*4;
                  tcp = (tcp_header *) (pkt_data + 14+(ihl));
                  printf("\n\tSource port: %02X %02X\n",(tcp->sport)&0xff,
(tcp->sport>>8)&0xff);
                  printf("\tDestination port: %02X %02X\n",(tcp->dport)&0xff,
(tcp->dport>>8)&0xff);
                  printf("\tSecuense number: %02X %02X %02X %02X\n",(tcp-
>sec_num)&0xff, (tcp->sec_num>>8)&0xff,(tcp->sec_num>>16)&0xff,(tcp-
>sec_num>>24)&0xff);
                  printf("\tAck number: %02X %02X %02X %02X\n",(tcp-
>ack_num)&0xff, (tcp->ack_num>>8)&0xff,(tcp->ack_num>>16)&0xff,(tcp-
>ack_num>>24)&0xff);
                  printf("\tChecksum: %02X %02X\n", (tcp->crc)&0xff, (tcp-
>crc>>8)&0xff);
                  printf("\tData Offset: %d\n", (tcp->d_offset_rsv>>4)&0x0f);
                  printf("\tFlags: ");
                  for(j=0; j<=7; j++)
                        printf("%d", (tcp->flags>>j)&0x01);
                  printf("\t");
                  if((tcp->flags)&0x01==1)
                        printf("CWR ");
                  if((tcp->flags>>1)&0x01==1)
                        printf("ECE ");
                  if((tcp->flags>>2)&0x01==1)
                        printf("URG ");
                  if((tcp->flags>>3)&0x01==1)
                        printf("ACK ");
                  if((tcp->flags>>4)&0x01==1)
                        printf("PSH ");
                  if((tcp->flags>>5)&0x01==1)
                        printf("RST ");
                  if((tcp->flags>>6)&0x01==1)
```

```
                        printf("SYN ");
                  if((tcp->flags>>7)&0x01==1)
                        printf("FIN ");
                  printf("\n\tUrgent Pointer: %02X  %02X\n",(tcp-
>upointer)&0xff, (tcp->upointer>>8)&0xff);
            }
            else
                  printf("Other Protocol");
      }
      else
            printf("No es IP...\n");
}
```

- UDPTRAMA.H

```
/* UDP header*/
typedef struct udp_header{
      u_short sport; //Source port
      u_short dport; //Destination port
      u_short len; //length
      u_short crc; //Checksum
}udp_header;

void UDP_Trama(u_char *param, const struct pcap_pkthdr *header, const u_char
*pkt_data)
{
      struct tm *ltime;
      char timestr[16];
      time_t local_tv_sec;

      /*unused parameters*/
      (VOID)(param);
      (VOID)(pkt_data);

      /* convert the timestamp to readable format */
      local_tv_sec = header->ts.tv_sec;
      //ltime=localtime(&local_tv_sec);
      //strftime( timestr, sizeof timestr, "%H:%M:%S", ltime);

      printf("\n");
      printf("------------------------------------------------------------------
--------------------------------------------------\n\n");
      int i;
    for (i=1; (i < header->caplen + 1 ) ; i++){
        printf("%.2x ", pkt_data[i-1]);
        if ( (i % 16) == 0) printf("\n");
    }

    printf("\nMAC destination: ");
      for(i=0;i<6;i++){
         printf("%02X:",pkt_data[i]);
      }
      printf("\nMAC source: ");
      for(i=6;i<12;i++){
```

```
         printf("%02X:",pkt_data[i]);
      }

      //printf("%s,%.6d len:%d\n", timestr, header->ts.tv_usec, header->len);

      printf(" \n\n");
    unsigned short tipo = (pkt_data[12]*256)+pkt_data[13];
    printf("Tipo: %d   %02X %02X \n",tipo,pkt_data[12],pkt_data[13]);
    if (tipo==2048){
            printf("Paquete IP..\n");
            ip_header *ih;
            u_int ip_len;
            /* retireve the position of the ip header */
            ih = (ip_header *) (pkt_data + 14); //length of ethernet header
            /* print ip addresses and udp ports */
            int j;
            for(j=5;j<9;j++){
                  printf("%d",(ih->ver_ihl>>j)&0x01);
            }
            printf(" .... = Version: ");
            printf("%d",(ih->ver_ihl>>4)&0x0f);

            char le[4];
            printf("\n.... ");
            for(j=0;j<4;j++){
                  le[j] = (ih->ver_ihl>>j)&0x01;
            }
            for(j=3;j>=0;j--)
                  printf("%d",(int)le[j]);

            u_short length = (u_short)((ih->ver_ihl)&0x0f)* (u_short)((ih-
>ver_ihl>>4)&0x0f);
            printf(" = Header Length : %d bytes ", length);
            printf("(%d)\n",(ih->ver_ihl)&0x0f);

            u_short lentotal = ((u_short)((ih->tlen)&0xff)*256) +
(u_short)((ih->tlen>>8)&0xff);
            printf("Longitud total: %d\n", lentotal);

            printf("Servicios Diferenciados: [precedence: ");
            for(j=5; j<8; j++)
                  printf("%d",(ih->tos>>j)&0x01);
            u_short CSI = (ih->tos)&0xe0;

            if(CSI==224)//111
                  printf(" (Network Control)]");
            else if(CSI==192)//110
                  printf(" (Internetwork control)]");
            else if(CSI==160)//101
                  printf(" (CRITIC/ECP)]");
            else if(CSI==128)//100
                  printf(" (Flash overrite)]");
            else if(CSI==96)//011
                  printf(" (Flash)]");
            else if(CSI==64)//010
```

```
                printf(" (Immediate)]");
            else if(CSI==32)//001
                printf(" (Priority)]");
            else if(CSI==0)//000
                printf(" (Routine)]");
            else
                printf(" (Unknown)]");

            printf(" [ECN: ");
            for(j=0; j<2; j++)
                printf("%d",(ih->tos>>j)&0x01);
            u_short ECN = (ih->tos)&0x03;

            if(ECN==0)//111
                printf(" (Sin capacidad ECN)]\n");
            else if(ECN==1)//001
                printf(" (Capacidad de transporte ECN (0))]\n");
            else if(ECN==2)//010
                printf(" (Capacidad de transporte ECN (1))]\n");
            else if(ECN==3)//011
                printf(" (Congestion encontrada)]\n");
            else
                printf(" (Unknown)]\n");

            printf("ID: %02X %02X", (ih->identification)&0xff, (ih-
>identification>>8)&0xff);

            printf("\n----Flags\nDon't Fragment: %d", (ih->flags_fo>>6)&0x01);
            (ih->flags_fo>>6)&0x01==1? puts(" Encendido"):puts(" Apagado");
            printf("More: %d", (ih->flags_fo>>5)&0x01);
            (ih->flags_fo>>5)&0x01==1? puts(" Encendido"):puts(" Apagado");
            u_short Foffset = ((u_short)((ih->flags_fo<<3)&0x1f)*256) +
(u_short)((ih->flags_fo>>8)&0xff);
            printf("Fragment offset: %02X %02X (%02X)", ((ih-
>flags_fo<<3))&0x1f, (ih->flags_fo>>8)&0xff, Foffset);

            printf("\nTTL: %02X (%d)", ih->ttl,ih->ttl);

            printf("\nProtocolo: %02X ", ih->proto);

            if(ih->proto==0)
                printf("Reserved");
            else if(ih->proto==1)
                printf("ICMP");
            else if(ih->proto==2)
                printf("IGMP");
            else if(ih->proto==6)
                printf("TCP");
            else if(ih->proto==17)
                printf("UDP");
            else
                printf("Other");

            printf("\nChecksum: %02X %02X", (ih->crc)&0xff, (ih->crc>>8)&0xff);
```

```
            printf("\nSource IP Address: %d.%d.%d.%d\nDestination IP Address:
%d.%d.%d.%d\n",ih->saddr.byte1,ih->saddr.byte2,ih->saddr.byte3,ih-
>saddr.byte4,ih->daddr.byte1,ih->daddr.byte2,ih->daddr.byte3,ih->daddr.byte4);

            if(ih->proto==0)
                  printf("Protocol Reserved");
            else if(ih->proto==17){
                  printf("Protocol UDP");
                  udp_header *udp;
                  u_char ihl = ((ih->ver_ihl)&0x0f)*4;
                  udp = (udp_header *) (pkt_data + 14+(ihl));
                  printf("\n\tSource port: %02X %02X\n",(udp->sport)&0xff,
(udp->sport>>8)&0xff);
                  printf("\tSecuense number: %02X %02X\n",(udp->dport)&0xff,
(udp->dport>>8)&0xff);
                  printf("\tLength: %02X %02X\n", (udp->len)&0xff, (udp-
>len>>8)&0xff);
                  printf("\tChecksum: %02X %02X\n", (udp->crc)&0xff, (udp-
>crc>>8)&0xff);
            }
            else
                  printf("Other Protocol");
      }
      else
            printf("No es IP...\n");
}
```