|  |  |  |
| --- | --- | --- |
| **ipn** | **INSTITUTO POLITÉCNICO NACIONAL**  **ESCUELA SUPERIOR DE CÓMPUTO** |  |

**Redes de Computadoras**

**“IP Internet Protocol”**

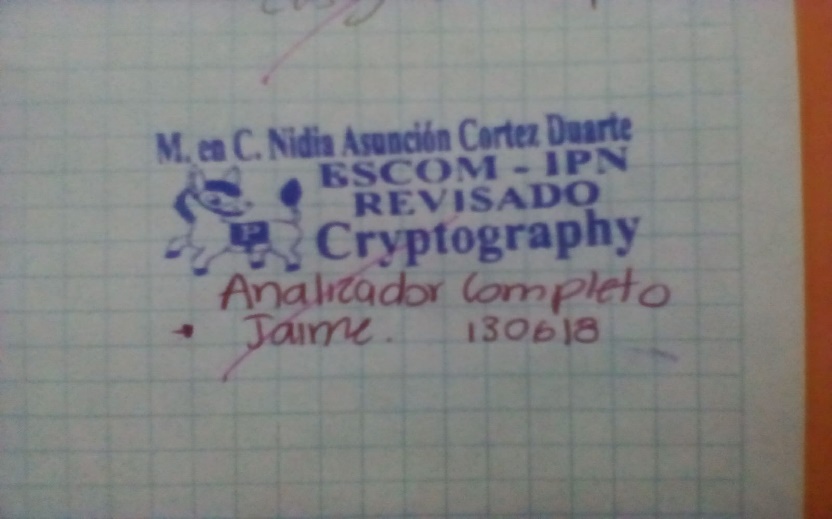
**By:**

**Bastida Prado Jaime Armando**

Professor:

M. en C. NIDIA ASUNCIÓN CORTEZ DUARTE

May 2018

****

**Index**

Contenido

[Problem: 1](#_Toc381082154)

[Hypothesis: 1](#_Toc381082155)

[Software (libraries, packages, tools): 5](#_Toc381082156)

[Procedure: 5](#_Toc381082157)

[Results (Data): 5](#_Toc381082158)

[Conclusions: 9](#_Toc381082159)

[References: 9](#_Toc381082160)

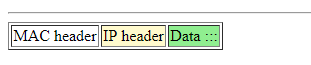
[Code 9](#_Toc381082161)

# Problem:

In this practice we are going to continue with the program for analyzing LLC frames by adding support for IP frames.

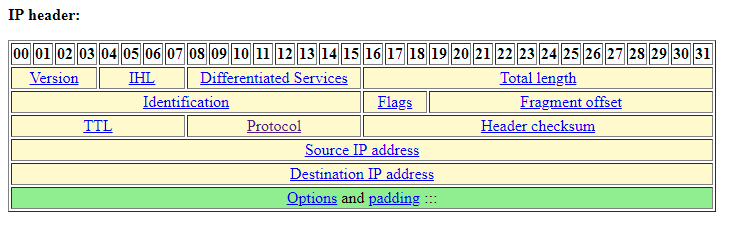
The Internet Protocol (IP) is the principal communications protocol in the Internet protocol suite for relaying datagrams across network boundaries. Its routing function enables internet working, and essentially establishes the Internet.

The IP header goes after the MAC header and before the data:



*Figure 1: Ethernet header*

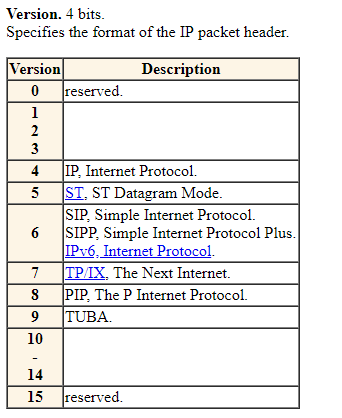
The IP header looks like this:



*Figure 2: IP header*

Its **EtherType** corresponds to the value **0x0800 = 2048**.

Version values can be any of the next values as seen in the table:

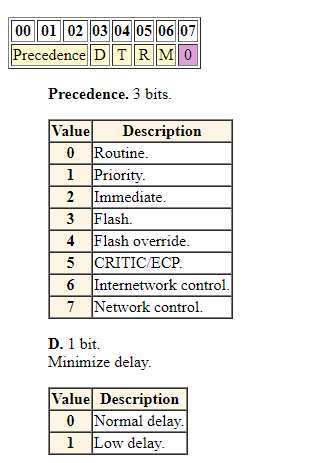


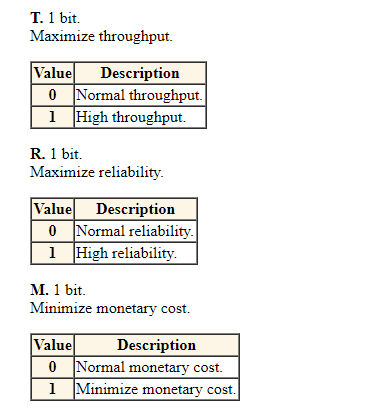
*Figure 3: Version codes and meaning*

The typically used are 4 and 6 meaning IPv4 and IPv6.

**IHL (Internet Header Length)** specifies the length of the IP packet header in 32-bit words. The minimum value for a valid header is 5.

**TTL (Type of Service)** specifies the parameters for the type of service requested. The M bit was added to this field in RFC 1394. For this practice we are not going to use the Precedence bits.

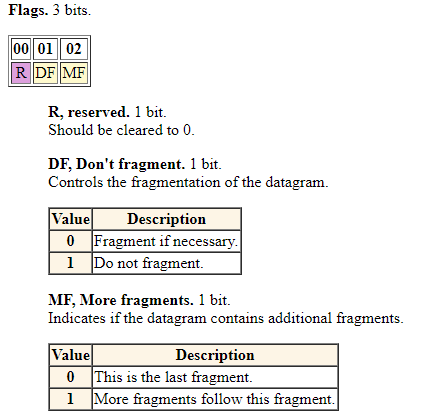




*Figure 4: Type of Service code and meaning*

**Total Length** contains the length of the datagram.

**Identification** used to identify the fragments of one datagram from those of another. The originating protocol module of an internet datagram sets the identification field to a value that must be unique for that source-destination pair and protocol for the time the datagram will be active in the internet system. The originating protocol module of a complete datagram clears the MF bit to zero and the Fragment Offset field to zero.



*Figure 5: Flags codes and meaning*

**Fragment Offset** used to direct the reassembly of a fragmented datagram.

**TTL (Time to Live)** a timer field used to track the lifetime of the datagram. When the TTL field is decremented down to zero, the datagram is discarded.

**Protocol** this field specifies the next encapsulated protocol. There are many protocols that could follow but we are only going to analyze TCP (value = 6), ICMP (value = 1) and UDP (value = 17).

**Header Checksum** a 16 bit one's complement checksum of the IP header and IP options.

**Source IP** **Address** IP address of the sender.

**Destination IP Address** IP address of the intended receiver.

**Options** variable length. We are not going to analyze the options just print them in case there are any that is when the value of IHL is greater than 20 bytes.

# Hypothesis:

The solution to the problem is to add the functionality to analyze IP frames to our original program for LLC frames by consulting all the information given by the teacher and in the web.

# Software (libraries, packages, tools):

The equipment used:

-Laptop

-Pencil

-Sheets

-The program was codified in C language using text editor Sublime Text 3

-Compiler: GNU (GCC) compiler.

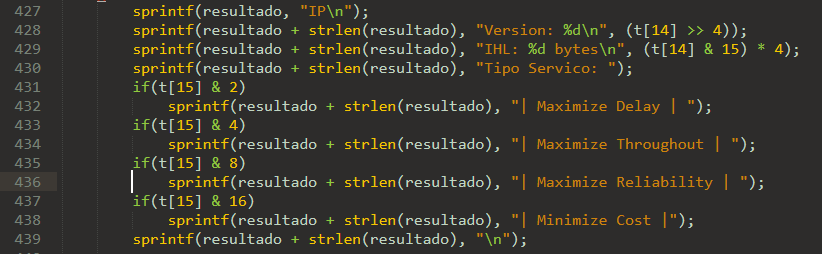
# Procedure:

Follow the steps given in the LLC report up to the point where the ToT variable is checked for the value 2048 inside an if statement:



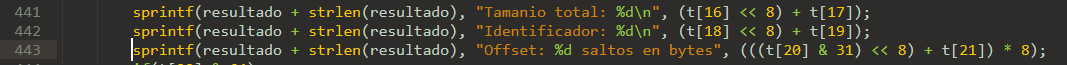
*Figure 6: IP case*

Now check for **Version**, **IHL** and **Type of Service** with simple shifts and binary operators:



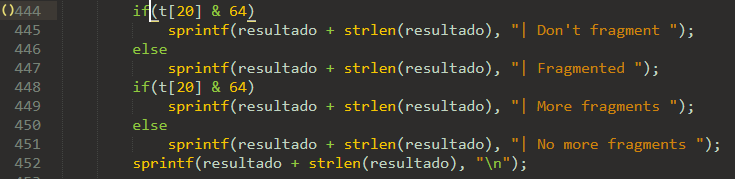
*Figure 7: Obtaining Version, IHL and Type of Service*

Next, get the values of **Total Length** and **Identification**. Note that since they are given in 16 bits we need to do a shift of 8 bits to the first byte t[16] and then add t[17], resulting in a number of 16 bits. The same logic applies to the **Offset** but with the difference that Offset is given in words of 8 bits so we need to multiply by 8:



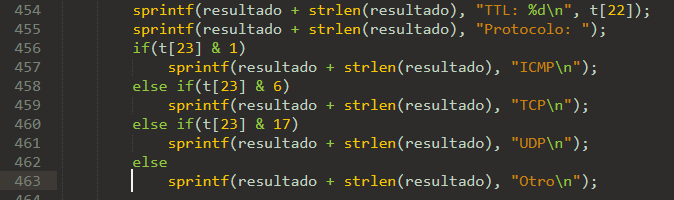
*Figure 9: Obtaining Total Length, Identification and Offset*

After that, check for the **Flags** by just checking if the correspondent bits are On or Off:



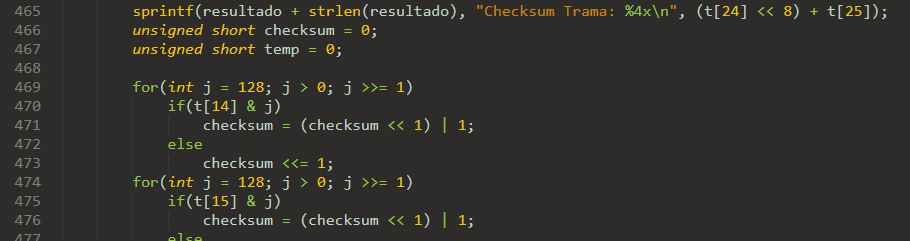
*Figure 10: Checking the Flags*

Now, get **TTL** and which protocol follows **IP:**

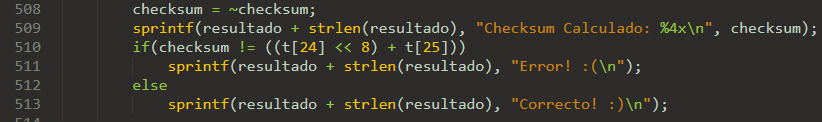


*Figure 11: Obtaining TTL and following Protocol*

The next thing to do is to apply the **Checksum** over the header and compare the result with the **Checksum** given:

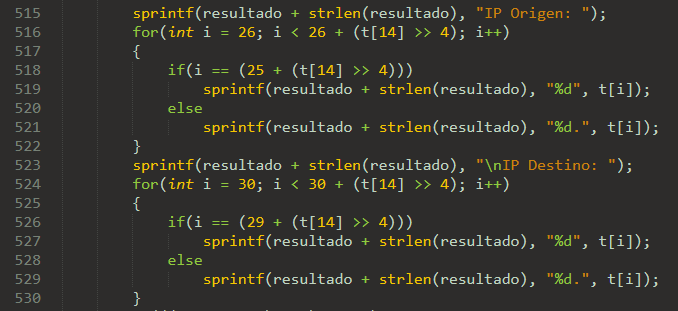


*Figure 12: Printing the included Checksum and calculating our own*



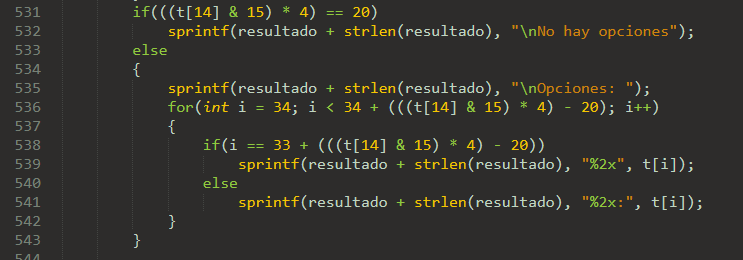
*Figure 13: Printing the checksum calculated and telling if correct or not*

Almost finally, print the **IP addresses:**



*Figure 14: Printing the addresses*

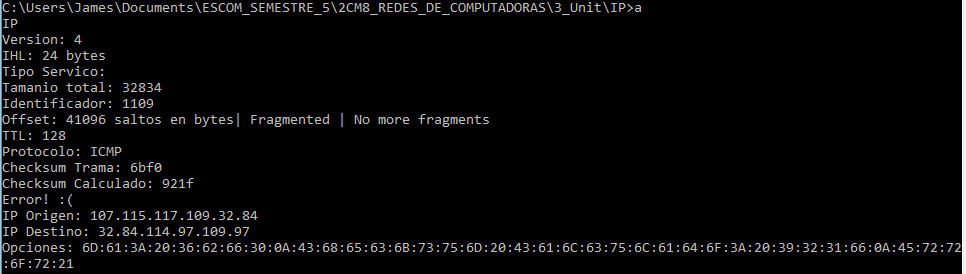
Finally, if the **IHL** value is 20 bytes then there are no **Options** otherwise print the **Options** in hex form:



*Figure 15: Printing Options if exist*

# Results (Data):

An Example:



*Figure 16: Result shown in console for an IP frame*

# Conclusions:

The hypothesis was correct since we only needed to consult the information given by the teacher and a very useful web page listed in the References section.

# References:

K.N. King, C Programming A Modern Approach Second Edition. W.W. Norton, 2008.

Network Sorcery. (2012). IP, Internet Protocol. Retrieved from <http://www.networksorcery.com/enp/protocol/ip.htm>

# Code

1. /\*Notes: Windows doesn't seem to recognize T[18] as number 6 for example in the ARP protocol, so I needed to use the number instead
2. of the variable, this could lead to several errors\*/
4. #include <stdio.h>
5. #include <string.h>
6. #include <stdlib.h>
8. **typedef** unsigned **char** **BYTE**;
10. **int** analizaTrama(**BYTE** t[], **BYTE** resultado[]);
12. **int** main(**void**)
13. {
14. //LLC frames
15. /\*BYTE t[ ][32] = { {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x03,0xf0,0xf0, 0x7f,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}, //Trama 1
17. {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x03,0xf0,0xf1, 0x73,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}, //Trama 2
19. {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x04,0xf0,0xf0, 0x01,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},//Trama 3
21. {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x04,0xf0,0xf1, 0x01,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}, //Trama 4
23. {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x12,0xf0,0xf0, 0x00,0x01,0x0e,0x00,0xff,0xef,0x19,0x8f,0xbc,0x05,0x7f,0x00,0x23,0x00,0x7f,0x23}, //Trama 5
25. {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x12,0xf0,0xf0, 0x00,0x03,0x0e,0x00,0xff,0xef,0x19,0x8f,0xbc,0x05,0x7f,0x00,0x23,0x00,0x7f,0x23}, //Trama 6
27. {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x04,0xf0,0xf1, 0x01,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}, //Trama 7
29. {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x04,0xf0,0xf1, 0x01,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}, //Trama 8
31. {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0xac,0xf0,0xf0, 0x02,0x02,0x0e,0x00,0xff,0xef,0x16,0x04,0x00,0x00,0x00,0x00,0x28,0x00,0x7f,0x23}, //Trama 9
33. {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x04,0xf0,0xf1, 0x01,0x04,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}, //Trama 10
35. {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x5f,0xf0,0xf0, 0x02,0x04,0x0e,0x00,0xff,0xef,0x16,0x0c,0x00,0x00,0x28,0x00,0x28,0x00,0x7f,0x23}, //Trama 11
37. {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x04,0xf0,0xf1, 0x01,0x04,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}, //Trama 12
39. {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x91,0xf0,0xf0, 0x04,0x04,0x0e,0x00,0xff,0xef,0x16,0x0c,0x00,0x00,0x28,0x00,0x28,0x00,0x7f,0x23}, //Trama 13
41. {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x04,0xf0,0xf1, 0x01,0x06,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}, //Trama 14
43. {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x46,0xf0,0xf0, 0x04,0x06,0x0e,0x00,0xff,0xef,0x16,0x0c,0x00,0x00,0x28,0x00,0x28,0x00,0x7f,0x23}, //Trama 15
45. {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x04,0xf0,0xf1, 0x01,0x06,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}, //Trama 16
47. {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x7e,0xf0,0xf0, 0x06,0x06,0x0e,0x00,0xff,0xef,0x16,0x0c,0x00,0x00,0x28,0x00,0x28,0x00,0x7f,0x23}, //Trama 17
49. {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x04,0xf0,0xf1, 0x01,0x08,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}, //Trama 18
51. {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x04,0xf0,0xf1, 0x01,0x08,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}, //Trama 19
53. {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x7e,0xf0,0xf0, 0x08,0x08,0x0e,0x00,0xff,0xef,0x16,0x0c,0x00,0x00,0x28,0x00,0x28,0x00,0x7f,0x23}, //Trama 20
55. {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x04,0xf0,0xf1, 0x01,0x0a,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}, //Trama 21
57. {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x04,0xf0,0xf1, 0x01,0x0a,0x0e,0x00,0xff,0xef,0x19,0x8f,0xbc,0x05,0x7f,0x00,0x23,0x00,0x7f,0x23}, //Trama 22
59. {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x12,0xf0,0xf0, 0x0a,0x0b,0x0e,0x00,0xff,0xef,0x14,0x00,0x00,0x00,0x28,0x00,0x00,0x00,0x7f,0x23}, //Trama 23
61. {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x04,0xf0,0xf1, 0x01,0x0d,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}, //Trama 24
63. {0x03,0x00,0x00,0x00,0x00,0x01,0x00,0x04,0xac,0x44,0x4d,0x02,0x00,0x8b,0xf0,0xf0, 0x03,0x2c,0x00,0xff,0xef,0x08,0x00,0x00,0x00,0x00,0x00,0x28,0x00,0x42,0x34,0x20}, //Trama 25
65. {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x35,0xf0,0xf0, 0x0c,0x0a,0x0e,0x00,0xff,0xef,0x16,0x04,0x00,0x00,0x00,0x00,0x28,0x00,0x7f,0x23}, //Trama 26
67. {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x35,0xf0,0xf0, 0x0a,0x0e,0x0e,0x00,0xff,0xef,0x16,0x04,0x00,0x00,0x00,0x00,0x28,0x00,0x7f,0x23}, //Trama 27
69. {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x12,0xf0,0xf0, 0x0e,0x0d,0x0e,0x00,0xff,0xef,0x14,0x00,0x00,0x00,0x28,0x00,0x00,0x00,0x7f,0x23}, //Trama 28
71. {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x04,0xf0,0xf1, 0x01,0x11,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}, //Trama 29
73. {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x12,0xf0,0xf0, 0x10,0x0d,0x0e,0x00,0xff,0xef,0x18,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7f,0x23}, //Trama 30
75. {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x04,0xf0,0xf1, 0x01,0x13,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}, //Trama 31
77. {0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x03,0xf0,0xf0, 0x53,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}, //Trama 32
79. {0x00,0x02,0xb3,0x9c,0xdf,0x1b,0x00,0x02,0xb3,0x9c,0xae,0xba,0x00,0x03,0xf0,0xf1, 0x73,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}}; //Trama 33\*/
80. //ARP frames
81. //Remove the "t1" I made that way because I wanted to prove something else
82. **BYTE** t1[20][125]={
83. /\* TRAMA 1 \*/
84. {
85. 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x23, /\* .......# \*/
86. 0x8b, 0x46, 0xe9, 0xad, 0x08, 0x06, 0x00, 0x04, /\* .F...... \*/
87. 0x08, 0x00, 0x06, 0x04, 0x00, 0x01, 0x00, 0x23, /\* .......# \*/
88. 0x8b, 0x46, 0xe9, 0xad, 0x94, 0xcc, 0x39, 0xcb, /\* .F....9. \*/
89. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x94, 0xcc, /\* ........ \*/
90. 0x39, 0xfe                                      /\* 9. \*/
91. },
93. /\*TRAMA 2 \*/
94. {
95. 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x00, 0x1f, /\* .#.F.... \*/
96. 0x45, 0x9d, 0x1e, 0xa2, 0x08, 0x06, 0x00, 0x01, /\* E....... \*/
97. 0x08, 0x00, 0x06, 0x04, 0x00, 0x02, 0x00, 0x1f, /\* ........ \*/
98. 0x45, 0x9d, 0x1e, 0xa2, 0x94, 0xcc, 0x39, 0xfe, /\* E.....9. \*/
99. 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x94, 0xcc, /\* .#.F.... \*/
100. 0x39, 0xcb, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* 9....... \*/
101. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
102. 0x00, 0x00, 0x00, 0x00                          /\* .... \*/
103. },
105. /\* TRAMA 3 \*/
106. {
107. 0x00, 0x1f, 0x45, 0x9d, 0x1e, 0xa2, 0x00, 0x23, /\* ..E....# \*/
108. 0x8b, 0x46, 0xe9, 0xad, 0x08, 0x00, 0x46, 0x00, /\* .F....F. \*/
109. 0x80, 0x42, 0x04, 0x55, 0x34, 0x11, 0x80, 0x11, /\* .B.U4... \*/
110. 0x6b, 0xf0, 0x94, 0xcc, 0x39, 0xcb, 0x94, 0xcc, /\* k...9... \*/
111. 0x67, 0x02, 0xaa, 0xbb, 0xcc, 0xdd, 0x04, 0x0c, /\* g....... \*/
112. 0x00, 0x35, 0x00, 0x2e, 0x85, 0x7c, 0xe2, 0x1a, /\* .5...|.. \*/
113. 0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
114. 0x00, 0x00, 0x03, 0x77, 0x77, 0x77, 0x03, 0x69, /\* ...www.i \*/
115. 0x73, 0x63, 0x05, 0x65, 0x73, 0x63, 0x6f, 0x6d, /\* sc.escom \*/
116. 0x03, 0x69, 0x70, 0x6e, 0x02, 0x6d, 0x78, 0x00, /\* .ipn.mx. \*/
117. 0x00, 0x1c, 0x00, 0x01                          /\* .... \*/
118. },
120. /\* TRAMA 4 \*/
121. {
122. 0x00, 0x02, 0xb3, 0x9c, 0xdf, 0x1b, 0x00, 0x02, /\* ........ \*/
123. 0xb3, 0x9c, 0xae, 0xba, 0x00, 0x04, 0xf0, 0xf0, /\* ........ \*/
124. 0x01, 0xc9, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
125. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
126. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
127. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
128. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
129. 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0x9b, 0x6d  /\* .....|.m \*/
130. },
132. /\* TRAMA 5 \*/
133. {
134. 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x23, /\* .......# \*/
135. 0x8b, 0x46, 0xe9, 0xad, 0x08, 0x06, 0x00, 0x10, /\* .F...... \*/
136. 0x08, 0x00, 0x06, 0x04, 0x00, 0x03, 0x00, 0x23, /\* .......# \*/
137. 0x8b, 0x46, 0xe9, 0xad, 0x94, 0xcc, 0x39, 0xcb, /\* .F....9. \*/
138. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x94, 0xcc, /\* ........ \*/
139. 0x3a, 0xe1                                      /\* :. \*/
140. },
142. /\* TRAMA 6\*/
143. {
144. 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x00, 0x1f, /\* .#.F.... \*/
145. 0x45, 0x9d, 0x1e, 0xa2, 0x08, 0x06, 0x00, 0x10, /\* E....... \*/
146. 0x08, 0x00, 0x06, 0x04, 0x00, 0x04, 0x00, 0x1f, /\* ........ \*/
147. 0x45, 0x9d, 0x1e, 0xa2, 0x94, 0xcc, 0x3a, 0xe1, /\* E.....:. \*/
148. 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x94, 0xcc, /\* .#.F.... \*/
149. 0x39, 0xcb, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* 9....... \*/
150. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
151. 0x00, 0x00, 0x00, 0x00                          /\* .... \*/
152. },
154. /\* TRAMA 7 \*/
155. {
156. 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x00, 0x1f, /\* .#.F.... \*/
157. 0x45, 0x9d, 0x1e, 0xa2, 0x08, 0x00, 0x45, 0x00, /\* E.....E. \*/
158. 0x00, 0x6f, 0x90, 0x30, 0x40, 0x00, 0xfb, 0x11, /\* .o.0@... \*/
159. 0x24, 0xe7, 0x94, 0xcc, 0x67, 0x02, 0x94, 0xcc, /\* $...g... \*/
160. 0x39, 0xcb, 0x00, 0x35, 0x04, 0x0c, 0x00, 0x5b, /\* 9..5...[ \*/
161. 0xe8, 0x60, 0xe2, 0x1a, 0x85, 0x80, 0x00, 0x01, /\* .`...... \*/
162. 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x03, 0x77, /\* .......w \*/
163. 0x77, 0x77, 0x03, 0x69, 0x73, 0x63, 0x05, 0x65, /\* ww.isc.e \*/
164. 0x73, 0x63, 0x6f, 0x6d, 0x03, 0x69, 0x70, 0x6e, /\* scom.ipn \*/
165. 0x02, 0x6d, 0x78, 0x00, 0x00, 0x1c, 0x00, 0x01, /\* .mx..... \*/
166. 0xc0, 0x14, 0x00, 0x06, 0x00, 0x01, 0x00, 0x00, /\* ........ \*/
167. 0x0e, 0x10, 0x00, 0x21, 0x04, 0x64, 0x6e, 0x73, /\* ...!.dns \*/
168. 0x31, 0xc0, 0x1a, 0x03, 0x74, 0x69, 0x63, 0xc0, /\* 1...tic. \*/
169. 0x1a, 0x77, 0xec, 0xdf, 0x29, 0x00, 0x00, 0x2a, /\* .w..)..\* \*/
170. 0x30, 0x00, 0x00, 0x0e, 0x10, 0x00, 0x12, 0x75, /\* 0......u \*/
171. 0x00, 0x00, 0x00, 0x2a, 0x30                    /\* ...\*0 \*/
172. },
174. /\* TRAMA 8 \*/
175. {
176. 0x00, 0x1f, 0x45, 0x9d, 0x1e, 0xa2, 0x00, 0x23, /\* ..E....# \*/
177. 0x02, 0x03, 0x04, 0x05, 0x08, 0x00, 0x45, 0x00, /\* .F....E. \*/
178. 0x00, 0x42, 0x04, 0x56, 0x00, 0x00, 0x80, 0x11, /\* .B.V.... \*/
179. 0x6b, 0xef, 0x94, 0xcc, 0x39, 0xcb, 0x94, 0xcc, /\* k...9... \*/
180. 0x67, 0x02, 0x04, 0x0c, 0x00, 0x35, 0x00, 0x2e, /\* g....5.. \*/
181. 0xff, 0x87, 0x68, 0x2a, 0x01, 0x00, 0x00, 0x01, /\* ..h\*.... \*/
182. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x77, /\* .......w \*/
183. 0x77, 0x77, 0x03, 0x69, 0x73, 0x63, 0x05, 0x65, /\* ww.isc.e \*/
184. 0x73, 0x63, 0x6f, 0x6d, 0x03, 0x69, 0x70, 0x6e, /\* scom.ipn \*/
185. 0x02, 0x6d, 0x78, 0x00, 0x00, 0x01, 0x00, 0x01  /\* .mx..... \*/
186. },
188. /\* TRAMA 9 \*/
189. {
190. 0x00, 0x1f, 0x45, 0x9d, 0x1e, 0xa2, 0x00, 0x23, /\* ..E....# \*/
191. 0x8b, 0x46, 0xe9, 0xad, 0x08, 0x00, 0x45, 0x00, /\* .F....E. \*/
192. 0x00, 0x3c, 0x04, 0x57, 0x00, 0x00, 0x80, 0x01, /\* .<.W.... \*/
193. 0x98, 0x25, 0x94, 0xcc, 0x39, 0xcb, 0x94, 0xcc, /\* .%..9... \*/
194. 0x3a, 0xe1, 0x08, 0x00, 0x49, 0x5c, 0x03, 0x00, /\* :...I\.. \*/
195. 0x01, 0x00, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, /\* ..abcdef \*/
196. 0x67, 0x68, 0x69, 0x6a, 0x6b, 0x6c, 0x6d, 0x6e, /\* ghijklmn \*/
197. 0x6f, 0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, /\* opqrstuv \*/
198. 0x77, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, /\* wabcdefg \*/
199. 0x68, 0x69                                      /\* hi \*/
200. },
202. /\* TRAMA 10 \*/
203. {
204. 0x00, 0x02, 0xb3, 0x9c, 0xdf, 0x1b, 0x00, 0x02, /\* ........ \*/
205. 0x02, 0x03, 0x04, 0x05, 0x00, 0x04, 0xf0, 0xf0, /\* ........ \*/
206. 0x0c, 0xc7, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
207. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
208. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
209. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
210. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
211. 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0x9b, 0x6d  /\* .....|.m \*/
212. },
214. /\* trama 11 \*/
215. {
216. 0x00, 0x02, 0xb3, 0x9c, 0xae, 0xba, 0x11, 0x11, /\* ........ \*/
217. 0x11, 0x11, 0x11, 0x11, 0x00, 0x03, 0xf0, 0xf0, /\* ........ \*/
218. 0x63, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
219. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
220. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
221. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
222. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
223. 0x00, 0x00, 0x00, 0x00, 0x43, 0x05, 0x90, 0x6d  /\* ....C..m \*/
224. },
226. /\* trama 12 \*/
227. {
228. 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x00, 0x1f, /\* .#.F.... \*/
229. 0x02, 0x03, 0x04, 0x05, 0x08, 0x00, 0x45, 0x00, /\* E.....E. \*/
230. 0x00, 0x3c, 0x01, 0xb5, 0x00, 0x00, 0x3f, 0x01, /\* .<....?. \*/
231. 0xdb, 0xc7, 0x94, 0xcc, 0x3a, 0xe1, 0x94, 0xcc, /\* ....:... \*/
232. 0x39, 0xcb, 0x00, 0x00, 0x51, 0x5c, 0x03, 0x00, /\* 9...Q\.. \*/
233. 0x01, 0x00, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, /\* ..abcdef \*/
234. 0x67, 0x68, 0x69, 0x6a, 0x6b, 0x6c, 0x6d, 0x6e, /\* ghijklmn \*/
235. 0x6f, 0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, /\* opqrstuv \*/
236. 0x77, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, /\* wabcdefg \*/
237. 0x68, 0x69                                      /\* hi \*/
238. },
240. /\* trama 13 \*/
241. {
242. 0x00, 0x02, 0xb3, 0x9c, 0xae, 0xba, 0x00, 0x01, /\* ........ \*/
243. 0x02, 0x03, 0x04, 0x05, 0x00, 0x03, 0xf0, 0xf1, /\* ........ \*/
244. 0x73, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
245. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
246. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
247. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
248. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
249. 0x00, 0x00, 0x00, 0x00, 0x43, 0x05, 0x90, 0x6d  /\* ....C..m \*/
250. },
252. /\* trama 14 \*/
253. {
254. 0x00, 0x1f, 0x45, 0x9d, 0x1e, 0xa2, 0x00, 0x1c, /\* ..E..... \*/
255. 0xc0, 0x7b, 0x35, 0xa1, 0x08, 0x00, 0x48, 0x00, /\* .{5...H. \*/
256. 0x00, 0x48, 0x5c, 0x7d, 0x00, 0x00, 0x80, 0x01, /\* .H\}.... \*/
257. 0x6c, 0x88, 0x94, 0xcc, 0x39, 0xc3, 0x94, 0xcc, /\* l...9... \*/
258. 0x00, 0x49, 0x07, 0x0b, 0x04, 0x00, 0x00, 0x00, /\* .I...... \*/
259. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x00, /\* ........ \*/
260. 0x3b, 0x5c, 0x02, 0x00, 0x10, 0x00, 0x61, 0x62, /\* ;\....ab \*/
261. 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69, 0x6a, /\* cdefghij \*/
262. 0x6b, 0x6c, 0x6d, 0x6e, 0x6f, 0x70, 0x71, 0x72, /\* klmnopqr \*/
263. 0x73, 0x74, 0x75, 0x76, 0x77, 0x61, 0x62, 0x63, /\* stuvwabc \*/
264. 0x64, 0x65, 0x66, 0x67, 0x68, 0x69              /\* defghi \*/
265. },
267. /\* trama 15\*/
268. {
269. 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x00, 0x1f, /\* .#.F.... \*/
270. 0x45, 0x9d, 0x1e, 0xa2, 0x80, 0x35, 0x00, 0x01, /\* E....5.. \*/
271. 0x08, 0x00, 0x06, 0x04, 0x00, 0x03, 0x00, 0x1f, /\* ........ \*/
272. 0x45, 0x9d, 0x1e, 0xa2, 0x94, 0xcc, 0x3a, 0xe1, /\* E.....:. \*/
273. 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x94, 0xcc, /\* .#.F.... \*/
274. 0x39, 0xcb, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* 9....... \*/
275. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
276. 0x00, 0x00, 0x00, 0x00, 0xd8, 0xee, 0xdf, 0xb0  /\* ........ \*/
277. },
279. /\* trama 16 \*/
280. {
281. 0x00, 0x02, 0xb3, 0x9c, 0xae, 0xba, 0x22, 0x22, /\* ........ \*/
282. 0x22, 0x22, 0x22, 0x22, 0x00, 0x03, 0xf0, 0xf0, /\* ........ \*/
283. 0x73, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
284. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
285. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
286. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
287. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
288. 0x00, 0x00, 0x00, 0x00, 0x43, 0x05, 0x90, 0x6d  /\* ....C..m \*/
289. },
291. /\*trama 17 \*/
292. {
293. 0x00, 0x02, 0xb3, 0x9c, 0xae, 0xba, 0x00, 0x02, /\* ........ \*/
294. 0xb3, 0x9c, 0xdf, 0x1b, 0x00, 0x03, 0xf0, 0xf0, /\* ........ \*/
295. 0x53, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
296. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
297. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
298. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
299. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
300. 0x00, 0x00, 0x00, 0x00, 0x43, 0x05, 0x90, 0x6d  /\* ....C..m \*/
301. },
303. /\* trama 18 \*/
304. {
305. 0x02, 0xff, 0x53, 0xc3, 0xe9, 0xab, 0x00, 0xff, /\* ..S..... \*/
306. 0x66, 0x7f, 0xd4, 0x3c, 0x08, 0x00, 0x45, 0x00, /\* f..<..E. \*/
307. 0x00, 0x30, 0x2c, 0x00, 0x40, 0x00, 0x80, 0x06, /\* .0,.@... \*/
308. 0x4b, 0x74, 0xc0, 0xa8, 0x01, 0x02, 0xc0, 0xa8, /\* Kt...... \*/
309. 0x01, 0x01, 0x04, 0x03, 0x00, 0x15, 0x00, 0x3b, /\* .......; \*/
310. 0xcf, 0x44, 0x00, 0x00, 0x00, 0x00, 0x70, 0x20, /\* .D....p  \*/
311. 0x20, 0x00, 0x0c, 0x34, 0x00, 0x00, 0x02, 0x04, /\*  ..4.... \*/
312. 0x05, 0xb4, 0x01, 0x01, 0x04, 0x02              /\* ...... \*/
313. },
315. /\*trama 19 \*/
316. {
317. 0x00, 0x02, 0xb3, 0x9c, 0xae, 0xba, 0x00, 0x02, /\* ........ \*/
318. 0xb3, 0x9c, 0xdf, 0x1b, 0x00, 0x03, 0xf0, 0xf1, /\* ........ \*/
319. 0x53, 0x0b, 0x0e, 0x00, 0xff, 0xef, 0x14, 0x00, /\* ........ \*/
320. 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x7f, 0x23, /\* ..(....# \*/
321. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
322. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
323. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /\* ........ \*/
324. 0x00, 0x00, 0x00, 0x00, 0x01, 0x99, 0x98, 0x6d  /\* .......m \*/
325. },
326. /\* trama 20 \*/
327. {
328. 0x00, 0xff, 0x66, 0x7f, 0xd4, 0x3c, 0x02, 0xff, /\* ..f..<.. \*/
329. 0x53, 0xc3, 0xe9, 0xab, 0x08, 0x00, 0x45, 0x00, /\* S.....E. \*/
330. 0x00, 0x30, 0x05, 0xc4, 0x40, 0x00, 0x80, 0x06, /\* .0..@... \*/
331. 0x71, 0xb0, 0xc0, 0xa8, 0x01, 0x01, 0xc0, 0xa8, /\* q....... \*/
332. 0x01, 0x02, 0x00, 0x15, 0x04, 0x03, 0x21, 0x5d, /\* ......!] \*/
333. 0x3a, 0x44, 0x00, 0x3b, 0xcf, 0x45, 0x70, 0x12, /\* :D.;.Ep. \*/
334. 0x44, 0x70, 0x8c, 0x11, 0x00, 0x00, 0x02, 0x04, /\* Dp...... \*/
335. 0x05, 0xb4, 0x01, 0x01, 0x04, 0x02              /\* ...... \*/
336. }};
337. /\*
338. BYTE t[] = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x08, 0x06, 0x00, 0x01,
339. 0x08, 0x00, 0x06, 0x04, 0x00, 0x01, 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x94, 0xcc, 0x39, 0xcb,
340. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x94, 0xcc, 0x39, 0xfe};
341. \*/
342. //For IP
344. **BYTE** t[] = {0x13, 0x42, 0x18, 0x63, 0x96, 0x12, 0x13, 0x42,  0x18, 0x96, 0x35, 0x72, 0x08, 0x00, 0x46, 0x00,
345. 0x80, 0x42, 0x04, 0x55, 0x34, 0x11, 0x80, 0x11,  0x6b, 0xf0, 0x94, 0xcc, 0x39, 0xcb, 0x94, 0xcc,
346. 0x67, 0x02, 0xaa, 0xbb, 0xcc, 0xdd, 0x04, 0x0c,  0x00, 0x35, 0x00, 0x2e, 0x85, 0x7c, 0xe2, 0x1a,
347. 0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x03, 0x77, 0x77, 0x77, 0x03, 0x69,
348. 0x73, 0x63, 0x05, 0x65, 0x73, 0x63, 0x6f, 0x6d,  0x03, 0x69, 0x70, 0x6e, 0x02, 0x6d, 0x78, 0x00,
349. 0x00, 0x1c, 0x00, 0x01};
351. **BYTE** resultado[150];
353. //For just one frame
355. **if**(analizaTrama(t, resultado))
356. printf("%s\n", resultado);
357. **else**
358. printf("No es trama LLC\n");

361. //For LLC frames
362. /\*
363. for(BYTE i = 0; i < 33; i++)
364. if(analizaTrama(t[i], resultado))
365. printf("%d %s\n", i + 1, resultado);
366. else
367. printf("No es trama LLC\n");
368. \*/
370. //For ARP frames
371. /\*
372. for(BYTE i = 0; i < 5; i++)
373. if(analizaTrama(t[i], resultado))
374. printf("%d %s\n\n", i + 1, resultado);
375. else
376. printf("No es trama LLC\n");
377. \*/
378. exit(EXIT\_SUCCESS);
379. }
381. **int** analizaTrama(**BYTE** t[], **BYTE** resultado[])
382. {
383. **BYTE** supervision[][5] = {"RR", "RNR", "REJ", "SREJ"};
384. **BYTE** uc[][6] = {"UI", "SIM", "-", "SARN", "UP", "-", "-", "SABM", "DISC", "-", "-", "SARME", "-", "-", "-", "SABME",
385. "SNRM", "-", "-", "XID", "-", "-", "-", "-", "-", "-", "-", "SNRME"};
386. **BYTE** ur[][5] = {"UI", "RIM", "-", "DM", "-", "-", "-", "-", "RD", "-", "-", "-", "UA", "-", "-", "-", "-", "FRMR", "-",
387. "XID", "-", "-", "-", "-", "-", "-", "-", "-"};
388. unsigned **short** **int** ToT = 0;
390. //Copiamos los bytes 12 y 13 de la trama en ToT (Tamaño o tipo)
391. **for**(**BYTE** j = 128; j > 0; j >>= 1)
392. **if**(t[12] & j)
393. ToT = (ToT << 1) | 1;
394. **else**
395. ToT <<= 1;
397. **for**(**BYTE** j = 128; j > 0; j >>= 1)
398. **if**(t[13] & j)
399. ToT = (ToT << 1) | 1;
400. **else**
401. ToT <<= 1;
403. //Tipo LLC
404. **if**(ToT <= 1500)
405. {
406. **switch**(t[16] & 3)
407. {
408. **case** 0: **case** 2:
409. sprintf(resultado, "T-I N(s) = %d, N(r) = %d", ((t[16] >> 1) & 255), ((t[17] >> 1) & 255));
410. **break**;
411. **case** 1:
412. sprintf(resultado, "T-S %s, N(r) = %d", supervision[(t[16] >> 2) & 3], ((t[17] >> 1) & 255));
413. **break**;
414. **case** 3:
415. **if**(((t[16] >> 4) & 1) && (t[15] & 1)) //Respuesta no numerada
416. sprintf(resultado, "T-U %s", ur[((t[16] >> 2) & 3) | ((t[16] >> 3) & 28)]);
417. **else** **if**(((t[16] >> 4) & 1) && !(t[15] & 0)) //Comando no numerado
418. sprintf(resultado, "T-U %s", uc[((t[16] >> 2) & 3) | ((t[16] >> 3) & 28)]);
419. **else** //?
420. sprintf(resultado, "Trama Intrusa");
421. **break**;
422. }
424. **return** 1;
425. }
426. **else** **if**(ToT == 2048) //Tipo IP
427. {
428. sprintf(resultado, "IP\n");
429. sprintf(resultado + strlen(resultado), "Version: %d\n", (t[14] >> 4));
430. sprintf(resultado + strlen(resultado), "IHL: %d bytes\n", (t[14] & 15) \* 4);
431. sprintf(resultado + strlen(resultado), "Tipo Servico: ");
432. **if**(t[15] & 2)
433. sprintf(resultado + strlen(resultado), "| Maximize Delay | ");
434. **if**(t[15] & 4)
435. sprintf(resultado + strlen(resultado), "| Maximize Throughout | ");
436. **if**(t[15] & 8)
437. sprintf(resultado + strlen(resultado), "| Maximize Reliability | ");
438. **if**(t[15] & 16)
439. sprintf(resultado + strlen(resultado), "| Minimize Cost |");
440. sprintf(resultado + strlen(resultado), "\n");
442. sprintf(resultado + strlen(resultado), "Tamanio total: %d\n", (t[16] << 8) + t[17]);
443. sprintf(resultado + strlen(resultado), "Identificador: %d\n", (t[18] << 8) + t[19]);
444. sprintf(resultado + strlen(resultado), "Offset: %d saltos en bytes", (((t[20] & 31) << 8) + t[21]) \* 8);
445. **if**(t[20] & 64)
446. sprintf(resultado + strlen(resultado), "| Don't fragment ");
447. **else**
448. sprintf(resultado + strlen(resultado), "| Fragmented ");
449. **if**(t[20] & 64)
450. sprintf(resultado + strlen(resultado), "| More fragments ");
451. **else**
452. sprintf(resultado + strlen(resultado), "| No more fragments ");
453. sprintf(resultado + strlen(resultado), "\n");
455. sprintf(resultado + strlen(resultado), "TTL: %d\n", t[22]);
456. sprintf(resultado + strlen(resultado), "Protocolo: ");
457. **if**(t[23] & 1)
458. sprintf(resultado + strlen(resultado), "ICMP\n");
459. **else** **if**(t[23] & 6)
460. sprintf(resultado + strlen(resultado), "TCP\n");
461. **else** **if**(t[23] & 17)
462. sprintf(resultado + strlen(resultado), "UDP\n");
463. **else**
464. sprintf(resultado + strlen(resultado), "Otro\n");
466. sprintf(resultado + strlen(resultado), "Checksum Trama: %4x\n", (t[24] << 8) + t[25]);
467. unsigned **short** checksum = 0;
468. unsigned **short** temp = 0;
470. **for**(**int** j = 128; j > 0; j >>= 1)
471. **if**(t[14] & j)
472. checksum = (checksum << 1) | 1;
473. **else**
474. checksum <<= 1;
475. **for**(**int** j = 128; j > 0; j >>= 1)
476. **if**(t[15] & j)
477. checksum = (checksum << 1) | 1;
478. **else**
479. checksum <<= 1;
481. //sprintf(resultado + strlen(resultado), "Checksum: %4x\n", checksum);
482. //No calculamos los t[24] ni t[25] porque corresponden al checksum previamente calculado
483. **for**(**int** i = 16; i < 14 + ((t[14] & 15) \* 4); i++)
484. {
485. **if**(i != 24)
486. {
487. **for**(**int** j = 128; j > 0; j >>= 1)
488. **if**(t[i] & j)
489. temp = (temp << 1) | 1;
490. **else**
491. temp <<= 1;
492. i++;
493. **for**(**int** j = 128; j > 0; j >>= 1)
494. **if**(t[i] & j)
495. temp = (temp << 1) | 1;
496. **else**
497. temp <<= 1;
499. **if**(((checksum >> 12) + (temp >> 12)) > 15)
500. checksum += 1;
501. //sprintf(resultado + strlen(resultado), "Temp: %4x\n", temp);
502. checksum += temp;
503. //sprintf(resultado + strlen(resultado), "Checksum: %4x\n", checksum);
504. //sprintf(resultado + strlen(resultado), "\n");
505. }
506. **else**
507. i++;
508. }
509. checksum = ~checksum;
510. sprintf(resultado + strlen(resultado), "Checksum Calculado: %4x\n", checksum);
511. **if**(checksum != ((t[24] << 8) + t[25]))
512. sprintf(resultado + strlen(resultado), "Error! :(\n");
513. **else**
514. sprintf(resultado + strlen(resultado), "Correcto! :)\n");
516. sprintf(resultado + strlen(resultado), "IP Origen: ");
517. **for**(**int** i = 26; i < 26 + (t[14] >> 4); i++)
518. {
519. **if**(i == (25 + (t[14] >> 4)))
520. sprintf(resultado + strlen(resultado), "%d", t[i]);
521. **else**
522. sprintf(resultado + strlen(resultado), "%d.", t[i]);
523. }
524. sprintf(resultado + strlen(resultado), "\nIP Destino: ");
525. **for**(**int** i = 30; i < 30 + (t[14] >> 4); i++)
526. {
527. **if**(i == (29 + (t[14] >> 4)))
528. sprintf(resultado + strlen(resultado), "%d", t[i]);
529. **else**
530. sprintf(resultado + strlen(resultado), "%d.", t[i]);
531. }
532. **if**(((t[14] & 15) \* 4) == 20)
533. sprintf(resultado + strlen(resultado), "\nNo hay opciones");
534. **else**
535. {
536. sprintf(resultado + strlen(resultado), "\nOpciones: ");
537. **for**(**int** i = 34; i < 34 + (((t[14] & 15) \* 4) - 20); i++)
538. {
539. **if**(i == 33 + (((t[14] & 15) \* 4) - 20))
540. sprintf(resultado + strlen(resultado), "%.2X", t[i]);
541. **else**
542. sprintf(resultado + strlen(resultado), "%.2X:", t[i]);
543. }
544. }
545. **return** 1;
546. }
547. **else** **if**(ToT == 2054) //ARP
548. {
549. sprintf(resultado, "ARP\n");
550. **switch**(t[15])
551. {
552. **case** 1:
553. sprintf(resultado + strlen(resultado), "HW: Ethernet\n");
554. **break**;
555. **case** 6:
556. sprintf(resultado + strlen(resultado), "HW: Token Ring\n");
557. **break**;
558. **case** 15:
559. sprintf(resultado + strlen(resultado), "HW: Frame Relay\n");
560. **break**;
561. **case** 16:
562. sprintf(resultado + strlen(resultado), "HW: Atm\n");
563. **break**;
564. }
565. **if**(t[16] == 8)
566. sprintf(resultado + strlen(resultado), "Protocolo: IP\n");
567. **if**(t[18] == 6)
568. sprintf(resultado + strlen(resultado), "Tamanio de la direccion HW: 6 bytes\n");
569. **if**(t[19] == 4)
570. sprintf(resultado + strlen(resultado), "Tamanio de la direccion de Protocolo: 4 bytes\n");
571. **else** **if**(t[19] == 6)
572. sprintf(resultado + strlen(resultado), "Tamanio de la direccion de Protocolo: 6 bytes\n");
574. **switch**(t[21])
575. {
576. **case** 1:
577. sprintf(resultado + strlen(resultado), "Op Code: ARP request\n");
578. **break**;
579. **case** 2:
580. sprintf(resultado + strlen(resultado), "Op Code: ARP response\n");
581. **break**;
582. **case** 8:
583. sprintf(resultado + strlen(resultado), "Op Code: Inverse ARP request\n");
584. **break**;
585. **case** 9:
586. sprintf(resultado + strlen(resultado), "Op Code: Inverse ARP response\n");
587. **break**;
588. }
590. sprintf(resultado + strlen(resultado), "Sender HW address: ");
591. **for**(**BYTE** i = 0; i < 6; i++)
592. {
593. **if**(i == 5)
594. sprintf(resultado + strlen(resultado), "%.2x", t[22 + i]);
595. **else**
596. sprintf(resultado + strlen(resultado), "%.2x:", t[22 + i]);
597. }
598. sprintf(resultado + strlen(resultado), "\nSender Protocol address: ");
599. **for**(**BYTE** i = 0; i < 4; i++)
600. {
601. **if**(i == 3)
602. sprintf(resultado + strlen(resultado), "%d", t[22 + i + 6]);
603. **else**
604. sprintf(resultado + strlen(resultado), "%d.", t[22 + i + 6]);
606. }
607. sprintf(resultado + strlen(resultado), "\nTarget HW address: ");
608. **for**(**BYTE** i = 0; i < 6; i++)
609. {
610. **if**(i == 5)
611. sprintf(resultado + strlen(resultado), "%.2x", t[22 + i + 6 + 4]);
612. **else**
613. sprintf(resultado + strlen(resultado), "%.2x:", t[22 + i + 6 + 4]);
614. }
615. sprintf(resultado + strlen(resultado), "\nTarget Protocol address: ");
616. **for**(**BYTE** i = 0; i < 4; i++)
617. {
618. **if**(i == 3)
619. sprintf(resultado + strlen(resultado), "%d", t[22 + i + 6 + 4 + 6]);
620. **else**
621. sprintf(resultado + strlen(resultado), "%d.", t[22 + i + 6 + 4 + 6]);
622. }
624. **return** 1;
625. }
627. **return** 0;
628. }