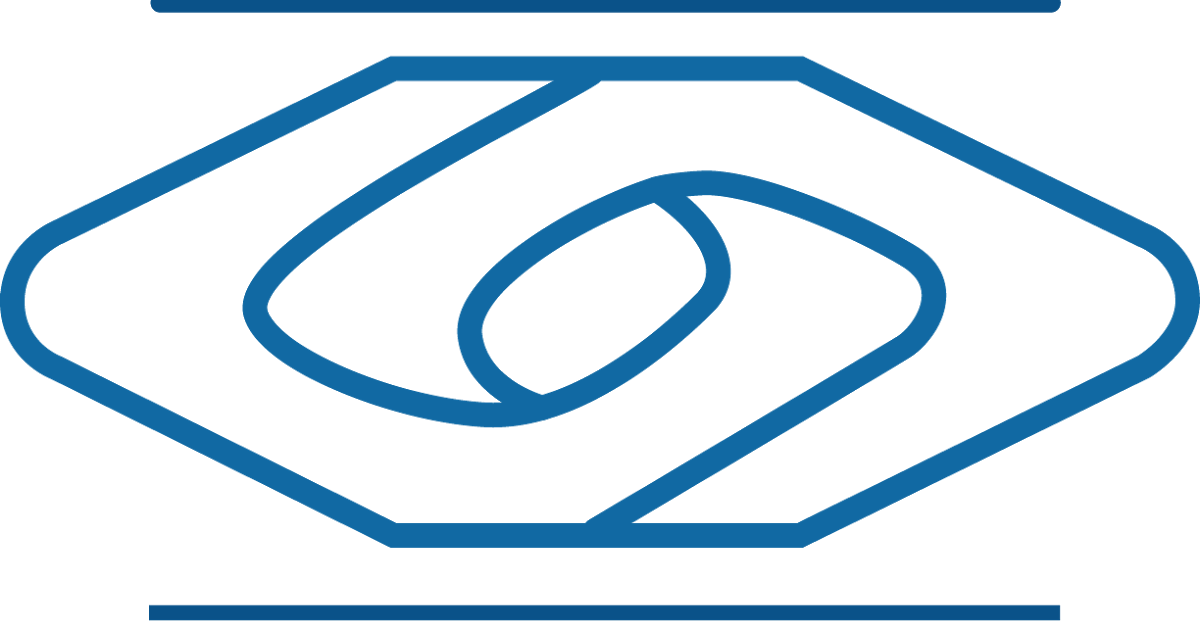
**Instituto Politécnico Nacional.**

**Escuela Superior de Cómputo.**







Materia: Redes de Computadoras.

Tema: Análisis de Tramas Ethernet.

Profesor: Axel Ernesto Moreno Cervantes.

Integrantes:

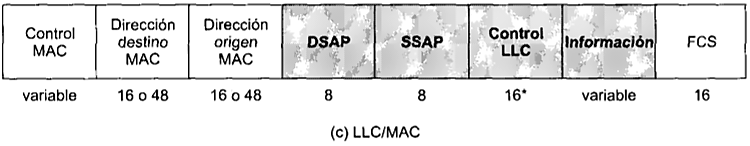
Rojas Alvarado Luis Enrique.

Miranda Sandoval Mario Alberto.

Grupo: 2CM10.

## **Introducción**

El control de enlace lógico o LLC (Logical link control) por sus siglas en inglés pertenece a la familia de estándares IEEE 802 para el control de enlace de datos. LLC utiliza algunas características de HDLC además de que incluye algunas que no tiene el anterior. En LLC las funciones para controlar el enlace se dividen en dos capas la capa de control de acceso al medio y la capa LLC que funciona por encima de la capa MAC.



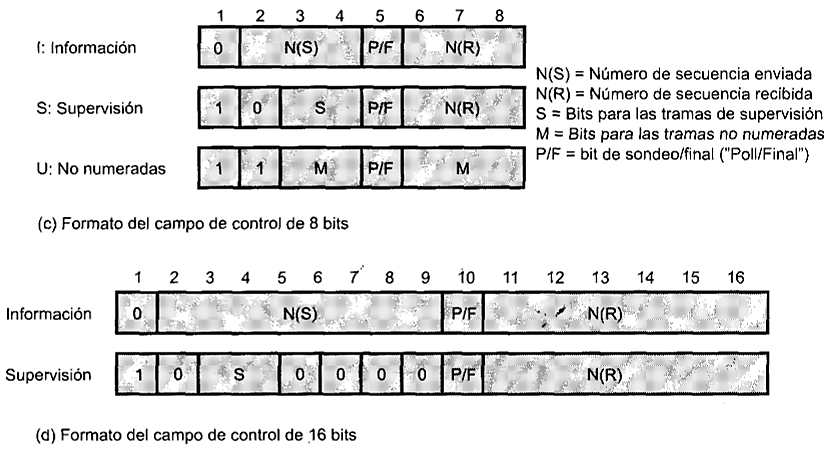
La MAC incluye las direcciones origen y destino para identificar a los dispositivos conectados en la LAN. Estas dos direcciones son necesarias ya que en el entorno LAN no existe el concepto de estación primaria o secundaria por lo que el emisor y el receptor deben ser identificados. La detección de errores se realiza en el nivel MAC utilizando CRC de 32 bits.

En la capa LLC hay cuatro campos. Los puntos de acceso al servicio del destino y del origen los cuales identifican al usuario lógico del LLC en los sistemas origen y destino. El campo de control del LLC tiene el campo de control limitado a la utilización de números de secuencia de 7 bits.

En este se definen tres tipos de tramas cada una de ellas con un formato diferente para el campo de control.

* **Tramas de información**. Transportan los datos generados por el usuario. Además, se incluye la información para control ARQ de errores y flujo.
* **Tramas de supervisión**. Proporcionan el mecanismo ARQ cuando la incorporación de las confirmaciones en las tramas de información no es factible.
* **Tramas no numeradas**. Proporcionan funciones complementarias para controlar el enlace.

El primer o los dos primeros bits del campo de control se utilizan para identificar el tipo de trama, además se incluye el bit poll/final.



LLC ofrece tres tipos de servicios. El servicio con modo de conexión, los otros dos son sin conexión y sin conexión confirmado.

**Desarrollo:**

El código para realizar la práctica es el siguiente:

1. **import** java.util.ArrayList;
2. **import** java.util.Date;
3. **import** java.util.List;
4. **import** java.io.\*;
6. **import** org.jnetpcap.Pcap;
7. **import** org.jnetpcap.PcapAddr;
8. **import** org.jnetpcap.PcapIf;
9. **import** org.jnetpcap.packet.PcapPacket;
10. **import** org.jnetpcap.packet.PcapPacketHandler;
11. **import** org.jnetpcap.PcapBpfProgram;

14. **public** **class** Captura {
16. /\*\*
17. \* Main startup method
18. \*
19. \* @param args
20. \*          ignored
21. \*/
22. **static** **int** no=1;
24. **public** **static** **int** ntrama(){
25. **return** no++;
26. }
27. **private** **static** String asString(**final** **byte**[] mac) {
28. **final** StringBuilder buf = **new** StringBuilder();
29. **for** (**byte** b : mac) {
30. **if** (buf.length() != 0) {
31. buf.append(':');
32. }
33. **if** (b >= 0 && b < 16) {
34. buf.append('0');
35. }
36. buf.append(Integer.toHexString((b < 0) ? b + 256 : b).toUpperCase());
37. }
39. **return** buf.toString();
40. }
42. **public** **static** **void** main(String[] args) {
43. Pcap pcap=**null**;
44. **try**{
45. BufferedReader br = **new** BufferedReader(**new** InputStreamReader(System.in));
46. List<PcapIf> alldevs = **new** ArrayList<PcapIf>(); // Will be filled with NICs
47. StringBuilder errbuf = **new** StringBuilder(); // For any error msgs
48. System.out.println("[0]-->Realizar captura de paquetes al vuelo");
49. System.out.println("[1]-->Cargar traza de captura desde archivo");
50. System.out.print("\nElige una de las opciones:");
51. **int** opcion = Integer.parseInt(br.readLine());
52. **if** (opcion==1){
54. /////////////////////////lee archivo//////////////////////////
55. //String fname = "archivo.pcap";
56. String fname = "paquetes3.pcap";
57. pcap = Pcap.openOffline(fname, errbuf);
58. **if** (pcap == **null**) {
59. System.err.printf("Error while opening device for capture: "+ errbuf.toString());
60. **return**;
61. }//if
62. } **else** **if**(opcion==0){
63. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
64. \* First get a list of devices on this system
65. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
66. **int** r = Pcap.findAllDevs(alldevs, errbuf);
67. **if** (r == Pcap.NOT\_OK || alldevs.isEmpty()) {
68. System.err.printf("Can't read list of devices, error is %s", errbuf
69. .toString());
70. **return**;
71. }
73. System.out.println("Network devices found:");
75. **int** i = 0;
76. **for** (PcapIf device : alldevs) {
77. String description =
78. (device.getDescription() != **null**) ? device.getDescription()
79. : "No description available";
80. **final** **byte**[] mac = device.getHardwareAddress();
81. String dir\_mac = (mac==**null**)?"No tiene direccion MAC":asString(mac);
82. System.out.printf("#%d: %s [%s] MAC:[%s]\n", i++, device.getName(), description, dir\_mac);
83. List<PcapAddr> direcciones = device.getAddresses();
84. **for**(PcapAddr direccion:direcciones){
85. System.out.println(direccion.getAddr().toString());
86. }//foreach
88. }//for
90. System.out.print("\nEscribe el número de interfaz a utilizar:");
91. **int** interfaz = Integer.parseInt(br.readLine());
92. PcapIf device = alldevs.get(interfaz); // We know we have atleast 1 device
93. System.out
94. .printf("\nChoosing '%s' on your behalf:\n",
95. (device.getDescription() != **null**) ? device.getDescription()
96. : device.getName());
98. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
99. \* Second we open up the selected device
100. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
101. /\*"snaplen" is short for 'snapshot length', as it refers to the amount of actual data captured from each packet passing through the specified network interface.
102. 64\*1024 = 65536 bytes; campo len en Ethernet(16 bits) tam máx de trama \*/
104. **int** snaplen = 64 \* 1024;           // Capture all packets, no trucation
105. **int** flags = Pcap.MODE\_PROMISCUOUS; // capture all packets
106. **int** timeout = 10 \* 1000;           // 10 seconds in millis

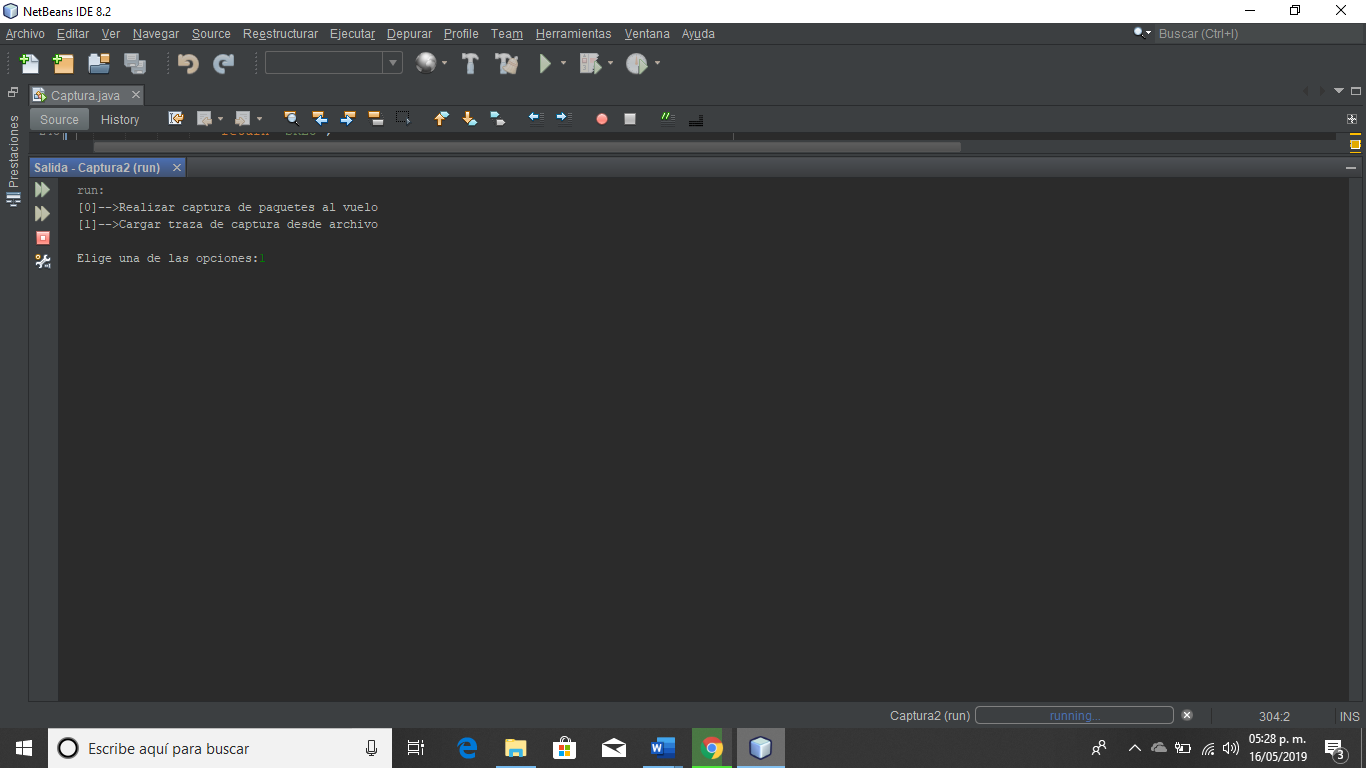
109. pcap = Pcap.openLive(device.getName(), snaplen, flags, timeout, errbuf);
111. **if** (pcap == **null**) {
112. System.err.printf("Error while opening device for capture: "
113. + errbuf.toString());
114. **return**;
115. }//if
117. /\*\*\*\*\*\*\*\*F I L T R O\*\*\*\*\*\*\*\*/
118. PcapBpfProgram filter = **new** PcapBpfProgram();
119. String expression =""; // "port 80";
120. **int** optimize = 0; // 1 means true, 0 means false
121. **int** netmask = 0;
122. **int** r2 = pcap.compile(filter, expression, optimize, netmask);
123. **if** (r2 != Pcap.OK) {
124. System.out.println("Filter error: " + pcap.getErr());
125. }//if
126. pcap.setFilter(filter);
127. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
128. }//else if
130. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
131. \* Third we create a packet handler which will receive packets from the
132. \* libpcap loop.
133. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
134. PcapPacketHandler<String> jpacketHandler = **new** PcapPacketHandler<String>() {
135. **public** **void** nextPacket(PcapPacket packet, String user) {
136. System.out.printf("\n\n Paquete recibido el %s caplen=%-4d longitud=%-4d %s\n\n",
137. **new** Date(packet.getCaptureHeader().timestampInMillis()),
138. packet.getCaptureHeader().caplen(),  // Length actually captured
139. packet.getCaptureHeader().wirelen(), // Original length
140. user                                 // User supplied object
141. );

144. /\*\*\*\*\*\*Desencapsulado\*\*\*\*\*\*\*\*/
145. **for**(**int** i=0;i<packet.size();i++){
146. System.out.printf("%02X ",packet.getUByte(i));
148. **if**(i%16==15)
149. System.out.println("");
150. }//if
152. System.out.println("\n\nEncabezado: "+ packet.toHexdump());
154. **int** longitud = (packet.getUByte(12)\*256)+packet.getUByte(13);
155. **int** trama=1;
156. **if**(longitud<=1500){
157. **int** PFBit;
158. System.out.printf("\n--->Trama IEEE802.3 %d\n",ntrama());
159. System.out.printf("\nTamaño: %d bytes, Valor: %X\n",longitud,longitud );
160. System.out.printf("|-->MAC Destino: %02X:%02X:%02X:%02X:%02X:%02X",packet.getUByte(0),packet.getUByte(1),packet.getUByte(2),packet.getUByte(3),packet.getUByte(4),packet.getUByte(5));
161. System.out.printf("\n|-->MAC Origen: %02X:%02X:%02X:%02X:%02X:%02X",packet.getUByte(6),packet.getUByte(7),packet.getUByte(8),packet.getUByte(9),packet.getUByte(10),packet.getUByte(11));
162. //System.out.printf("\n |-->DSAP: %02X",packet.getUByte(14));
163. //System.out.println(packet.getUByte(15)& 0x00000001);
164. **int** ssap = packet.getUByte(15);
165. **int** dsap=packet.getUByte(14);
166. String i\_g=((dsap&0x1)==1)?"Grupal":"Individual";
167. String c\_r = ((ssap&0x1)==1)?"Respuesta":"Comando";
168. **int** CRBit=((ssap&0x1)==1)?1:0;
169. System.out.printf("\n|-->SSAP: Valor: %02X   %s\n",packet.getUByte(15), c\_r);
170. System.out.printf("|-->DSAP: Valor: %02X   %s\n",packet.getUByte(14), i\_g);
171. **int** control=packet.getUByte(16);
172. **int** extendido=packet.getUByte(17);
174. **if**(extendido==0){
175. PFBit=((control>>4)&0b1);
176. }**else**{
177. PFBit=extendido&0b1;
178. }
179. String tipo="";
181. **if**((control&0b11)==0b11){
182. System.out.printf("El control es un tipo no numerado %08d\n",Integer.parseInt(Integer.toString(control,2)));
183. tipo=tipoNoEnumerado(CRBit, control, tipo);
184. System.out.printf("Tipo: %s\n", tipo);
185. System.out.printf("P/F: %s=%d\n", POLL\_FINAL[CRBit], PFBit);
186. }**else** **if**((control&0b11)==0b01){
187. **if** (extendido == 0) {
188. System.out.printf("El control es de tipo supervision %08d\n", Integer.parseInt(Integer.toString(control, 2)));
189. System.out.printf("El valor N(R)=%X\n", control>>5);
190. } **else** {
191. System.out.printf("El control es de tipo Supervision %08d %08d\n", Integer.parseInt(Integer.toString(extendido, 2)),Integer.parseInt(Integer.toString(control, 2)));
192. System.out.printf("El valor N(R)=%X\n", extendido>>1);
193. }
194. tipo=supervision(control, tipo);
195. System.out.printf("Tipo: %s\n", tipo);
196. System.out.printf("P/F: %s=%d\n", POLL\_FINAL[CRBit], PFBit);
197. }**else** **if**((control&0b11)==0b0){
198. **if** (extendido == 0) {
199. System.out.printf("El control es de tipo informacion %08d\n", Integer.parseInt(Integer.toString(control, 2)));
200. System.out.printf("El valor N(R)=%X\n", control>>5);
201. System.out.printf("El valor N(S)=%X\n", (control>>1)&0b111);
202. } **else** {
203. System.out.printf("El control es de tipo informacion %08d %08d\n",Integer.parseInt(Integer.toString(extendido, 2)),Integer.parseInt(Integer.toString(control, 2)));
204. System.out.printf("El valor N(R)=%X\n", extendido>>1);
205. System.out.printf("El valor N(S)=%X\n", control>>1);
206. }
207. System.out.printf("P/F: %s=%d\n", POLL\_FINAL[CRBit], PFBit);
208. }**else**
209. System.out.println("Error");
211. System.out.println("");
213. } **else** {
214. //System.out.println("-->Trama ETHERNET numero-->"+cont.getCont());
215. }
216. }
217. };

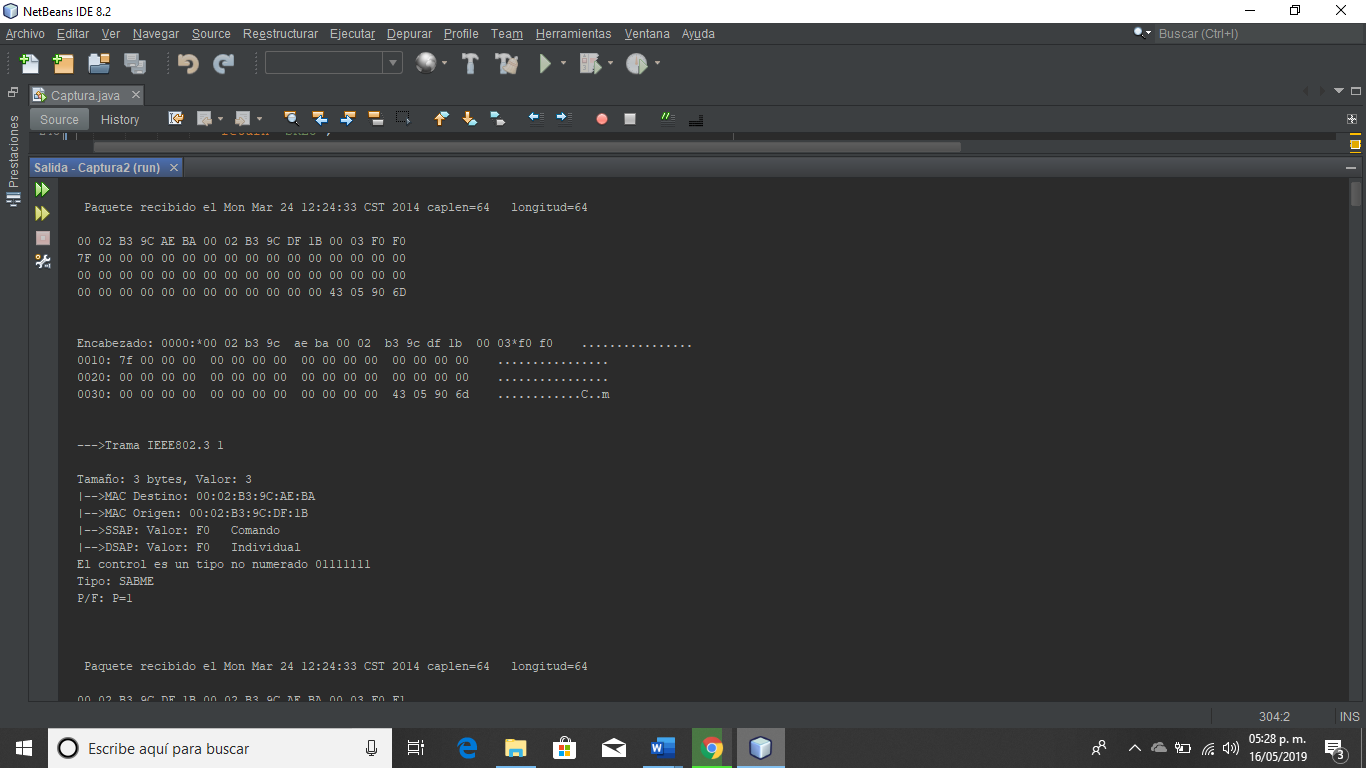
220. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
221. \* Fourth we enter the loop and tell it to capture 10 packets. The loop
222. \* method does a mapping of pcap.datalink() DLT value to JProtocol ID, which
223. \* is needed by JScanner. The scanner scans the packet buffer and decodes
224. \* the headers. The mapping is done automatically, although a variation on
225. \* the loop method exists that allows the programmer to sepecify exactly
226. \* which protocol ID to use as the data link type for this pcap interface.
227. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
228. pcap.loop(-1, jpacketHandler, " ");
230. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
231. \* Last thing to do is close the pcap handle
232. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
233. pcap.close();
234. }**catch**(IOException e){e.printStackTrace();}
235. }
237. **public** **static** String supervision(**int** control, String tipo){
238. **if**(((control>>2)&0b11)==RR){
239. **return** "RR";
240. }**else** **if**(((control>>2)&0b11)==REJ){
241. **return** "REJ";
242. }**else** **if**(((control>>2)&0b11)==RNR){
243. **return** "RNR";
244. }**else** **if**(((control>>2)&0b11)==SREJ){
245. **return** "SREJ";
246. }**else**
247. **return** "";
248. }
250. **public** **static** String tipoNoEnumerado(**int** CRBit, **int** control, String tipo){
251. **if**(CRBit==0){
252. **if**(((control^SNRM)==0) || ((control^SNRM)==16)){
253. tipo="SNRM";
254. }**else** **if**(((control^SNRME)==0) || ((control^SNRME)==16)){
255. tipo="SNRME";
256. }**else** **if**(((control^SARM\_DM)==0) || ((control^SARM\_DM)==16)){
257. tipo="SARM\_DM";
258. }**else** **if**(((control^SABM)==0) || ((control^SABM)==16)){
259. tipo="SABM";
260. }**else** **if**(((control^SABME)==0) || ((control^SABME)==16)){
261. tipo="SABME";
262. }**else** **if**(((control^UI)==0) || ((control^UI)==16)){
263. tipo="UI";
264. }**else** **if**(((control^DISC\_RD)==0) || ((control^DISC\_RD)==16)){
265. tipo="DISC\_RD";
266. }**else** **if**(((control^RSET)==0) || ((control^RSET)==16)){
267. tipo="RSET";
268. }**else** **if**(((control^XID)==0) || ((control^XID)==16)){
269. tipo="XID";
270. }
271. }**else**{
272. **if** ((control^SARM\_DM)==0 || (control^SARM\_DM)==16){
273. tipo="DM";
274. }**else** **if** ((control^UI)==0 || (control^UI)==16){
275. tipo="UI";
276. }**else** **if** ((control^UA)==0 || (control^UA)==16){
277. tipo="UA";
278. }**else** **if** ((control^DISC\_RD)==0 || (control^DISC\_RD)==16){
279. tipo="RD";
280. }**else** **if** ((control ^ XID) == 0 || (control ^ XID) == 16){
281. tipo = "XID";
282. }
283. }
284. **return** tipo;
285. }
287. **static** **final** String[] POLL\_FINAL=**new** String[]{"P","F"};
289. **static** **final** **int** SNRM=0b10000011;
290. **static** **final** **int** SNRME=0b11001111;
291. **static** **final** **int** SARM\_DM=0b00001111;
292. **static** **final** **int** SABM=0b00101111;
293. **static** **final** **int** SABME=0b01101111;
294. **static** **final** **int** UI=0b00000011;
295. **static** **final** **int** UA=0b01100011;
296. **static** **final** **int** DISC\_RD=0b01000011;
297. **static** **final** **int** RSET=0b10001111;
298. **static** **final** **int** XID=0b10101111;
300. **static** **final** **int** RR=0b00;
301. **static** **final** **int** REJ=0b10;
302. **static** **final** **int** RNR=0b01;
303. **static** **final** **int** SREJ=0b11;
304. }

**Pruebas:**

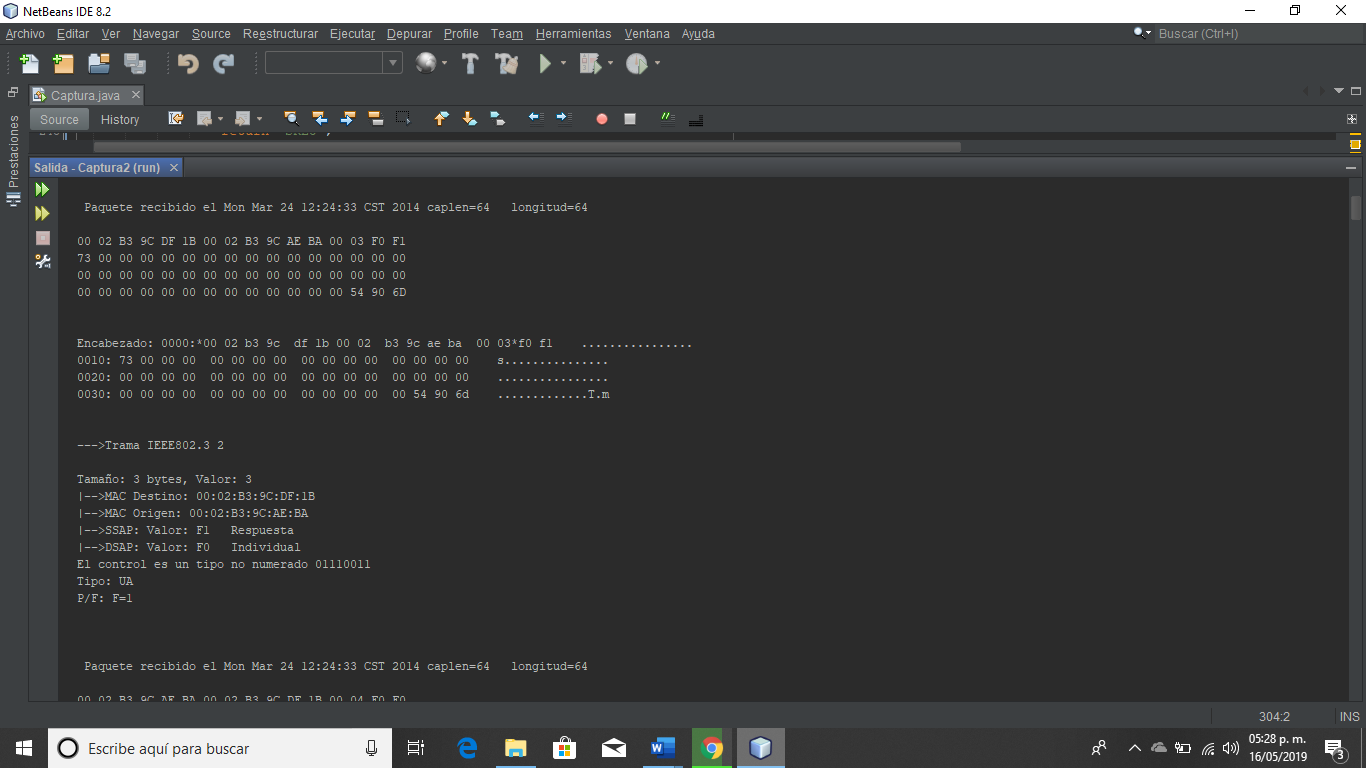
A continuación, se hacen pruebas donde a través de un archivo. pcap, se nos facilitan algunas tramas para su análisis.



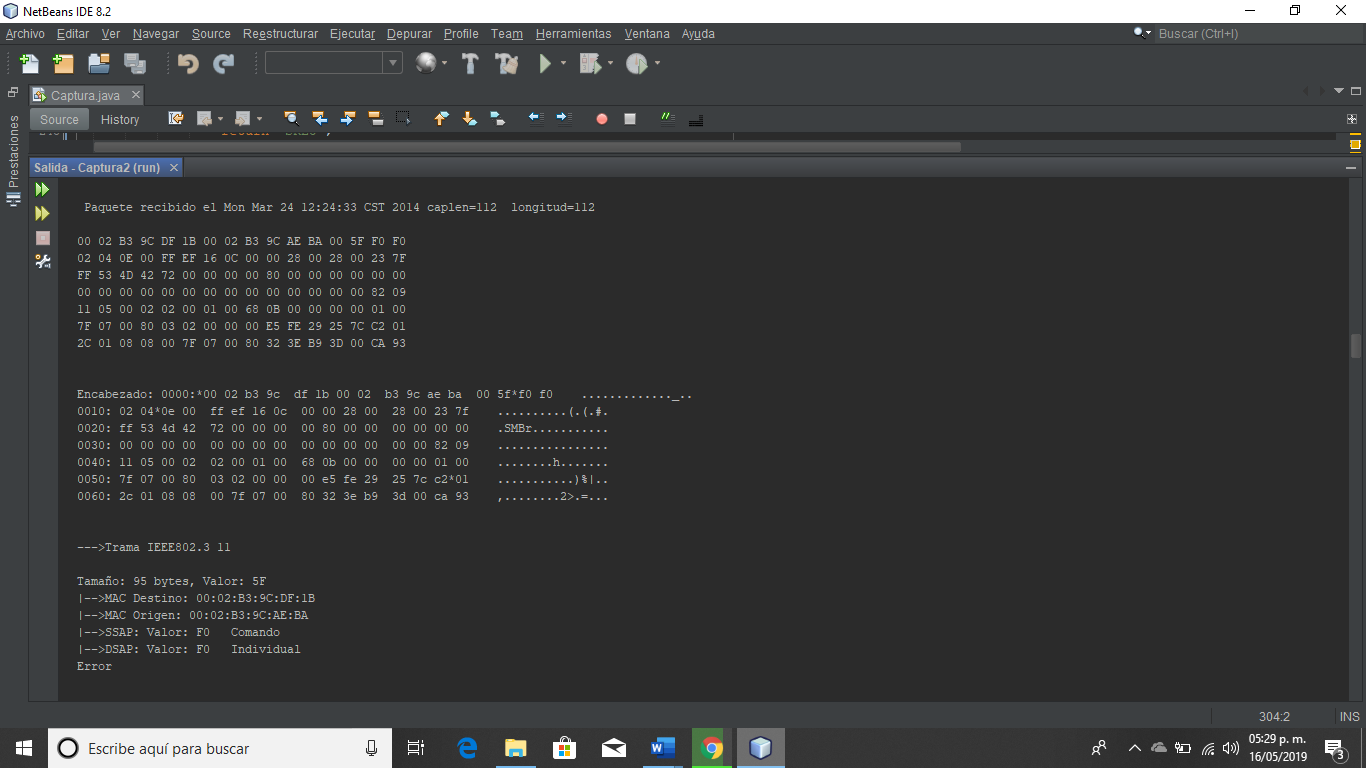
**Trama 1:**



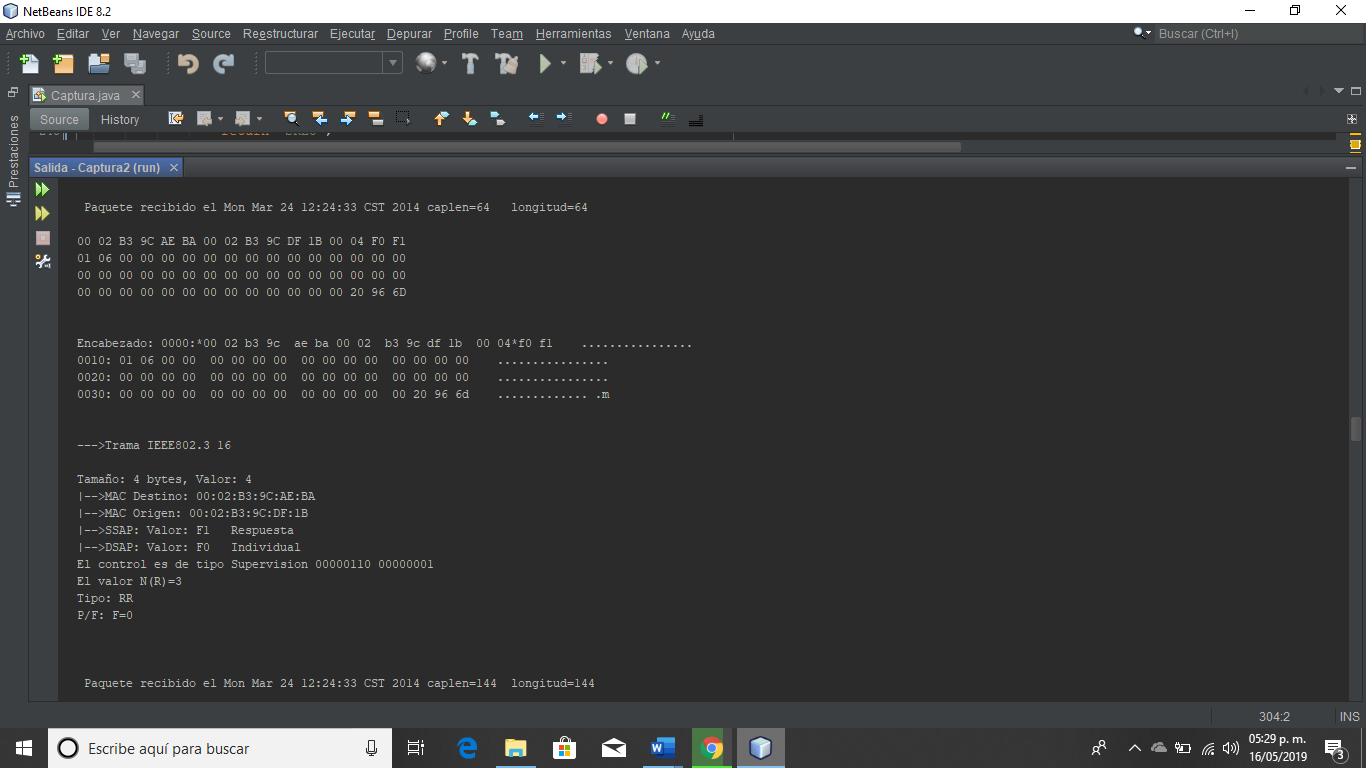
**Trama 2:**



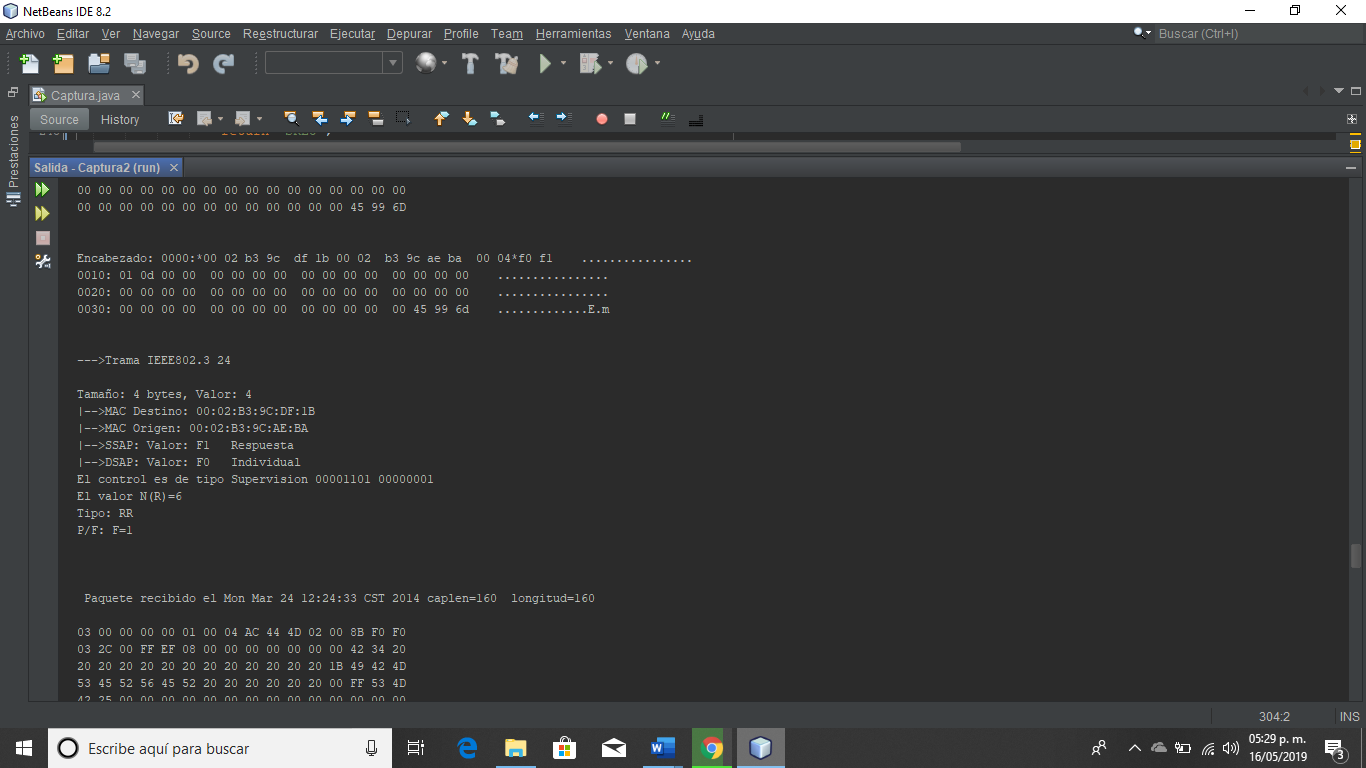
**Trama 11:**



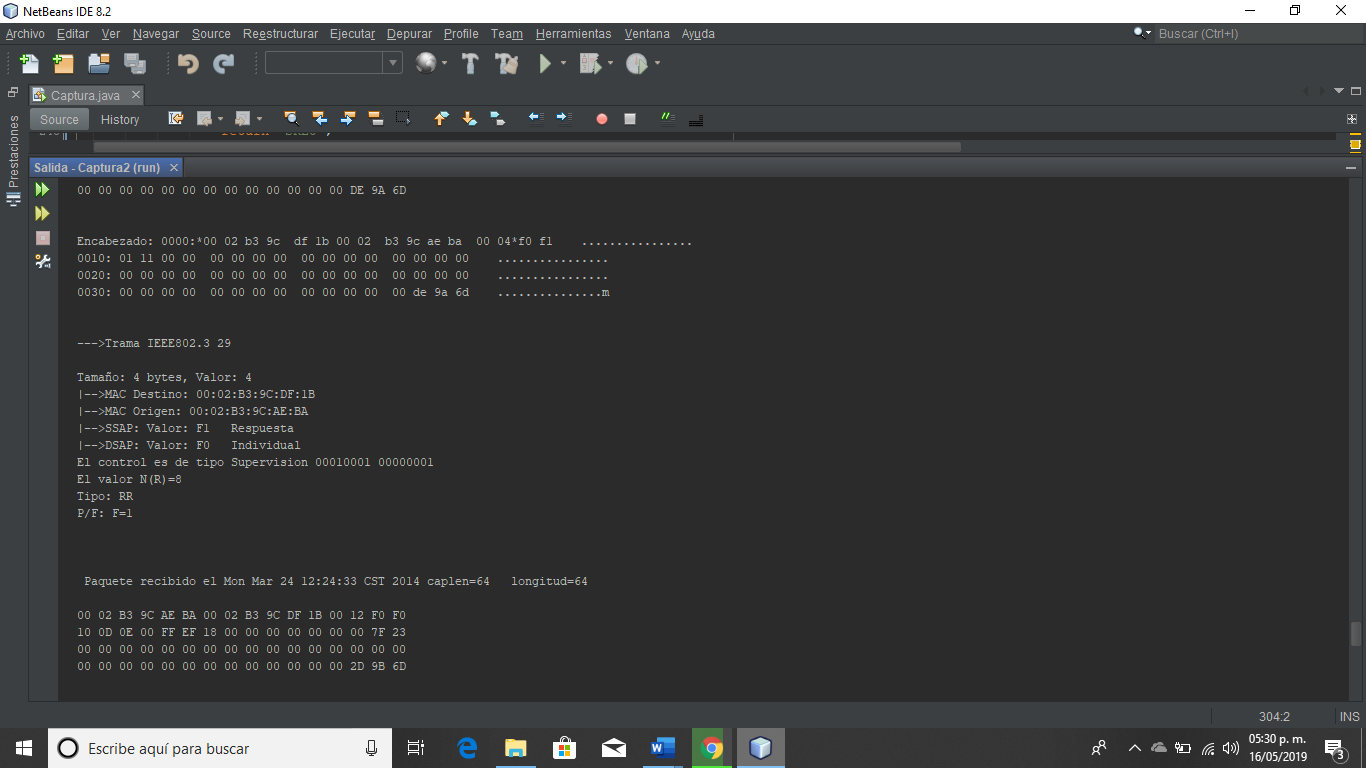
**Trama 16:**



**Trama 24:**



**Trama 29:**



**Conclusiones.**

**Mario Alberto Miranda Sandoval.**

En esta práctica comprendí mejor el análisis de tramas, ya que este es importante para poder desarrollar este programa, por lo que sin el conocimiento no se hubiera podido haber realizado, posteriormente comprendí de mejor manera lo que es el protocolo LLC, además, de saber que no es el único que existe.

**Rojas Alvarado Luis Enrique.**

En ésta práctica pudimos observar el comportamiento de una trama identificando las diferentes partes que la forman en su encabezado de IP, por lo que usamos el programa proporcionado por el profesor desde su servidor simplemente para hacer un simple análisis de las tramas.