
ESCOM - IPN

Practica 1:

Captura de Tramas e Instalación de Jnetpcap

REDES DE CÓMPUTADORAS 2CM10

Oscar Andrés Rosas Hernandez Arturo Rivas Rojas

Junio 2018

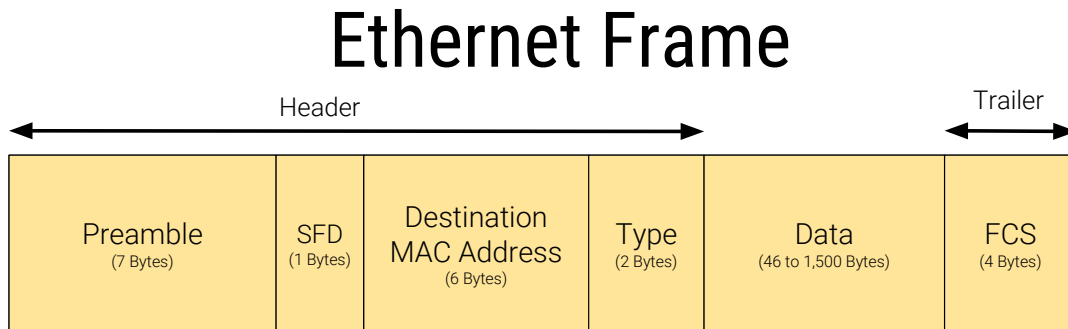
Índice

1. Protocolo Ethernet	2
1.1. Frame	2
1.2. Explicación	2
2. Dirección MAC	4
3. Practica en Si	5
3.1. Instalación	5
3.2. Evidencias	7
4. Conclusiones	11

1. Protocolo Ethernet

1.1. Frame

Si el valor que leo en tamaño o tipo es menor a 1500 implica que sigue el protocolo LLC y el tamaño del mismo.



1.2. Explicación

Los campos que componen una trama ethernet son los siguientes:

- **Preámbulo (Preamble)**

Campo con una secuencia de bits utilizada para sincronizar y estabilizar el medio físico antes de iniciar la transmisión. Es una secuencia de unos y ceros, conocida que permite a los nodos saber que está llegando un nuevo frame.

El patrón es el siguiente:

10101010 10101010 10101010 10101010 10101010 10101010 10101010

Tamaño: 7 Bytes

- **SFD (Start Frame Delimiter)**

Delimitador de inicio de trama. Campo que contiene la secuencia 10101011.

Indica el inicio de una trama de datos.

Tamaño: 1 Byte

■ Dirección de Destino (Destination Address)

Campo que contiene la dirección MAC a la que se envía la trama.

El bit más a la izquierda del campo indica cuando la dirección es individual (indicado por un 0) o un grupo de direcciones (indicado por un 1).

El segundo bit desde la izquierda indica cuando la dirección destino es globalmente administrada (indicado por un 0).

La capa de enlace de datos del remitente añade la dirección de destino a la trama. La capa de enlace de datos del destinatario examina la dirección de destino para identificar los mensajes a recibir.

Tamaño: 6 Bytes

■ Dirección de Origen (Source Address)

Campo que contiene la dirección MAC del dispositivo que envía la trama.

La dirección de origen es siempre una dirección individual y el bit más a la izquierda es siempre 0.

Con ella el receptor conoce a quien debe dirigir las respuestas del mensaje.

Tamaño: 6 Bytes

■ Tipo de Protocolo o Longitud

Indica el tipo de protocolo de capa superior.

Este campo es el que distingue a las tramas IEEE 802.3 de las tramas Ethernet.

Valores para este campo iguales o menores de x05DC (1500 en decimal) indican que es una trama IEEE 802.3 y el valor representa la longitud del campo de datos.

Valores para este campo iguales o mayores de x0600 indican que es una trama Ethernet y el valor representa el tipo de protocolo.

Tamaño: 2 Bytes

- **Datos and Pad(Payload)**

Contiene los datos a transferir entre origen y destino. Si este campo fuera menor de 46 bytes se añade un campo de “relleno”, es decir pad para mantener el tamaño mínimo de paquete.

Este campo contiene los datos transferidos desde el origen hasta el destino. El tamaño máximo este campo es de 1500 bytes. Si el tamaño de este campo es menor a 46 bytes, entonces es necesario el uso del campo siguiente (Pad) para añadir bytes hasta que el tamaño de la trama alcance el valor mínimo.

Inicia en el byte 15 y se tomará hasta la longitud que se obtiene al multiplicar el último número a la derecha del byte 15 por 4. Generalmente son 20 posiciones, es decir, del byte 15 al byte 34.

Tamaño: 46 a 1,500 Bytes

- **FCS (Frame Check Sequence)**

Secuencia de verificación de trama.

Campo que contiene un valor de para control de errores, CRC (Cyclical Redundancy Check). La verificación de redundancia cíclica (CRC), consiste en un valor calculado por el emisor que resume todos los datos de la trama. El receptor calcula nuevamente el valor y, si coincide con el de la trama, entiende que la trama se ha

El campo FCS es generado ó calculado sobre los campos dirección de destino, la dirección de origen, el tipo/longitud y datos.

Tamaño: 4 Bytes

2. Dirección MAC

La dirección MAC es un identificador único de 48 bits para identificar la totalidad de dispositivos de red como por ejemplo tarjetas de red Ethernet, tarjetas de red wifi o inalámbricas, etc. Las direcciones MAC son identificadores únicos a nivel mundial para cada dispositivo y por lo tanto es imposible encontrar 2 dispositivos de red que tengan la misma dirección MAC. La entidad que se encarga de definir como se deben definir las direcciones MAC, es la IEEE. La dirección MAC consta de 48 bits y viene expresada en 12 dígitos hexadecimales.

Los primeros 24 bits, o 6 dígitos hexadecimales, de la dirección MAC contienen un código de identificación del fabricante o vendedor OUI.

Los últimos 24 bits, o 6 dígitos hexadecimales, están administrados por cada fabricante y presentan, por lo general el número de serie de la tarjeta

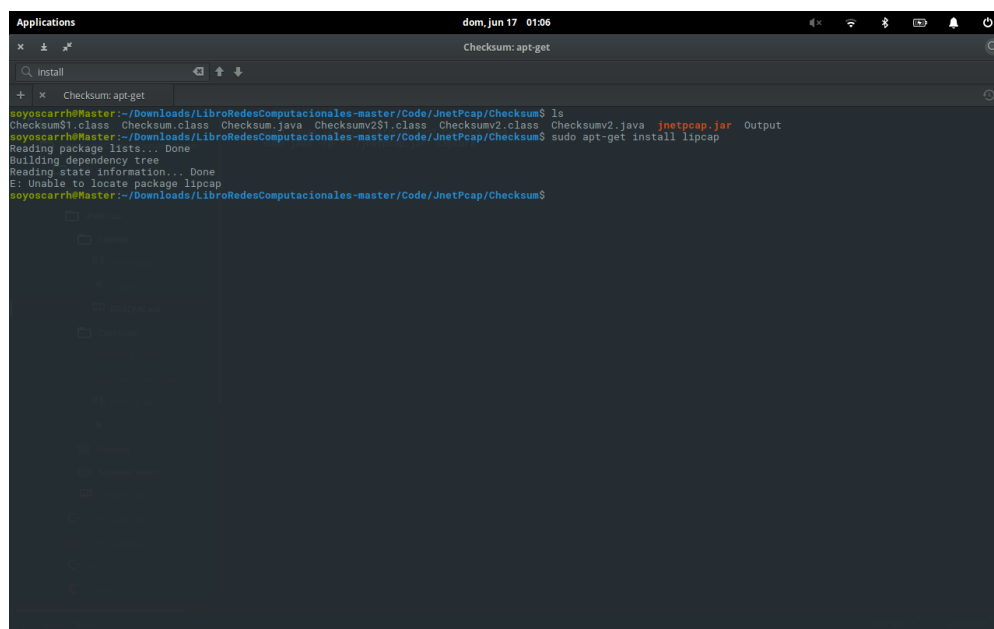
3. Practica en Si

3.1. Instalación

Para el funcionamiento de esta practica tenemos que tener el .jar de jnetpcap, pero también tenemos que tener en nuestro sistema operativo el archivo .so, generalmente en “urs/lib”.

También tenemos que descargar libpcap o winpcap dependiendo del sistema operativo.

A continuación mostramos exactamente lo que necesitamos:



```
Applications
dom, jun 17 01:06
Checksum: apt-get

+ x Checksum: apt-get
soyoscarrheMaster:~/Downloads/LibroRedesComputacionales-master/Code/JnetPcap/Checksum$ ls
Checksum1.class Checksum.class Checksum.java Checksumv2$1.class Checksumv2.class Checksumv2.java jnetpcap.jar Output
soyoscarrheMaster:~/Downloads/LibroRedesComputacionales-master/Code/JnetPcap/Checksum$ sudo apt-get install libpcap
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package libpcap
soyoscarrheMaster:~/Downloads/LibroRedesComputacionales-master/Code/JnetPcap/Checksum$
```

Figura 1: Insalación

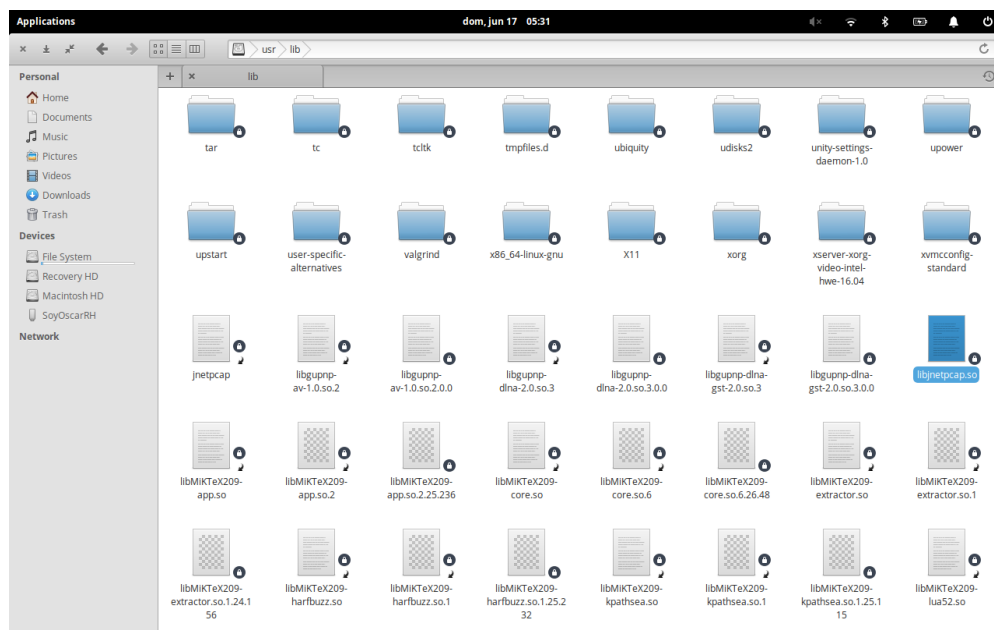


Figura 2: Instalación

3.2. Evidencias

Primeramente el código:

De acuerdo con el formato básico de una trama Ethernet que fue explicado en el marco teórico, la solución esta dada por simplemente recorrer el arreglo de bytes que estamos recibiendo.

```

1 import java.util.ArrayList;
2 import java.util.Date;
3 import java.util.List;
4 import java.io.*;
5
6 import org.jnetpcap.Pcap;
7 import org.jnetpcap.PcapIf;
8 import org.jnetpcap.packet.PcapPacket;
9 import org.jnetpcap.packet.PcapPacketHandler;
10 import org.jnetpcap.PcapBpfProgram;
11 import org.jnetpcap.protocol.lan.Ethernet;
12 import org.jnetpcap.protocol.tcpip.*;
13 import org.jnetpcap.protocol.network.*;
14 import org.jnetpcap.nio.JBuffer;
15 import org.jnetpcap.packet.Payload;
16 import org.jnetpcap.protocol.network.Arp;
17 import org.jnetpcap.protocol.lan.IEEE802dot2;
18 import org.jnetpcap.protocol.lan.IEEE802dot3;
19
20
21 public class ShowMacAddress {
22
23     /**
24     * Main startup method
25     *
26     * @param args
27     *         ignored
28     */
29     private static String asString(final byte[] mac) {
30         final StringBuilder buf = new StringBuilder();
31         for (byte b : mac) {
32             if (buf.length() != 0) {
33                 buf.append(':');
34             }
35             if (b >= 0 && b < 16) {
36                 buf.append('0');
37             }
38             buf.append(Integer.toHexString((b < 0) ? b + 256 : b).toUpperCase());
39         }
40     }
41     return buf.toString();
42 }
43
44 public static void main(String[] args) {
45     List<PcapIf> alldevs = new ArrayList<PcapIf>(); // Will be filled with NICs
46     StringBuilder errbuf = new StringBuilder(); // For any error msgs
47
48     /* *****
49     * First get a list of devices on this system
50     * ***** */
51     int r = Pcap.findAllDevs(alldevs, errbuf);
52     if (r == Pcap.NOT_OK || alldevs.isEmpty()) {
53         System.err.printf("Can't read list of devices, error is %s", errbuf
54             .toString());
55         return;
56     }
57
58     System.out.println("Network devices found:");
59
60     int i = 0;
61     try {
62         for (PcapIf device : alldevs) {
63             String description =
64                 (device.getDescription() != null) ? device.getDescription()
65                 : "No description available";
66             final byte[] mac = device.getHardwareAddress();
67             String dir_mac = (mac == null) ? "No tiene direccion MAC" : asString(mac);
68             System.out.printf("#%d: %s [%s] MAC:[%s]\n", i++, device.getName(),
69                 description, dir_mac);
70         }
71
72         PcapIf device = alldevs.get(7); // We know we have atleast 1 device
73         System.out
74             .printf("\nChoosing '%s' on your behalf:\n",

```



```

75         (device.getDescription() != null) ? device.getDescription()
76         : device.getName());
77
78         /*****
79         * Second we open up the selected device
80         *****/
81         /*****
82         * "snaplen" is short for 'snapshot length', as it refers to the amount of actual
83         * data captured from each packet passing through the specified network interface.
84         * 64*1024 = 65536 bytes; campo len en Ethernet(16 bits) tam máx de trama */
85
86         int snaplen = 64 * 1024; // Capture all packets, no truncation
87         int flags = Pcap.MODE_PROMISCUOUS; // capture all packets
88         int timeout = 10 * 1000; // 10 seconds in millis
89
90         Pcap pcap =
91         Pcap.openLive(device.getName(), snaplen, flags, timeout, errbuf);
92
93         if (pcap == null) {
94             System.err.printf("Error while opening device for capture: "
95                 + errbuf.toString());
96             return;
97         } //if
98
99         /*****F I L T R O*****/
100         PcapBpfProgram filter = new PcapBpfProgram();
101         String expression = ""; // "port 80";
102         int optimize = 0; // 1 means true, 0 means false
103         int netmask = 0;
104         int r2 = pcap.compile(filter, expression, optimize, netmask);
105         if (r2 != Pcap.OK) {
106             System.out.println("Filter error: " + pcap.getErr());
107         } //if
108         pcap.setFilter(filter);
109         /*****
110         * Third we create a packet handler which will receive packets from the
111         * libpcap loop.
112         *****/
113         PcapPacketHandler<String> jpacketHandler = new PcapPacketHandler<String>() {
114
115             public void nextPacket(PcapPacket packet, String user) {
116
117                 System.out.printf("Received packet at %s caplen=%-4d len=%-4d %s\n",
118                     new Date(packet.getCaptureHeader().timestampInMillis()),
119                     packet.getCaptureHeader().caplen(), // Length actually captured
120                     packet.getCaptureHeader().wirelen(), // Original length
121                     user // User supplied object
122                 );
123
124                 /*****Desencapsulado*****/
125                 for (int i=0; i<packet.size(); i++){
126                     System.out.printf("%02X ", packet.getUByte(i));
127                     if (i%16 == 15)
128                         System.out.println("");
129                 }
130                 System.out.println("\n\nEncabezado: "+ packet.toHexdump());
131
132             }
133         };
134
135         /*****
136         * Fourth we enter the loop and tell it to capture 10 packets. The loop
137         * method does a mapping of pcap.datalink() DLT value to JProtocol ID, which
138         * is needed by JScanner. The scanner scans the packet buffer and decodes
139         * the headers. The mapping is done automatically, although a variation on
140         * the loop method exists that allows the programmer to sepecify exactly
141         * which protocol ID to use as the data link type for this pcap interface.
142         *****/
143         pcap.loop(10, jpacketHandler, "jNetPcap rocks!");
144
145         /*****
146         * Last thing to do is close the pcap handle
147         *****/
148         pcap.close();
149         catch (IOException e) {e.printStackTrace();}
150     }
151 }
152

```

Luego ejecutarlo nos dará lo siguiente:

```

Applications
dom, Jun 17 01:04
Checksum: 10

+ x Checksum: 10
44 32 34 30 38 34 46 36 46 30 42 32 37 43 35 45
44 04 5f 73 75 62 0b 5f 67 6f 67 6c 65 63 61
73 74 04 5f 74 63 70 05 6c 6f 63 61 6c 00 00 0c
00 01 c0 3c 00 00 01

MACo = 01 00 5e 00 00 fb MACd = c0 ee fb 24 fa 25 Tipo = 0800
IPv4
El complemento a uno de la suma de IPv4: 0000
UDP
El complemento a uno de la suma de UDP: 0000

Encabezado: 0000:*01 00 5e 00 00 fb c0 ee fb 24 fa 25 08 00*45 00 ...^.....$.%.E.
0010: 00 7a b7 aa 48 00 ff 11 20 a0 c0 a8 01 44 e0 00 .Z..0...D...
0020: 00 fb 14 e9 14 e9 00 66 2f 43 a0 1b 00 00 00 02 .....f/c.....
0030: 00 00 00 00 00 00 2a 5f 25 39 45 35 45 37 43 38 .....*%9E5E7C8
0040: 46 34 37 39 38 39 35 32 36 43 39 42 43 44 39 35 F47989526C9BCD95
0050: 44 32 34 30 38 34 46 36 46 30 42 32 37 43 35 45 D24084F6F0B27C5E
0060: 44 04 5f 73 75 62 0b 5f 67 6f 6f 67 6c 65 63 61 D..sub..googleca
0070: 73 74 04 5f 74 63 70 05 6c 6f 63 61 6c 00 00 0c st..tcp.local...
0080: 00 01 c0 3c 00 0c 00 01* .....<....

Received packet at Sun Jun 17 01:04:07 CDT 2018 caplen=382 len=382 jNetPcap rocks!
01 00 5e 00 00 fb 80 d2 1d 29 9b 1a 08 00 45 00
01 70 00 00 40 00 ff 11 d7 93 c0 a8 01 45 e0 00
00 fb 14 e9 14 e9 01 5c fa d3 00 00 84 00 00 00
00 01 00 00 00 03 0b 5f 67 6f 6f 67 6c 65 63 61
73 74 04 5f 74 63 70 05 6c 6f 63 61 6c 00 00 0c
00 01 00 00 00 78 00 2e 2b 43 68 72 6f 6d 65 63
61 73 74 2d 61 32 35 39 65 39 33 36 39 38 37 38
32 36 31 65 61 32 30 63 37 37 36 37 36 37 64 38
34 36 62 66 c0 0c c0 2e 00 10 00 01 00 00 11 94
00 a3 23 69 64 3d 61 32 35 39 65 39 33 36 39 38
37 38 32 36 31 65 61 32 38 63 37 37 36 37 36 37
64 38 34 36 62 66 23 63 64 3d 39 42 46 41 38 35
41 45 32 45 45 34 36 30 35 41 36 30 46 42 35 41
35 45 34 33 30 39 45 36 37 31 03 72 6d 3d 05 76
65 3d 38 35 0d 6d 64 3d 43 68 72 6f 6d 65 63 61
73 74 12 69 63 3d 2f 73 65 74 75 70 2f 69 63 6f

```

Figura 3: Ejemplo

```

Applications
dom, jun 17 01:08
Checksum:java
install
+ x Checksum:java
0050: d1 78 57 be 30 42 2b ae ed 2d 0c 23 9c b9 b8 df ..xW.0B+.-.#....
0060: b7 7f 4b ed 16 ea dc 3e* ..K....>

Received packet at Sun Jun 17 01:06:52 CDT 2018 caplen=66 len=66 jNetPcap rocks!
00 21 7c bd b5 b1 ac 37 43 a8 61 d7 08 00 45 00
00 34 a8 d9 40 00 40 06 36 a5 c0 a8 01 54 d8 3a
c1 0e 9a e6 01 b5 1c 0a 51 50 33 af 33 e3 00 10
01 7c 22 0f 00 00 01 01 08 0a 00 4d 06 df 69 be
15 74

MACo = 00 21 7c bd b5 b1 MACd = ac 37 43 a8 61 d7 Tipo = 0000
IPv4
El complemento a uno de la suma de IPv4: 0000
TCP
El complemento a uno de la suma de TCP: 0000

Encabezado: 0000:00 21 7c bd b5 b1 ac 37 43 a8 61 d7 08 00+45 00 .[|....7C.a...E.
0010: 00 34 a8 d9 40 00 40 06 36 a5 c0 a8 01 54 d8 3a .4...@.6....T.:
0020: c1 0e+9a e6 01 b5 1c 0a 51 50 33 af 33 e3 00 10 .....QP3.3...
0030: 01 7c 22 0f 00 00 01 01 08 0a 00 4d 06 df 69 be .[|.....M..1.
0040: 15 74* ..t

Received packet at Sun Jun 17 01:06:52 CDT 2018 caplen=42 len=42 jNetPcap rocks!
ff ff ff ff ff ff 00 21 7c bd b5 b1 08 06 00 01
00 00 06 04 00 01 00 21 7c bd b5 b1 c0 a8 01 fe
ff ff ff ff ff ff c0 a8 01 40

MACo = ff ff ff ff ff ff MACd = 00 21 7c bd b5 b1 Tipo = 0006

Encabezado: 0000:ff ff ff ff ff ff 00 21 7c bd b5 b1 08 06+00 01 .....[|.....
0010: 00 00 06 04 00 01 00 21 7c bd b5 b1 c0 a8 01 fe .....[|.....
0020: ff ff ff ff ff ff c0 a8 01 40* .....@

Received packet at Sun Jun 17 01:06:53 CDT 2018 caplen=42 len=42 jNetPcap rocks!
ff ff ff ff ff ff 00 21 7c bd b5 b1 08 06 00 01
00 00 06 04 00 01 00 21 7c bd b5 b1 c0 a8 01 fe
ff ff ff ff ff ff c0 a8 01 40

MACo = ff ff ff ff ff ff MACd = 00 21 7c bd b5 b1 Tipo = 0006

Encabezado: 0000:ff ff ff ff ff ff 00 21 7c bd b5 b1 08 06+00 01 .....[|.....
0010: 00 00 06 04 00 01 00 21 7c bd b5 b1 c0 a8 01 fe .....[|.....

```

Figura 4: Ejemplo

4. Conclusiones

- Oscar Andrés Rosas Hernandez

Al finalizar esta práctica pude entender mejor como están estructuradas las direcciones MAC, así como también comprender la importancia de estas al ser utilizadas como identificadores.

La práctica fue realmente enriquecedora en cuanto a los temas del curso, ya que me ayudo a entender mejor sobre la estructura de las tramas Ethernet. También por otro lado, me proporciono claridad sobre el principio de encapsulación que se debe de cumplir entre cada una de las capas del modelo OSI.

De igual forma fue fácil comprender la estructura de las tramas que están siendo recibidas o enviadas.

- Arturo Rivas Rojas

Esta práctica es posible realizarla en dos lenguajes de programación: Java y C

Optamos por trabajar en Java debido al paradigma orientado a objetos, que nos ayudará a usar métodos ya implementados en nuestras clases. Utilizamos el IDE Netbeans como apoyo debido a su flexibilidad.

Las MAC destino, MAC origen y Tipo fueron obtenidos gracias al análisis de la trama Ethernet.

Referencias

- [1] E. Ariganello, Redes Cisco. Guia de estudio para la certificacion CCNA 640-802, 2da Edicion, 201