

**Instituto Politecnico Nacional**

**ESCOM “ESCUELA SUPERIOR DE CÓMPUTO”**

*REDES DE COMPUTADORAS*

*PRÁCTICA 4: IMPLEMENTACIÓN DE UN PROTOCOLO*

PROFE: Axel Moreno Cervantes

ALUMMNOS: Rojas Alvarado Luis Enrique

Mario Alberto Miranda Sandoval

GRUPO: 2CM10

**Introducción.**

En [informática](https://es.wikipedia.org/wiki/Inform%C3%A1tica) y [telecomunicación](https://es.wikipedia.org/wiki/Telecomunicaci%C3%B3n), un **protocolo de comunicaciones** es un sistema de reglas que permiten que dos o más entidades de un [sistema de comunicación](https://es.wikipedia.org/wiki/Telecomunicaci%C3%B3n#Sistema_de_comunicaci%C3%B3n) se comuniquen entre ellas para transmitir [información](https://es.wikipedia.org/wiki/Informaci%C3%B3n) por medio de cualquier tipo de variación de una [magnitud física](https://es.wikipedia.org/wiki/Magnitud_f%C3%ADsica). Se trata de las reglas o el estándar que define la [sintaxis](https://es.wikipedia.org/wiki/Sintaxis), [semántica](https://es.wikipedia.org/wiki/Sem%C3%A1ntica) y [sincronización](https://es.wikipedia.org/wiki/Sincronizaci%C3%B3n) de la [comunicación](https://es.wikipedia.org/wiki/Comunicaci%C3%B3n), así como también los posibles [métodos de recuperación de errores](https://es.wikipedia.org/wiki/C%C3%B3digos_detectores_y_correctores_de_error). Los protocolos pueden ser implementados por [hardware](https://es.wikipedia.org/wiki/Hardware), por [software](https://es.wikipedia.org/wiki/Software), o por una combinación de ambos.

También se define como un conjunto de normas que permite la comunicación entre ordenadores, estableciendo la forma de identificación de estos en la red, la forma de transmisión de los datos y la forma en que la información debe procesarse.

Los sistemas de comunicación utilizan formatos bien definidos (protocolo) para intercambiar mensajes. Cada mensaje tiene un significado exacto destinado a obtener una respuesta de un rango de posibles respuestas predeterminadas para esa situación en particular. Normalmente, el comportamiento especificado es independiente de cómo se va a implementar. Los protocolos de comunicación tienen que estar acordados por las partes involucradas. Para llegar a dicho acuerdo, un protocolo puede ser desarrollado dentro de [estándar técnico](https://es.wikipedia.org/wiki/Norma_(tecnolog%C3%ADa)). Un [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) describe el mismo para los cálculos, por lo que existe una estrecha analogía entre los protocolos y los lenguajes de programación: «los protocolos son a las comunicaciones como los lenguajes de programación son a los cómputos».​ Un protocolo de comunicación, también llamado en este caso **protocolo de red**, define la forma en la que los distintos mensajes o [tramas](https://es.wikipedia.org/wiki/Trama_de_red) de bit circulan en una [red de computadoras](https://es.wikipedia.org/wiki/Red_de_computadoras).

**Acerca de nuestro protocolo.**

Los primeros 6 bytes corresponden a la MAC destino (0-5 bytes).

Los siguientes 6 bytes corresponden a la MAC origen (6-11 bytes).

Para el campo de tipo tenemos los bytes 12 y 13.

Posteriormente para el envío de datos se contarán con los bytes 14 al 1000.

Por último, para su verificación usamos un checksum que toma los bytes 1001 y 1002.

Checksum

1001-1002 bytes

Datos

14-1000 bytes

Tipo

12-13 bytes

MAC Origen

6-11 bytes

MAC Destino

0-5 bytes

**Desarrollo.**

El código que se modifico fue el siguiente, se coloca la parte donde el código que nos había dado el profesor fue modificado, en adición a este se le añadió un checksum.

1. **byte**[] trama = **new** **byte**[1500];
3. **for**(**int** k=0;k<MACo.length;k++){
4. trama[k] = (**byte**) 0xff;
5. trama[k+6]=MACo[k];
6. }
8. trama[12]= (**byte**) 0x16; //tipo sin asignar
9. trama[13]= (**byte**) 0x01; //tipo sin asignar rfc 1340

12. /////////////////Abre Archivo
14. File archivo = **null**;
15. FileReader frr = **null**;
16. BufferedReader brr;
17. brr = **null**;
19. **try** {
20. archivo = **new** File ("C:\\Users\\Wicho\\Documents\\ESCOM\\Redes\\Práctica4\\TEXTO.txt");
21. frr = **new** FileReader (archivo);
22. brr = **new** BufferedReader(frr);
23. // Lectura del fichero
24. FileInputStream ficheroStream = **new** FileInputStream(archivo);
25. **byte** contenido[] = **new** **byte**[(**int**)archivo.length()];
26. ficheroStream.read(contenido);
28. }**catch**(Exception e){
29. e.printStackTrace();
30. }**finally**{
31. **try**{
32. **if**( **null** != frr ){
33. frr.close();
34. }
35. }**catch** (Exception e2){
36. e2.printStackTrace();
37. }
38. }
40. //String linea=new String(contenido);
42. //System.out.println(linea);
44. String mensaje= "Un mensaje para el protocolo practica numero 4";
46. **byte**[]buf = mensaje.getBytes();
47. **int** tam = buf.length;
48. **if**(tam<1000){
49. **for**(**int** c=0;c<tam;c++)
50. trama[16+c]=buf[c];
51. }**else**{
52. System.out.println("El mensaje es muy largo... maximo 1000 bytes");
53. System.exit(1);
54. }
56. ///se define la MAC a la que se enviará
57. trama[0]= (**byte**) 0x70;
58. trama[1]= (**byte**) 0xC9;
59. trama[2]= (**byte**) 0x4A;
60. trama[3]= (**byte**) 0xB4;
61. trama[4]= (**byte**) 0xCF;
62. trama[5]= (**byte**) 0xE1;
63. trama[14]=0;
64. trama[15]=0;
66. //Arrays.fill(a, (byte) 0xff);
67. ByteBuffer b = ByteBuffer.wrap(trama);



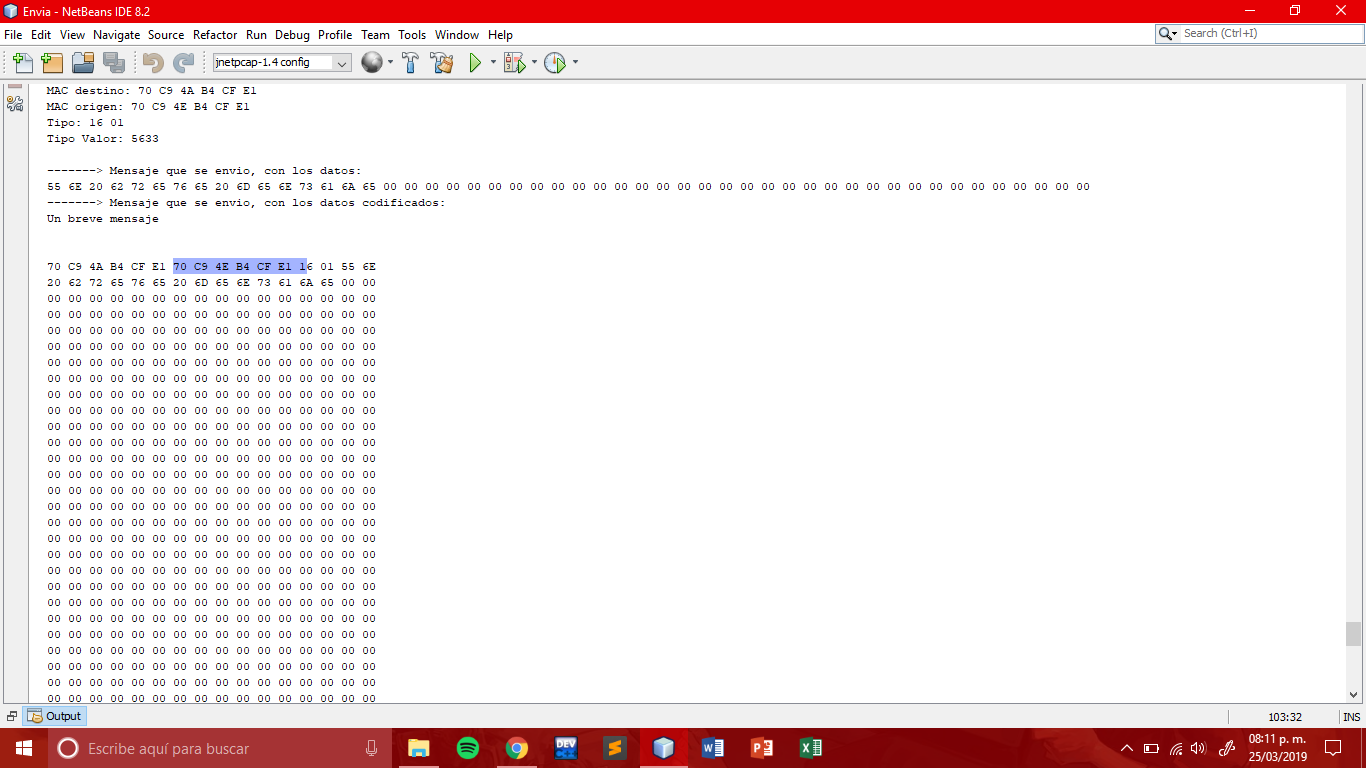

73. /\*\*\*\*\*\*\*\*F I L T R O\*\*\*\*\*\*\*\*/
74. PcapBpfProgram filter = **new** PcapBpfProgram();
75. String expression ="ether proto 0x1601"; // "port 80";
76. **int** optimize = 0; // 1 means true, 0 means false
77. **int** netmask = 0;
78. **int** r2 = pcap.compile(filter, expression, optimize, netmask);
79. **if** (r2 != Pcap.OK) {
80. System.out.println("Filter error: " + pcap.getErr());
81. }//if
82. pcap.setFilter(filter);
83. PcapPacketHandler<String> jpacketHandler = **new** PcapPacketHandler<String>() {

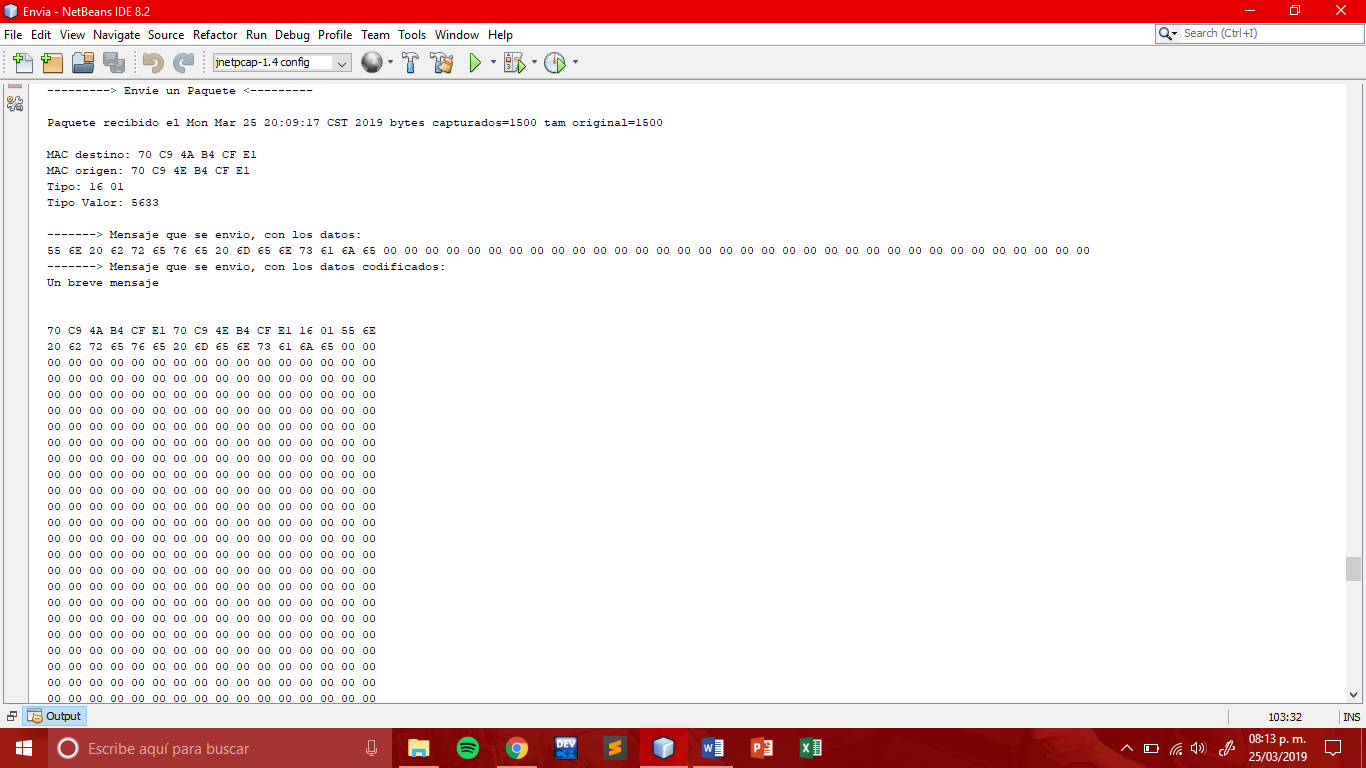
86. **public** **void** nextPacket(PcapPacket packet, String user) {
88. System.out.printf("Paquete recibido el %s bytes capturados=%-4d tam original=%-4d %s\n\n",
89. **new** Date(packet.getCaptureHeader().timestampInMillis()),
90. packet.getCaptureHeader().caplen(),  // Length actually captured
91. packet.getCaptureHeader().wirelen(), // Original length
92. user                                 // User supplied object
93. );
94. /\*\*\*\*\*\*Desencapsulado\*\*\*\*\*\*\*\*/
95. System.out.print("MAC destino: ");
96. **for**(**int** i=0;i<6;i++){
97. System.out.printf("%02X ",packet.getUByte(i));
98. }
100. System.out.println("");
102. System.out.print("MAC origen: ");
103. **for**(**int** i=6;i<12;i++){
104. System.out.printf("%02X ",packet.getUByte(i));
105. }
107. System.out.println("");
109. System.out.print("Tipo: ");
110. **for**(**int** i=12;i<14;i++){
111. System.out.printf("%02X ",packet.getUByte(i));
112. }
114. System.out.println("");
116. **int** tipo = (packet.getUByte(12)\*256)+packet.getUByte(13);
118. System.out.printf("Tipo Valor: %d \n",tipo);
120. **if**(tipo==5633){ //0x1601
122. System.out.print("\n-------> Mensaje que se envio, con los datos: \n");
124. **byte**[]t = packet.getByteArray(14, 1000);
125. **int** longitud =0;
126. **for**(**int** k=0;k<t.length;k++){
127. System.out.printf("%02X ",t[k]);
128. **if**(t[k]!=0)
129. longitud++;
130. }
132. System.out.print("\n-------> Mensaje que se envio, con los datos codificados: \n");
134. String datos = **new** String(t);
135. System.out.println(datos+"jajaja \n\n");
136. System.out.println(longitud);
137. **for**(**int** l=0;l<packet.size();l++){
138. System.out.printf("%02X ",packet.getUByte(l));
139. **if**(l%16==15)
140. System.out.println("");
141. }
143. System.out.printf("//////////////////////////////////////////////////\n\n");
145. longitud = longitud+2;

148. **long** checkSum=Checksum.calculateChecksum(packet.getByteArray(14,longitud));
149. **if**(checkSum==0){
150. System.out.printf("\nEl envío es correcnto %02X\n",checkSum);
151. trama[14]=(**byte**)((checkSum>>8)&0xFF);
152. trama[15]=(**byte**)(checkSum&(0XFF));
153. **for**(**int** l=0;l<packet.size();l++){
154. System.out.printf("%02X ",trama[l]);
155. **if**(l%16==15){
156. System.out.println("");
157. }
158. }
159. }
160. **else**{
161. System.out.println("La wea no funciona");
162. }
163. }//if
165. //System.out.println("\n\nEncabezado: "+ packet.toHexdump());

168. }
169. };
170. //pcap.loop(1, jpacketHandler, "");
171. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
172. \* Fourth We send our packet off using open device
173. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
174. **for**(**int** zz=0;zz<10;zz++){
175. //if (pcap.sendPacket(b) != Pcap.OK) {
176. **if**(pcap.inject(b)<0){
177. **if** (pcap.sendPacket(trama) != Pcap.OK) {
178. System.err.println(pcap.getErr());
179. }
180. System.out.println("---------> Envie un Paquete <--------- \n");
181. **try**{
182. Thread.sleep(500);
183. }**catch**(InterruptedException e){}
184. pcap.loop(1, jpacketHandler, "");
185. }//for
186. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
187. \* Lastly we close
188. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
189. pcap.close();
191. }**catch**(Exception e){
192. e.printStackTrace();
193. }//catch
194. }

**Pruebas de funcionamiento.**





**Conclusión.**

**Mario Alberto Miranda Sandoval.**

Al principio creía que esta práctica no iba a ser posible de hacerse, ya que esta en un inicio había dado muchos problemas. Pero después de un arduo tiempo de trabajo se logró, dejando en claro como funcionan los protocolos, ya que en nuestro caso solo fue necesario hacer que las MAC estuvieran en el mismo segmento de red para que después lo que siguiera fuera nuestro segmento de datos y el campo de checksum, para garantizar el envío.

**Rojas Alvarado Luis Enrique**

En ésta práctica supimos cómo es que se implementa un protocolo para la comunicación, nosotros lo hicimos en una sola máquina, pudiéndose hacer en dos máquinas remotas, lo elegimos así para la mejor visualización del protocolo y porque se facilita más identificar errores. Probamos cada dato que se mandara e imprimimos cada dato recibido y enviado para tener el mejor control del envío de datos y del programa en general.