

Instituto Politécnico Nacional

Escuela Superior de Computo

Redes de computadoras

“Analizador de Tramas”

Versión 1.- LLC

Por :

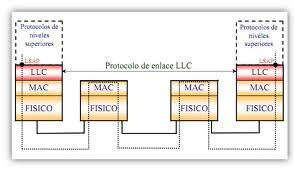
Ortiz Meraz Isaac Baruch

Profesor:

M. en C. NIDIA ASUNCIÓN CORTEZ DURANTE

Abril 2020

**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.

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

*Interfaz LLC – MAC:* Específica los servicios que la subcapa de LLC requiere de la subcapa MAC, independientemente de la topología de la subred y del tipo de acceso al medio.

*Interfaz LLC –* Capa de Red Modelo OSI: Especifica los servicios que la Capa de Red Modelo OSI obtiene de la Capa de Enlace Modelo OSI, independientemente de su configuración.

**Proceso de enmascaramiento**

La cabecera LLC incluye dos campos de dirección adicionales de 8 bit, llamados service access points or SAPs en terminología OSI; cuando la fuente y el destino SAP son puestos al valor 0xAA, el servicio SNAP es requerido. La cabecera SNAP permite usar valores EtherType con todos los protocolos IEEE 802, así como usar protocolos de espacio de ID privados. En IEEE 802.3x-1997, el estándar IEEE Ethernet fue modificado explícitamente para permitir el uso del campo de 16-bit después de la dirección MAC para utilizarlo como un campo de longitud o de tipo.

Estructura

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Delimitador | Dirección | Control | Información | FCS | Delimitador |
| 8 bits | 8 bits extensible | 8 o 16 bits | variable | 16 o 32 | 8 |

I: Información

1 2 3 4 5 6 7 8

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | N(S) | P/F | N(R) |

S: Supervisión

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 0 | S | P/F | N(R) |

U: No numerada

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | M | P/F | M |

N(S) = Número de secuencia enviado

N(R) = Número de secuencia recibido

S = Bits de función supervisora

M = Bits de función no numerada

P/F = Bit de sondeo/fin

Información

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | N(S) | P/F | N(R) |

Supervisión

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | S | 0 | 0 | 0 | 0 | P/F | N(R) |

Para determinar el tipo o tamaño de la trama nos basamos en 3 valores, primero tenemos el valor de los bytes 12 y 13 de la trama, que si recorremos el byte 12 2 espacios a la izquierda y le sumamos el 13 nos dará un número, con este número comparamos con los valores 1500, 2048 y 2054, si el número es menor a 1500, solo se está dando el tamaño en bytes, si el número es igual a 2048 nos dice que es IP y si el número es igual a 2054 nos dice que es ARP.

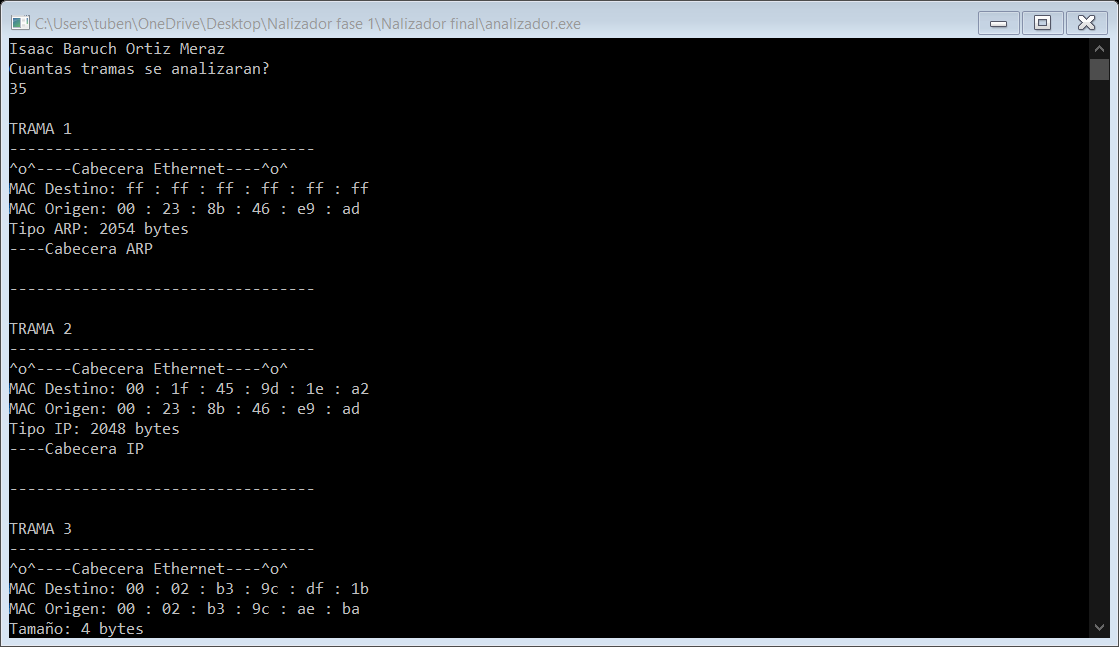
Ahora para determinar si tenemos una trama de información supervisión o no numerada tenemos que hacer una máscara con el byte 16 y el 3 en binario que es 11, esto nos dará 00, 01, 10 o 11 como resultado, donde con la estructura antes podemos saber que 00 y 10 es una trama de información (T-I), 01 es una trama de supervisión (T-S) y 11 corresponde a una trama no numerada (T-U).

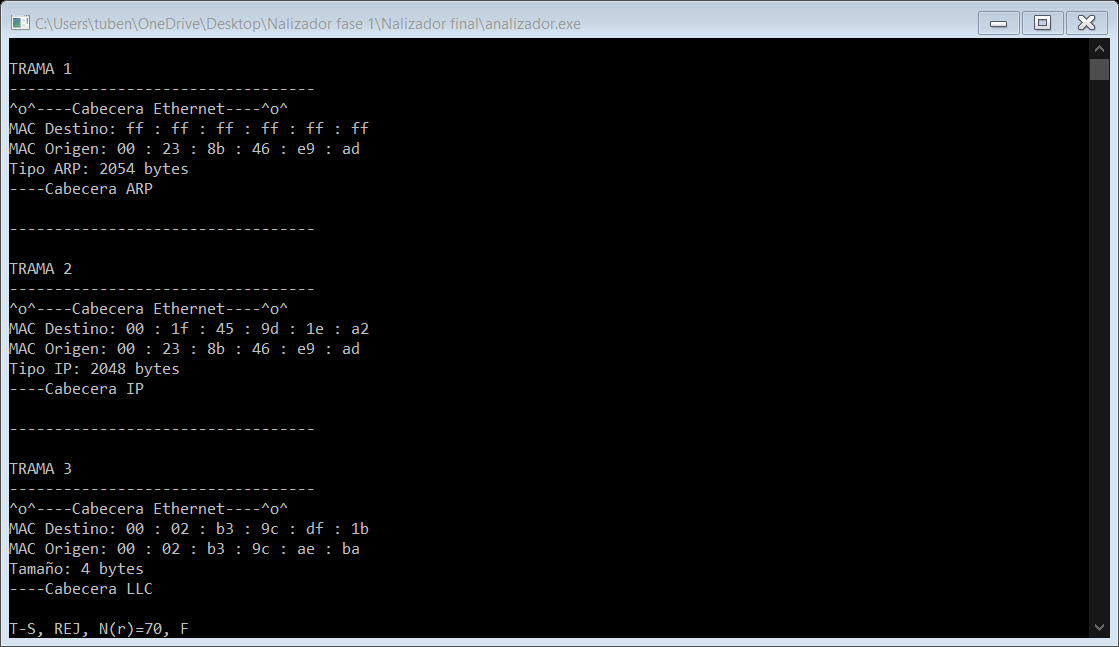
Para obtener el valor de los bits S y M que corresponden a trama de Supervisión y No numerada nos tenemos que fijar en diferentes bytes.

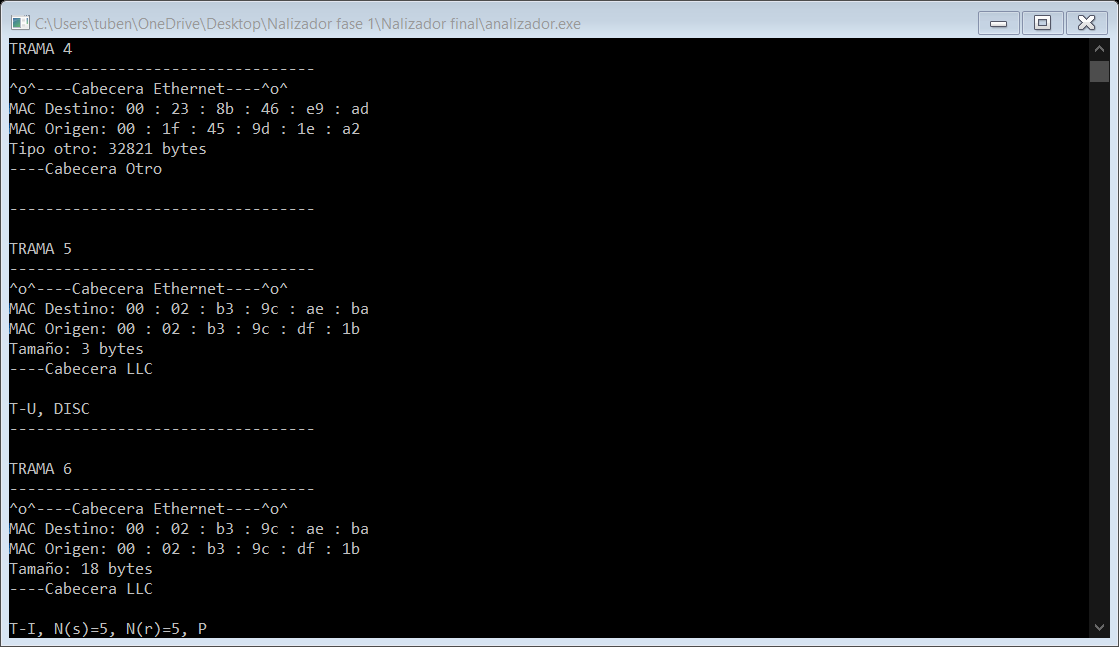
Para el primer caso tenemos que fijarnos en el byte 16 de la trama recorrido 2 bits a la derecha y hacer una máscara con 3 (11), lo que nos dará de nuevo 00, 01, 10 o 1 que corresponden a RR, RNR, REJ y SREJ.

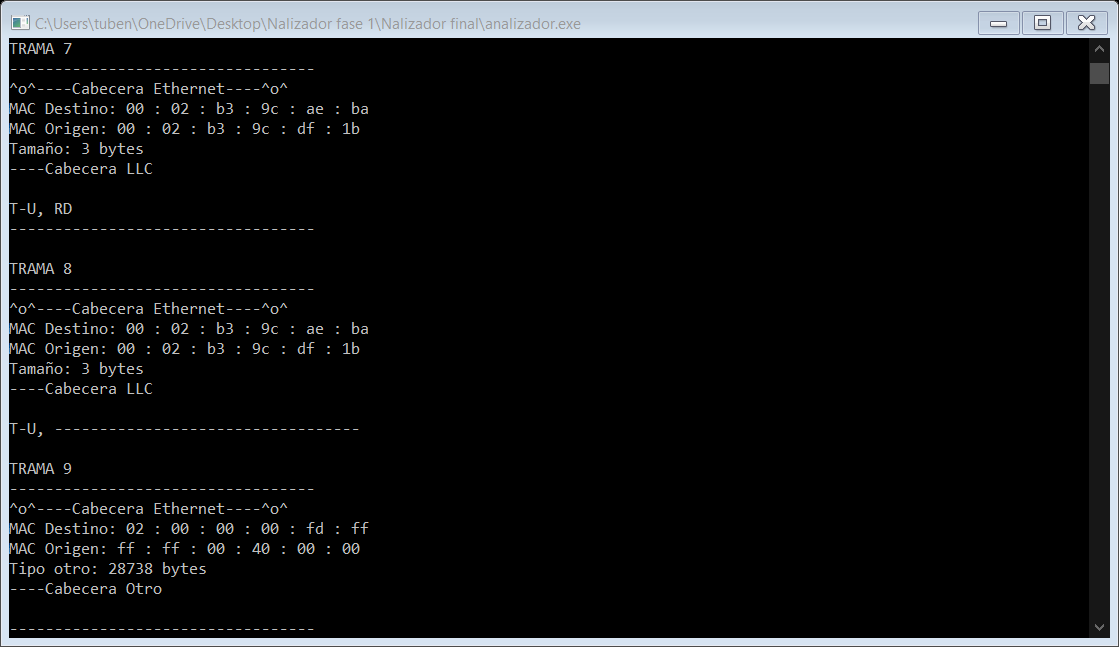
En el segundo caso hacemos una máscara con el byte 16 y 16 (10000) pero en este caso, el valor que nos arrojará no es suficiente ya que cambia el significado dependiendo de si es orden o respuesta, para saber esto solo necesitamos comparar el byte 15 con 1 para saber si el bit p/f es 0 o es 1 y en caso de ser 1 vamos ahora a comparar el byte 16 recorrido 2 bits a la derecha y 3 (11) o el byte 16 recorrido 3 bits a la derecha y 28 (11100) y dependiendo del resultado y si es de orden o de respuesta nos dará la respuesta correspondiente a nuestros arreglos (s[ ], UR[ ] y UC[ ]).

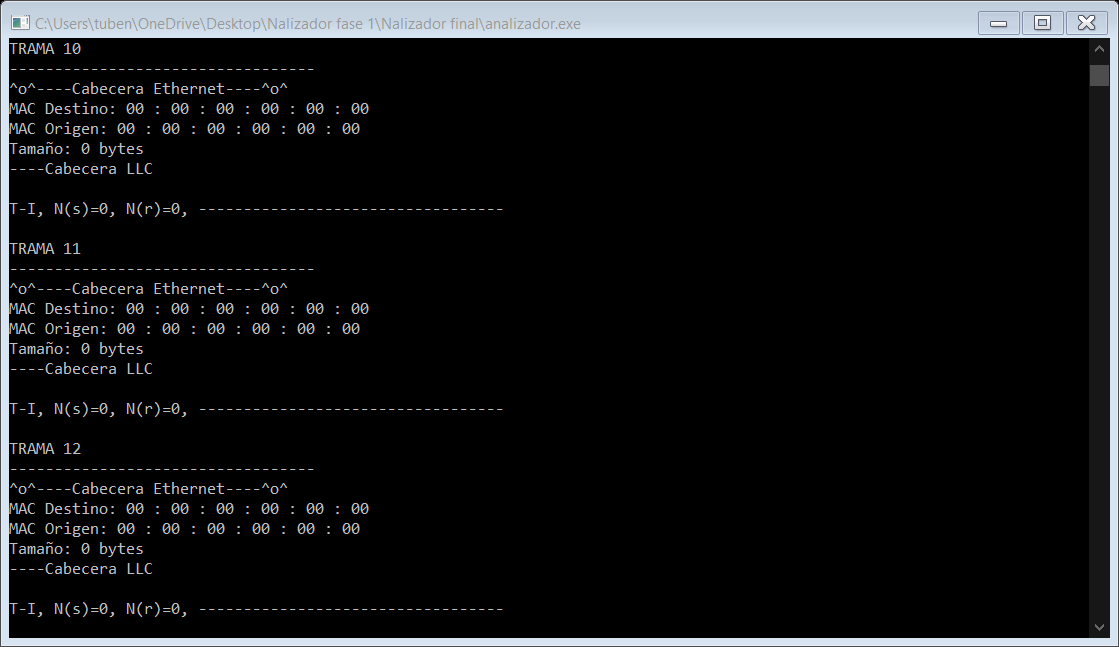
**Capturas**

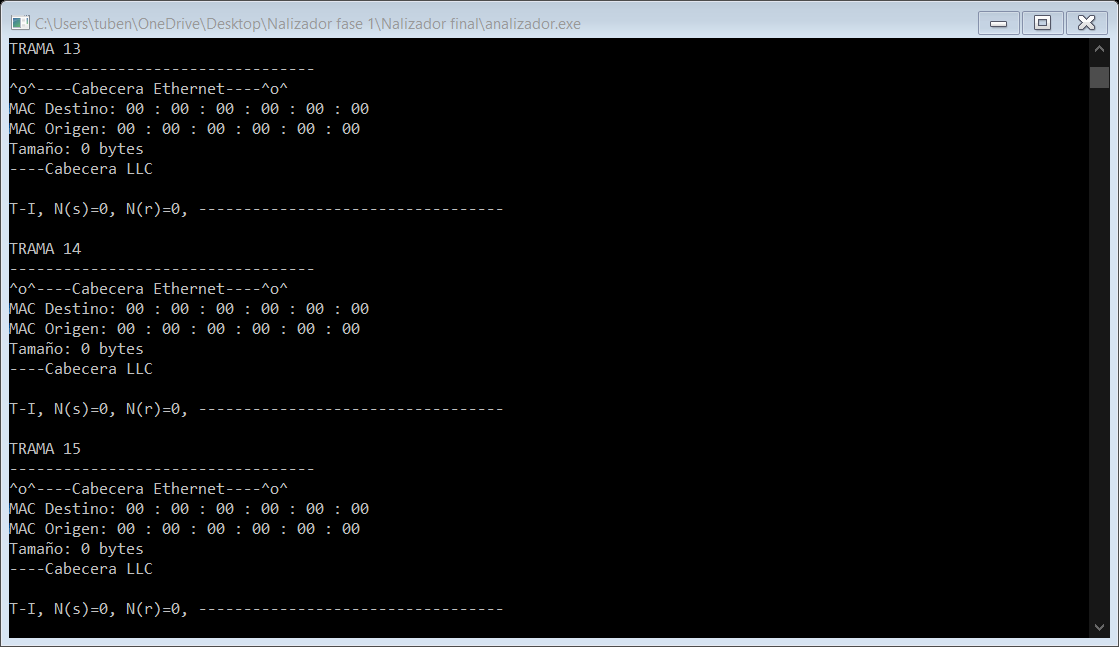
****

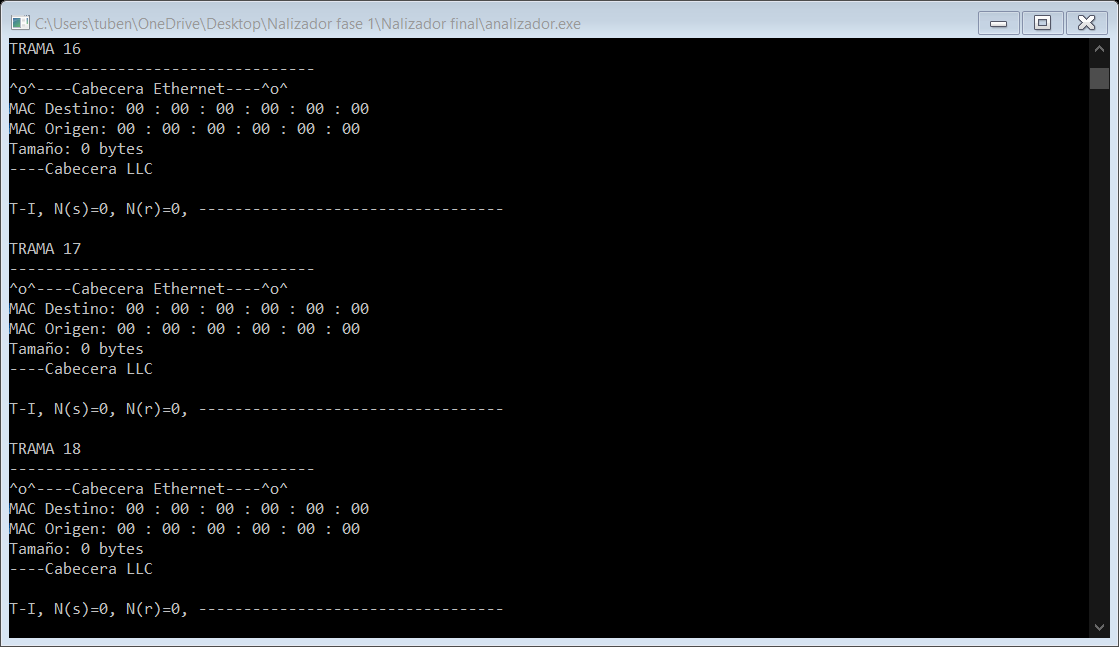
****

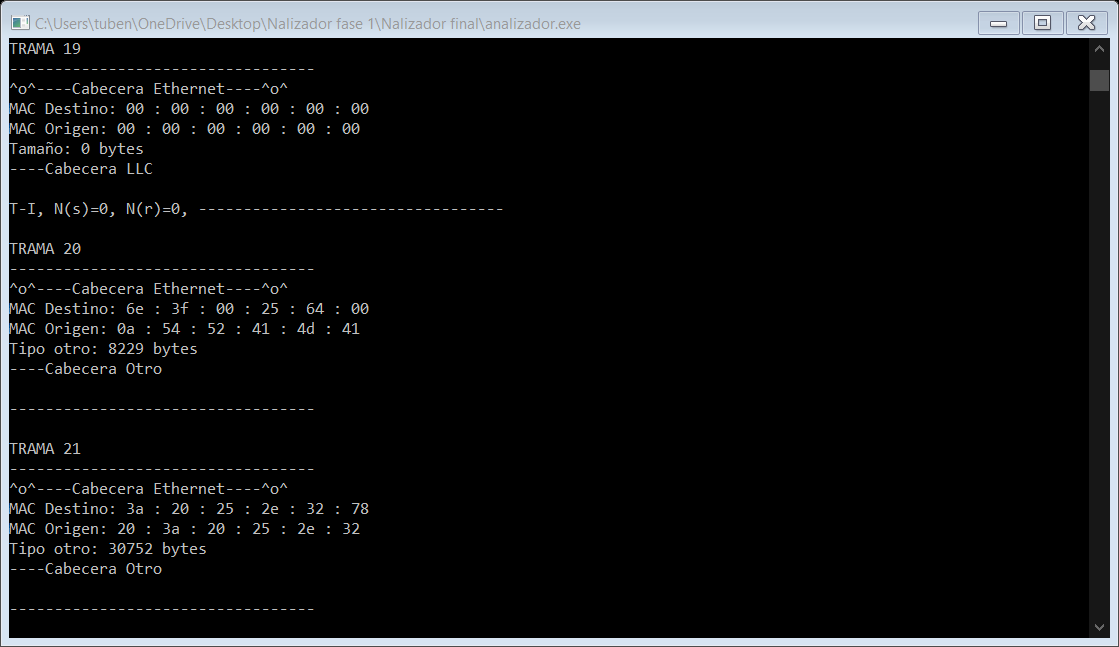
****

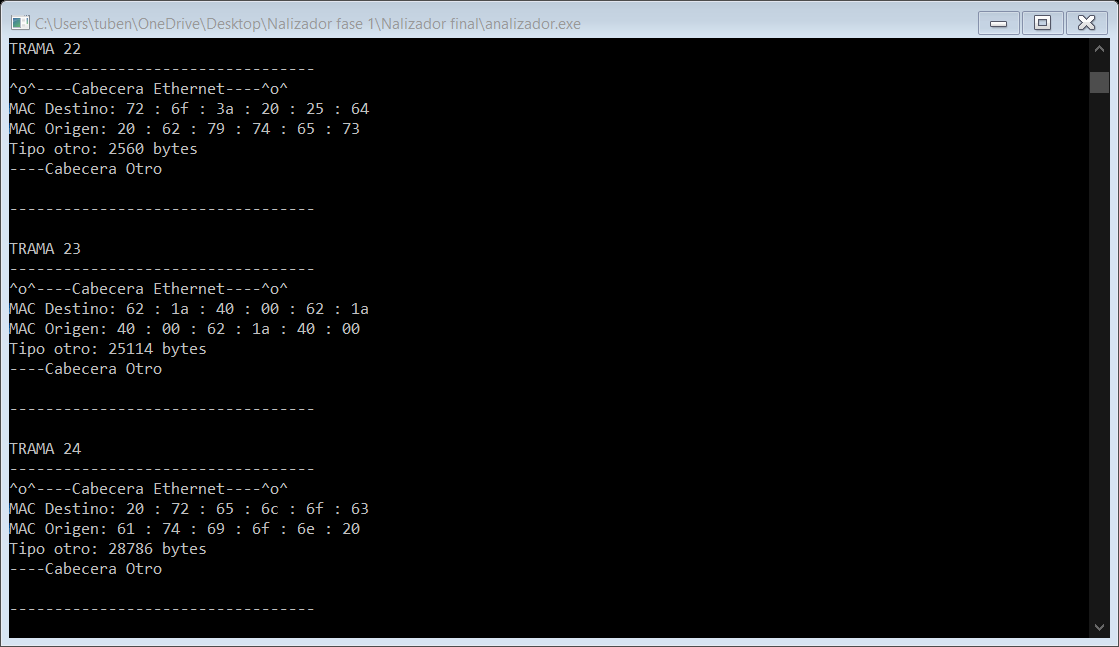
****

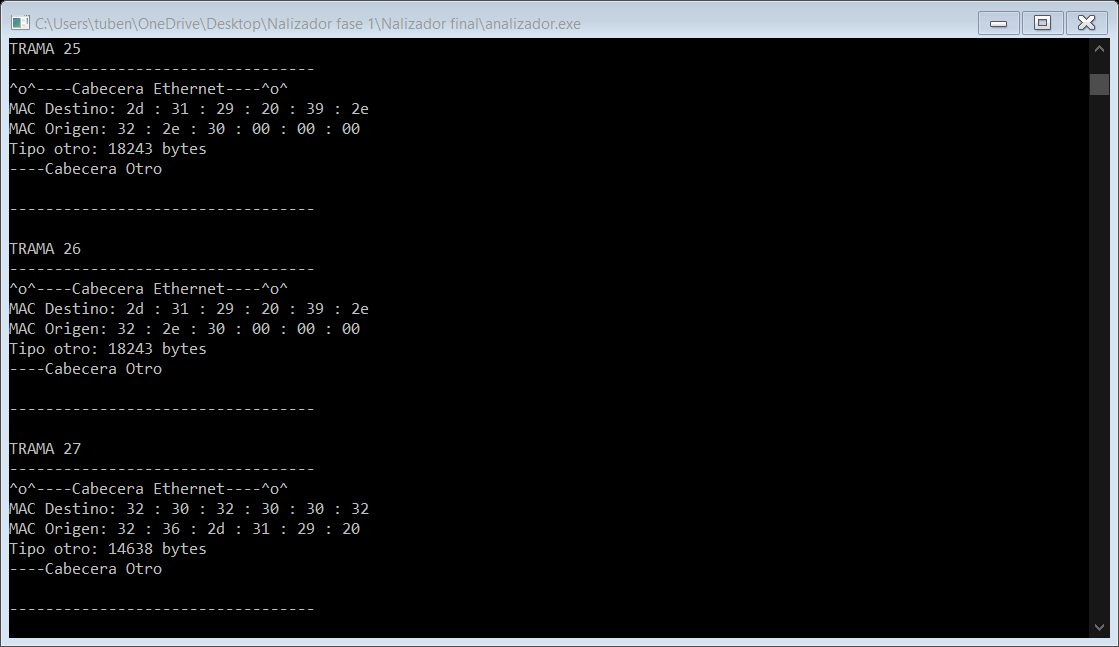
****

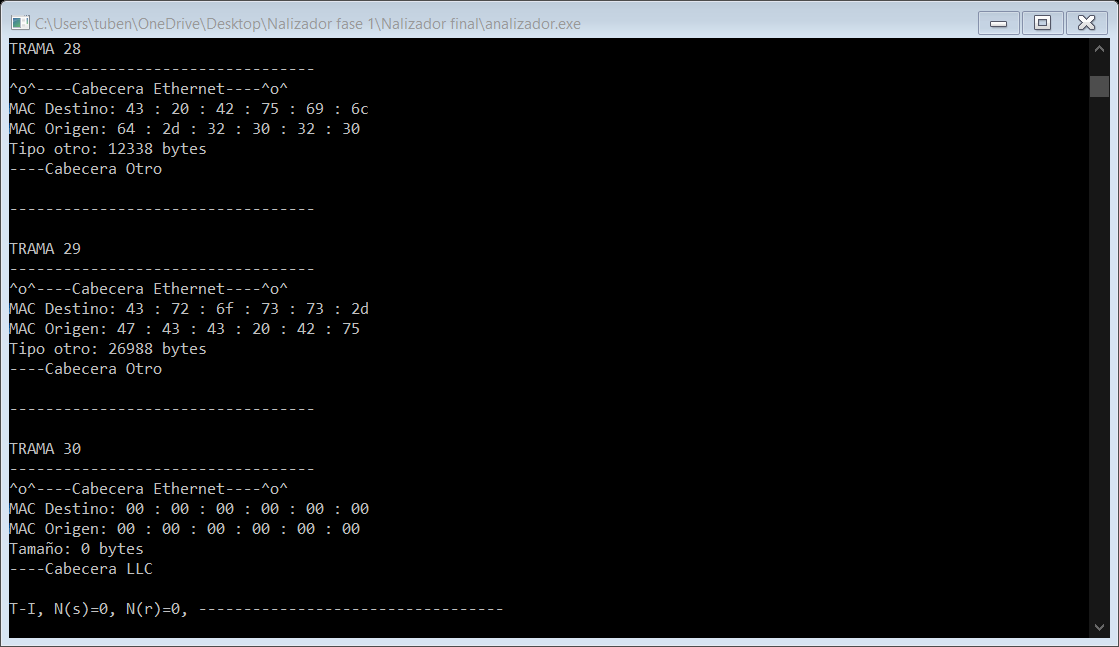
****

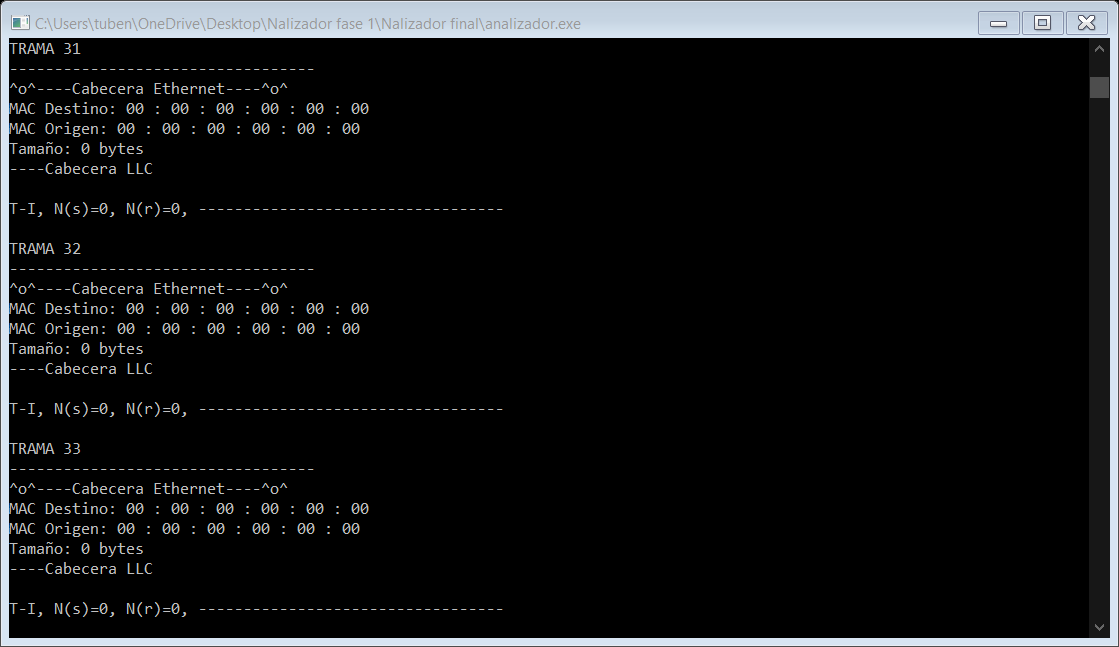
****

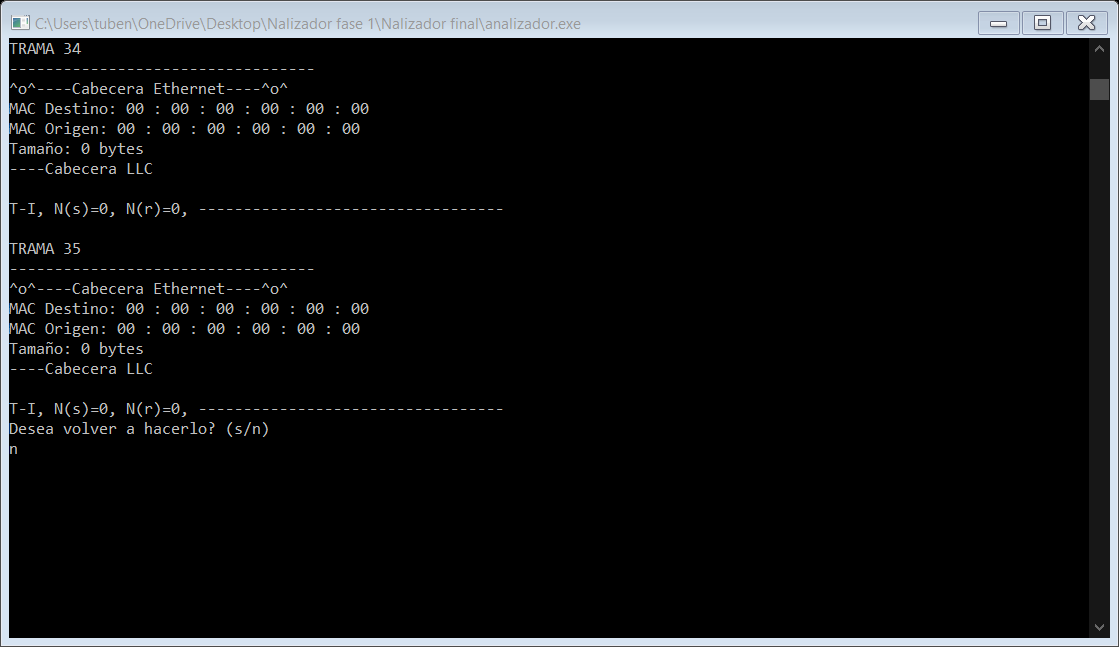
****

****

****

****

****

****

**Código**

1. #include <stdio.h>
3. **void** analiza(unsigned **char** []);
4. unsigned **char** S[][5]={"RR","RNR","REJ","SREJ"};
5. unsigned **char** MC[][6]={"UI","SIM","","SARM","UP","","","SABM","DISC","","","SARME","","","","SABME",
6. "SNRM","","","RSET","","","","XID","","","","SNRME","","","","SNRM"};
7. unsigned **char** MR[][5]={"UI","RIM","","DM","","","","","RD","","","","UA","TEST","","",""
8. "","FRMR","","","","","","XID","","","","","TEST","","",""};
9. unsigned **char** T[][200]={{ //t1
10. 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x08, 0x06, 0x00, 0x10,
11. 0x08, 0x00, 0x06, 0x04, 0x00, 0x04, 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x94, 0xcc, 0x39, 0xcb,
12. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x94, 0xcc, 0x39, 0xfe
13. },
14. {  //t2
15. 0x00, 0x1f, 0x45, 0x9d, 0x1e, 0xa2, 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x08, 0x00, 0x46, 0x00,
16. 0x80, 0x42, 0x04, 0x55, 0x34, 0x11, 0x80, 0x11, 0x6b, 0xf0, 0x94, 0xcc, 0x39, 0xcb, 0x94, 0xcc,
17. 0x67, 0x02, 0xaa, 0xbb, 0xcc, 0xdd, 0x04, 0x0c, 0x00, 0x35, 0x00, 0x2e, 0x85, 0x7c, 0xe2, 0x1a,
18. 0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x77, 0x77, 0x77, 0x03, 0x69,
19. 0x73, 0x63, 0x05, 0x65, 0x73, 0x63, 0x6f, 0x6d, 0x03, 0x69, 0x70, 0x6e, 0x02, 0x6d, 0x78, 0x00,
20. 0x00, 0x1c, 0x00, 0x01
21. },
22. {  //t3
23. 0x00, 0x02, 0xb3, 0x9c, 0xdf, 0x1b, 0x00, 0x02, 0xb3, 0x9c, 0xae, 0xba, 0x00, 0x04, 0xf0, 0xf1,
24. 0x09, 0x8d, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
25. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
26. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0x9b, 0x6d
27. },
29. {  //t4
30. 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x00, 0x1f, 0x45, 0x9d, 0x1e, 0xa2, 0x80, 0x35, 0x00, 0x01,
31. 0x08, 0x00, 0x06, 0x04, 0x00, 0x03, 0x00, 0x1f, 0x45, 0x9d, 0x1e, 0xa2, 0x94, 0xcc, 0x3a, 0xe1,
32. 0x00, 0x23, 0x8b, 0x46, 0xe9, 0xad, 0x94, 0xcc, 0x39, 0xcb, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
33. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xd8, 0xee, 0xdf, 0xb0
34. },
35. { //t5
36. 0x00, 0x02, 0xb3, 0x9c, 0xae, 0xba, 0x00, 0x02, 0xb3, 0x9c, 0xdf, 0x1b, 0x00, 0x03, 0xf0, 0xf0,
37. 0x53, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
38. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
39. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x43, 0x05, 0x90, 0x6d
40. },
42. {  //T6
43. 0x00, 0x02, 0xb3, 0x9c, 0xae, 0xba, 0x00, 0x02, 0xb3, 0x9c, 0xdf, 0x1b, 0x00, 0x12, 0xf0, 0xf0,
44. 0x0a, 0x0b, 0x0e, 0x00, 0xff, 0xef, 0x14, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x7f, 0x23,
45. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
46. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x99, 0x98, 0x6d
47. },
48. { //t7
49. 0x00, 0x02, 0xb3, 0x9c, 0xae, 0xba, 0x00, 0x02, 0xb3, 0x9c, 0xdf, 0x1b, 0x00, 0x03, 0xf0, 0xf1,
50. 0x53, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
51. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
52. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x43, 0x05, 0x90, 0x6d
53. },
54. { //t8
55. 0x00, 0x02, 0xb3, 0x9c, 0xae, 0xba, 0x00, 0x02, 0xb3, 0x9c, 0xdf, 0x1b, 0x00, 0x03, 0xf0, 0xf0,
56. 0x43, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
57. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
58. 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x43, 0x05, 0x90, 0x6d
59. }
60. };
61. unsigned **short** **int** ToT;
62. unsigned **char** i;
64. **int** main(){
65. unsigned **char** respuesta;
66. unsigned **char** opc1;
67. **do**{
68. printf("Ortiz Meraz Isaac Baruch\n");
69. printf("Cuantas tramas se analizaran?\n");
70. scanf("%d", &opc1);
71. **for**(i=0;i<opc1;i++){
72. printf("\nTRAMA %d \n", i+1);
73. printf("----------------------------------\n");
74. analiza(T[i]);
75. printf("----------------------------------\n");
76. }
77. printf("Desea volver a hacerlo? (s/n)\n");
78. fflush(stdin);
79. scanf("%s", &respuesta);
80. }**while** (respuesta != 'n');
81. }

84. **void** analiza(unsigned **char** T[]){
86. ToT = (T[12]<<8|T[13]);
88. printf("^o^----Cabecera Ethernet----^o^\n");
89. printf("MAC Destino: %.2x : %.2x : %.2x : %.2x : %.2x : %.2x\n", T[0], T[1], T[2], T[3], T[4], T[5]);
90. printf("MAC Origen: %.2x : %.2x : %.2x : %.2x : %.2x : %.2x\n", T[6], T[7], T[8], T[9], T[10], T[11]);
92. **if**(ToT <= 1500){
93. printf("Tama%co: %d bytes\n",164, ToT);
94. printf("----Cabecera LLC\n\n");
95. **switch** (T[16]&3)
96. {
97. **case** 0:
98. printf("T-I, N(s)=%d, N(r)=%d, ", T[16]>>1, T[17]>>1);
99. **if**(T[17]&1){
100. **if**(T[15]&1)
101. printf("F\n");
102. **else**
103. printf("P\n");
104. }
105. **break**;
106. **case** 1:
107. printf("T-S, %s, N(r)=%d, ", S[(T[16]>>2)&3], T[17]>>1);
108. **if**(T[17]&1){
109. **if**(T[15]&1)
110. printf("F\n");
111. **else**
112. printf("P\n");
113. }
114. **break**;
115. **case** 2:
116. printf("T-I, N(s)=%d, N(r)=%d, ", T[16]>>1, T[17]>>1);
117. **if**(T[17]&1){
118. **if**(T[15]&1)
119. printf("F\n");
120. **else**
121. printf("P\n");
122. }
123. **break**;
124. **case** 3:
125. printf("T-U, ");
126. **if**(T[16]&16){
127. **if**(T[15]&1)
128. printf("%s\n", MR[((T[16]>>2)&3)|((T[16]>>3)&28)]);
129. **else**
130. printf("%s\n", MC[((T[16]>>2)&3)|((T[16]>>3)&28)]);
132. **if** (((T[16]>>2)&3)|(((T[16]>>3)&28)==11) || (((T[16]>>2)&3)|(((T[16]>>3)&28)==15)) | ((T[16]>>2)&3)|((T[16]>>3)&28)==27)
133. printf("\n");
134. }
135. **break**;
136. }
137. }**else**{
138. **if**(ToT <= 2048){
139. printf("Tipo IP: %d bytes\n", ToT);
140. printf("----Cabecera IP\n\n");
141. }**else**{
142. **if**(ToT <= 2054){
143. printf("Tipo ARP: %d bytes\n", ToT);
144. printf("----Cabecera ARP\n\n");
145. }**else**{
146. printf("Tipo otro: %d bytes\n", ToT);
147. printf("----Cabecera Otro\n\n");
148. }
149. }
150. }
152. }

**Referencia**

* 1. Apuntes de clase
  2. https://luisagabph.wordpress.com/2013/04/02/protocolo-llc/