



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

Práctica 4

Implementación de un protocolo de
comunicación para transferencia de archivos

Unidad de aprendizaje: Redes de computadoras

Grupo: 2CM10

Alumnos(a):

Nicolás Sayago Abigail
Ramos Díaz Enrique

Profesor(a):

Moreno Cervantes Axel

30 de Abril de 2018

Índice

1	Introducción	2
2	Marco Teórico	2
2.1	Protocolo. Definición	2
2.2	Características de un protocolo	2
2.3	Clasificación	2
3	Diseño de la solución	3
3.1	MAC Destino y MAC Origen	3
3.2	Tipo	4
3.3	Numero de Tramas	4
3.4	Protocolo	4
3.5	P/G - Control	4
3.6	Longitud del archivo	4
3.7	Longitud del nombre del archivo	4
3.8	Nombre del archivo	4
3.9	Archivo	4
3.10	Checksum	4
3.11	Aplicación	5
3.11.1	Interfaces de red y MAC Destino	5
3.11.2	Selección y lectura del archivo	5
3.11.3	Llenado y división (si es necesario) de la trama	5
3.11.4	Envío la trama con el archivo leído	5
3.11.5	Recibo la trama	6
3.11.6	Escritura el archivo recibido en la trama	6
4	Implementación de la solución	7
4.1	Selección y lectura del archivo	7
4.2	Llenado de la trama	7
4.3	Envío la trama con el archivo leído	9
4.4	Recibo la trama	10
4.5	Escritura el archivo recibido en la trama	11
5	Funcionamiento	13
	Referencias	16

1. Introducción

En este reporte se mostrará como crear una aplicación en Java, con ayuda de la librería Jnetpcap, que implemente un protocolo diseñado por nosotros mismos, capaz de enviar y recibir archivos de todo en tipo utilizando tramas y ayudandonos del protocolo Ethenert Type II, de una computadora a otras demás conectadas en red. Nos basaremos en la clase llamada Envía.

2. Marco Teórico

2.1. Protocolo. Definición

En informática y telecomunicación, un protocolo es un conjunto de reglas que regulan el intercambio de información entre dispositivos.

Existen diversos protocolos de acuerdo a cómo se espera que sea la comunicación. Algunos protocolos, por ejemplo, se especializarán en el intercambio de archivos (FTP); otros pueden utilizarse simplemente para administrar el estado de la transmisión y los errores (como es el caso de ICMP), etc.

2.2. Características de un protocolo

- Sintaxis: Forma de los datos, codificación y niveles de señal.
- Semántica: Información de control para manejo de errores.
- Temporización: Duración de cada símbolo, sincronización y secuencia.

2.3. Clasificación

- Monolíticos: En la aproximación monolítica, una modificación cualquiera de detalles implicara que toda la aplicación debería modificarse, con el riesgo de introducir errores difíciles de localizar.
- Estructurados: Otro método es la técnica de diseño e implementación estructurada, que consiste en un conjunto de protocolos organizados con una estructura por capas o jerárquica.

La comunicación entre dos entidades puede ser directa o indirecta. La comunicación es directa cuando los datos y la información de control pasaran directamente entre las entidades sin la intervención de un agente activo. Cuando los sistemas se conectan a través de una red conmutada ya no existe un protocolo directo.

Un protocolo puede ser simétrico o asimétrico. En los simétricos se involucran entidades pares. Se ejecuta un proceso de intercambio (Arquitectura Cliente/Servidor). Normalmente esto implica que un ordenador sondea una serie de terminales.



Por ultimo un protocolo puede ser estándar o no estándar. El no estándar es aquel que se diseña y se implementa para una comunicación particular, o al menos para un ordenador con un modelo particular. Y el estándar se implementa para una comunicación bastante organizada la cual requiere de una serie de estándares para poder llevarse acabo.

3. Diseño de la solución

Antes que nada, debemos de diseñar el esquema de nuestro protocolo. Nos basamos en las tramas Ethernet y un poco en las IEEE802.3, así como con las del protocolo de transporte UDP.

MAC DESTINO (6 Bytes)				MAC ORIGEN (6 Bytes)		
Tipo (2 Bytes)	P/G (1 B)	# Tramas (2 Bytes)	Protocolo (1 B)	Longitud Archivo (2 Bytes)	Longitud Nombre (2 Bytes)	Nombre Archivo (0-50 Bytes)
Archivo (0-1460 Bytes)						Checksum (2 Bytes)

3.1. MAC Destino y MAC Origen

La MAC Origen se obtiene automáticamente gracias a una funcion de Jnetpcap.

La MAC Destino es digitada por el usuario dentro de una caja de texto.

3.2. Tipo

Utilizamos los bytes 0x16 y 0x01, el cual es un tipo sin asignar según la norma RFC 1340.

3.3. Numero de Tramas

En las posiciones 14 y 15 colocamos el numero de tramas totales a enviar en que se divide un archivo, en 2 bytes.

3.4. Protocolo

Optamos por identificar a nuestro protocolo con el byte 0xC1.

3.5. P/G - Control

Un byte de control que determina si el archivo es pequeño (cabe en 1 sola trama) y se denota con un 0xA0, o si es grande (se necesita más de una trama) y se denota con un 0xA1.

3.6. Longitud del archivo

Guardamos la longitud de los bytes del archivo a enviar. Nos sera de gran ayuda al momento del desencapsulado en el receptor.

3.7. Longitud del nombre del archivo

Guardamos la longitud de los bytes del nombre del archivo a enviar. Nos sera de gran ayuda al momento del desencapsulado en el receptor.

3.8. Nombre del archivo

Se guarda en bytes el nombre del archivo. La maxima longitud es de 50 bytes.

3.9. Archivo

Se guarda en bytes el archivo a enviar. La máxima longitud es de 1460 bytes.

3.10. Checksum

Se utiliza como control de errores en caso de que las tramas recibidas hayan llegado con algun byte perdido, incorrecto o adicional. Es de gran ayuda al momento de leer los bytes del archivo.

3.11. Aplicación

3.11.1. Interfaces de red y MAC Destino

Debemos obtener las interfaces de red disponibles en las computadoras, y seleccionar una para abrir un canal de comunicación entre ellas.

Para esto, la aplicacion cargara automaticamente todas las interfaces de red disponibles y las colocara en un ComboBox, utilizando la libreria jnetpcap y la funcion findalldevs(). El usuario será capaz de seleccionar una de ellas.

Es de vital importancia que las computadoras estén conectadas en red.

Se procede a dar clic en el boton **Start**, que creara el medio de transmisión entre las computadoras por medio de un hilo y la funcion openlive(). Se asigna la MAC de origen.

Posteriormente, el usuario debe registrar la MAC Destino, es decir, la MAC de la computadora a la cual se desea enviar el archivo.

3.11.2. Selección y lectura del archivo

Este modulo nos fue proporcionado en clase.

Básicamente creamos un JFileChooser para navegar por nuestros directorios de la computadora en la que estamos y así elegir un archivo. Este se lee y se carga en un objeto de tipo RandomAccessFile, que nos servirá para colocar el archivo como un arreglo de bytes en la trama.

Tambien guardamos el nombre del archivo y el directorio de su ubicación.

3.11.3. Llenado y división (si es necesario) de la trama

Luego de que usuario haya seleccionado y se haya leído el archivo, procedemos a llenar la trama.

Para convertir el archivo en arreglo de bytes, utilizamos el metodo readFully() y le enviamos como parametro el arreglo en donde se depositaran los bytes del archivo.

Colocamos dentro de un arreglo de bytes todas y cada una de las partes de nuestro esquema del protocolo, en el orden que hemos explicado anteriormente.

Sin embargo, si el tamaño del archivo excede los 1460 bytes de la trama, es necesario separarlo en n tramas. Así que dividimos la longitud total del archivo entre 1460, para obtener la cantidad de tramas que se enviarán.

Se llenan una a una, colocando todos los campos igual, a excepcion del campo archivo, que serán las partes en que se dividió el archivo (de forma ordenada).

3.11.4. Envío la trama con el archivo leído

Una vez obtenidas una o las n tramas del archivo a enviar, utilizamos la funcion sendPacket() que se ejecutara dentro de un hilo, esto para permitirnó enviar más de una trama al mismo tiempo.

Para controlar el orden de llenado y envio, colocamos un `Thread.sleep()` después de cada una de estas operaciones en una trama. Asi evitamos que se envíen en desorden y el receptor pueda recibirlas inmediatamente que se envían.

3.11.5. Recibo la trama

Para recibir las tramas, creamos un objeto de tipo `PcapPacketHandler` que tendra la funcion `nextPacket()`.

Después creamos un hilo, para poder recibir más de una trama al mismo tiempo, y aquí colocamos un `pcap.loop()` de forma infinita, para recibir todas las tramas que queramos.

Dentro del `nextPacket()` se encuentra la funcion `analizaTrama()`, donde se hara el desencapsulado de esta y la escritura del archivo en la maquina receptora.

3.11.6. Escritura el archivo recibido en la trama

En la funcion `analizaPacket()`, revisamos es si la MAC de Destino de la trama coincide con la MAC de la maquina receptora; tambien revisamos que la trama sea de tipo 0x1601.

Ya verificada, lo primero que hacemos es revisar el campo de Numero de Tramas, recordemos que las tramas llegan en orden gracias al `Thread.sleep()`.

Esto es necesario porque vamos a meter los bytes del campo Archivo en un `ArrayList`, y es necesario saber cuantas veces vamos a extraer esos bytes de las N tramas en que se dividio el archivo antes de ser enviado.

Revisamos el checksum de estos campos y lo comparamos con el valor que viene en la trama, para asi identificar errores en los bytes. Si se encuentra algun error, el archivo se escribe igual, pero se avisa que posee bytes incorrectos o faltantes.

Cuando ya tenemos todos los bytes del archivo listos en nuestro `ArrayList`, creamos un objeto de tipo `RandomAccessFile`, y le enviamos como parametro la ruta en donde se escribira el archivo.

La ruta por defecto es C:\received

Luego, creamos un arreglo con la longitud total del archivo en bytes, y vamos depositando aqui los bytes del `ArrayList`.

Finalmente, se ejecuta el método `write()` en el `RandomAccessFile` y le enviamos como parametro el arreglo con los bytes completos.

Si todo salio bien, aparece un mensaje que confirma la escritura del archivo, y se podrá acceder a éste en la ruta especificada.

4. Implementación de la solución

4.1. Selección y lectura del archivo

```
1      try{ // LECTURA DEL ARCHIVO
2          /* Caja de dialogo: Te muestra los archivos de la maquina.
3             Tiene el boton aceptar y cancelar*/
4          JFileChooser jf = new JFileChooser();
5          //jf.setMultiSelectionEnabled(true);
6          // Valor de la constante
7          int r = jf.showOpenDialog(null);
8          if(r == JFileChooser.APPROVE_OPTION){
9              // Devuelve el archivo
10             File f = jf.getSelectedFile();
11             try{
12                 archivoOrigen = new RandomAccessFile(f, "r");
13             }catch(IOException e){
14                 JOptionPane.showMessageDialog(null, "No se ha podido leer
15                     el archivo.", "Error", JOptionPane.ERROR_MESSAGE);
16             }
17             // Para mostrar las propiedades del archivo
18             nombre_arc = f.getName();
19             path = f.getAbsolutePath();
20             // Mostramos el nombre del archivo
21
22             long tam = f.length();
23             // Obtenemos el n mero de tramas
24             int cociente =(int)tam/1460;
25             int residuo = (int)tam-(cociente*1460);
26             partes = cociente+1;
27             System.out.println("NUMERO DE TRAMAS: "+partes);
28             // Los datos primitivos almacena valores.
29             // Lee datos desde cualquier tipo de datos primitivos
30             DataInputStream dis = new DataInputStream(new FileInputStream
31                 (path));
32             RutaArchivoSelec.setText(path);
33             EnviarArchivo.setEnabled(true);
34         }
35     }
36     catch(IOException io){ //
37         JOptionPane.showMessageDialog(null, "No se ha podido leer
38             el archivo.", "Error", JOptionPane.ERROR_MESSAGE);
39     }
```

4.2. Llenado de la trama

```
1      public static void llenarTrama(byte[] trama, byte[] MACdestino,
2      byte[] MACorigen, byte[] narch, byte[] arch, int tamname,
3      int tamarch, int numTramasTotal) throws IOException{
4          //Llena la trama ARCHIVO a enviar
5          for(int k=0; k<tamarch; k++){
6              System.out.printf(" %02X ", arch[k]);
```



```

7
8      // COLOCA EN LAS POSICION 0-5 MAC DESTINO Y 6-11 MAC
9      System.out.println("");
10     for(int k=0;k<6;k++){
11         trama[k] = MACdestino[k];
12         trama[k+6]=MACorigen[k];
13         System.out.printf("%02X ", trama[k]);
14     }
15
16     int byte_long1, byte_long2, long1_n, long2_n, totalTram1, totalTram2;
17     long chk1, chk2;
18
19     trama[12]= (byte) 0x16; //tipo sin asignar BYTE 12
20     trama[13]= (byte) 0x01; //tipo sin asignar rfc 1340    BYTE 13
21
22     totalTram1 = (numTramasTotal>>8)&0xFF;
23     totalTram2 = (numTramasTotal)&0x00FF;
24
25     trama[15] = (byte) totalTram1; // Numero de total de tramas
26     dividida en 2 bytes
27     trama[16] = (byte) totalTram2;
28
29     trama[17]= (byte) 0xC1; //Protocolo 0xC1
30
31     byte_long1=(tamarch>>8)&0xFF;
32     byte_long2=(tamarch)&0x00FF;
33
34     trama[18]= (byte) byte_long1; //Longitud del mensaje divida en
35     2 bytes
36     trama[19]= (byte) byte_long2;
37
38     long1_n=(tamname>>8)&0xFF;
39     long2_n=(tamname)&0x00FF;
40
41     trama[20]= (byte) long1_n; //Longitud del nombre divida en 2 bytes
42     trama[21]= (byte) long2_n;
43
44     chk1=(Checksum.calculateChecksum(arch)>>8)&0xFF;
45     chk2=(Checksum.calculateChecksum(arch))&0x00FF;
46
47     trama[trama.length-2]= (byte) chk1; //Checksum del mensaje
48     trama[trama.length-1]= (byte) chk2;
49
50     for(int c=0;c<tamname;c++)
51         trama[22+c]=narch[c];
52
53     for(int c=0;c<tamarch;c++)
54         trama[22+tamname+c]=arch[c];
55
56     if(tamarch<=1460) {
57         trama[14]= (byte) 0xA0; //Si el mensaje es peque o, sera P=A0
58     }
59     else
60         trama[14]= (byte) 0xA1; //Si el mensaje es grande, sera G=A1
61 }

```

4.3. Envió la trama con el archivo leído

```

1      try{
2          MACd=stringToMAC (MACDestino.getText());
3
4          if(getLong(nombre_arc.getBytes())>50)
5              JOptionPane.showMessageDialog(null, "El nombre del archivo es
6                  muy largo");
7          else{
8              byte[] buf = new byte[(int)archivoOrigen.length()];
9              // Lee el archivo origen
10             archivoOrigen.readFully(buf);
11
12             if((int)archivoOrigen.length()>1460){
13                 int b, indice=0;
14                 int tamMaxArchivo;
15                 for (int a=0; a<(partes-1); a++){
16                     b=a;
17                     tamMaxArchivo = 1460;
18                     byte[] tramal = new byte[getLong(nombre_arc.getBytes())
19                         +tamMaxArchivo+24];
20                     byte[] arc = new byte[1460];
21                     indice = b*1460;
22                     for(int i=0; i<tamMaxArchivo; i++){
23                         arc[i] = buf[indice];
24                         indice++;
25                     }
26                     llenarTrama(tramal, MACd, MACo, nombre_arc.getBytes(),
27                         arc, getLong(nombre_arc.getBytes()), tamMaxArchivo,
28                         partes);
29                     System.out.printf("\nSE LLENO LA TRAMA      " +
30                         (int)(a+1) + "\n");
31                     enviarTrama(tramal);
32                     System.out.printf("SE ENVIO LA TRAMA      " +
33                         (int)(a+1) + "\n\n");
34                     Thread.sleep(500);
35                 }
36
37                 tamMaxArchivo=(int)archivoOrigen.length()-((partes-1)*1460);
38                 byte[] tramaN = new byte[getLong(nombre_arc.getBytes())+
39                     tamMaxArchivo+24];
40                 byte[] arcN = new byte[1460];
41                 b = (partes-1)*1460;
42                 for(int i=0; i<tamMaxArchivo; i++){
43                     arcN[i] = buf[indice];
44                     indice++;
45                 }
46
47                 llenarTrama(tramaN, MACd, MACo, nombre_arc.getBytes(), arcN,
48                     getLong(nombre_arc.getBytes()), tamMaxArchivo, partes);
49                 System.out.printf("\nSE LLENO LA TRAMA "+ partes + "\n");
50                 enviarTrama(tramaN);
51                 System.out.printf("SE ENVIO LA TRAMA "+ partes + "\n");
52                 JOptionPane.showMessageDialog(null, "Se ha enviado

```

```

53         el archivo "+nombre_arc+" correctamente.");
54
55     }
56     else{
57         byte[] trama1 = new byte[getLong(nombre_arc.getBytes())
58         +(int)archivoOrigen.length()+24];
59         llenarTrama(trama1, MACd, MACo, nombre_arc.getBytes(), buf
60         , getLong(nombre_arc.getBytes()), (int)archivoOrigen.length(),
61         partes);
62         enviarTrama(trama1);
63         System.out.printf("\nSE ENVIO LA TRAMA\n");
64         JOptionPane.showMessageDialog(null, "Se ha enviado el
65         archivo "+nombre_arc+" correctamente.");
66     }
67 }
68 }
69 catch(Exception e){
70     e.printStackTrace();
71     JOptionPane.showMessageDialog(null, "No se ha podido enviar
72     el archivo "+nombre_arc, "Error", JOptionPane.ERROR_MESSAGE);
73 }
74
75
76 private static void enviarTrama(byte [] trama){
77     Thread hilo = new Thread(new Runnable() {
78         public void run() {
79             pcapA.sendPacket(trama);
80         }
81     });
82     hilo.start();
83 }

```

4.4. Recibo la trama

```

1 public static void recibirPaquetes() {
2     try{
3         PcapPacketHandler<String> jPacketHandler = new
4             public void nextPacket(PcapPacket paquete, String txt){
5                 analizarPaquete(paquete);
6             }
7     };
8     hiloPrincipal = new Thread(new Runnable() {
9         public void run() {
10             pcapA.loop(Pcap.LOOP_INFINITE, jPacketHandler, "");
11         }
12     });
13     hiloPrincipal.start();
14 } catch(Exception e) {
15     Logger.getLogger(VentanaProtocolo.class.getName()).log(Level.SEVERE,
16     null, e);
17 }
18 }

```

4.5. Escritura el archivo recibido en la trama

```

1  public static void analizarPaquete(PcapPacket packet) {
2      /******Desencapsulado******/
3      //Verifica la MAC destino, para evitar capturar todos los paquetes,
4      solo los dirigidos a esa MAC
5      boolean mac=true;
6      if(ChkEmisor.isSelected()) {
7          for(int k=0;k<6;k++) {
8              if(MACo[k] !=(byte) packet.getUByte(k) ) {
9                  if( (byte) packet.getUByte(k) == (byte) 0xff)
10                     mac=true;
11                 else
12                     mac=false;
13             }
14         }
15     }
16
17     if(packet.getUByte(12) == 22 && packet.getUByte(13) == 1 && mac){
18
19         System.out.printf("\n\nPaquete recibido el %s bytes capturados=%-4d
20         tam original=%-4d\n",
21             new Date(packet.getCaptureHeader().timestampInMillis()),
22             packet.getCaptureHeader().caplen(), // Length actually captured
23             packet.getCaptureHeader().wirelen() // Original length
24         );
25
26         //IMPRIME EL TOTAL DE TRAMAS QUE CONFORMAN EL ARCHIVO
27         int TramasTotal=(packet.getUByte(15)==0)?packet.getUByte(16):
28         (packet.getUByte(15)*256)+packet.getUByte(16);
29         System.out.println("NUMERO TOTAL DE TRAMAS: "+TramasTotal+"\n");
30
31         // PONE NOMBRE DE LA TRAMA EN EL ARCHIVO
32         int lname=(packet.getUByte(20)==0)?packet.getUByte(21):
33         (packet.getUByte(20)*256)+packet.getUByte(21);
34         byte []n=new byte[lname];
35         int j=0;
36
37         for(int k=(22);k<(lname+22);k++)
38             {
39                 n[j]=(byte) packet.getUByte(k);
40                 j++;
41             }
42         String namexd=getNombreArchivo(n);
43
44         // PEGA EL ARCHIVO
45         int larchivo=(packet.getUByte(18)==0)?packet.getUByte(19):
46         (packet.getUByte(18)*256)+packet.getUByte(19);
47         j=0;
48
49         System.out.println("Trama: "+ntramas);
50         if(ntramas<(TramasTotal+1)) {
51             //REVISAMOS ERRORES POR MEDIO DE CHECKSUM
52             byte[] ck=new byte[larchivo];

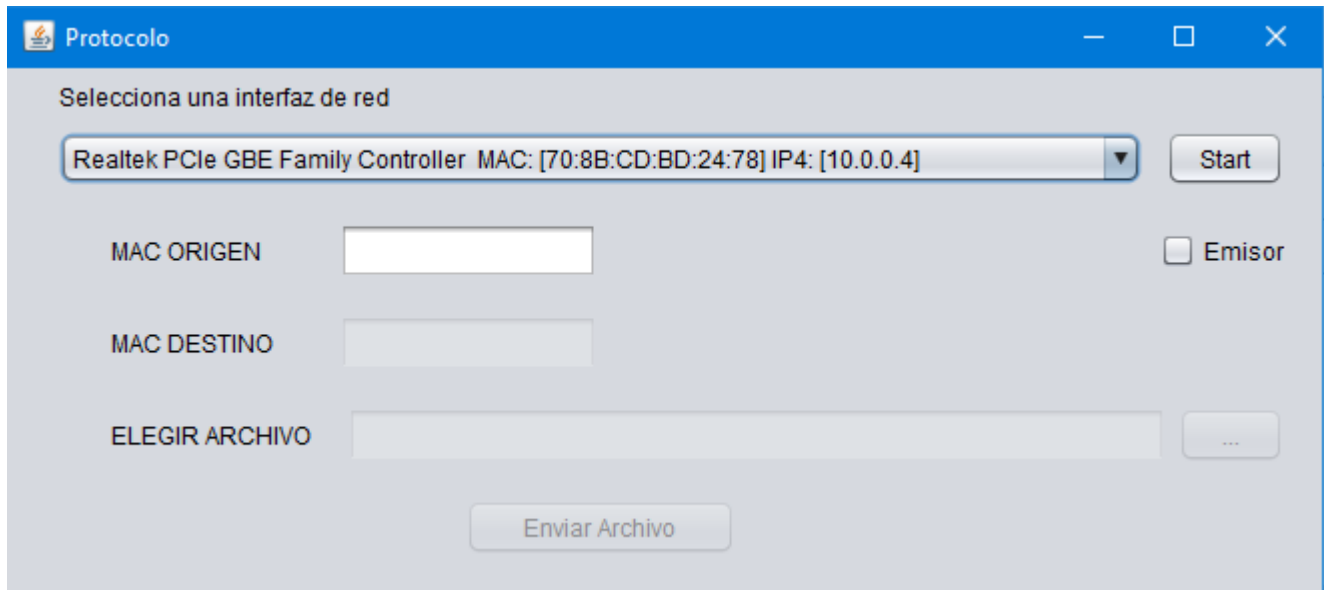
```

```

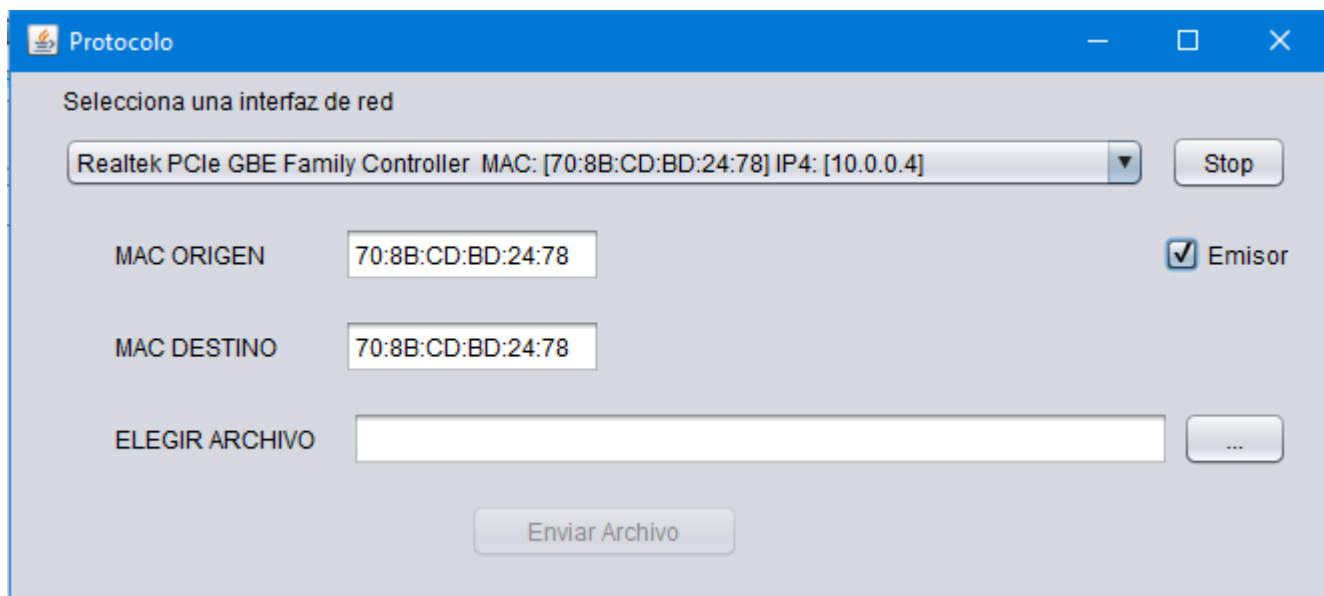
53     int cks=(packet.getUByte(packet.size()-2)==0)?packet.getUByte
54     (packet.size()-1):(packet.getUByte(packet.size()-2)*256)
55     +packet.getUByte(packet.size()-1);
56     for(int k=(lname+22);k<(lname+22+larchivo);k++)
57     {
58         ck[j]=(byte)packet.getUByte(k);
59         a.add(contadorArchivo,(byte)packet.getUByte(k));
60         System.out.printf("%02X ", a.get(j));
61         contadorArchivo++;
62         j++;
63     }
64     System.out.println("\n"+a.size()+"\n\n");
65
66     if(cks!=Checksum.calculateChecksum(ck))
67         System.out.println("\nChecksum.... AVISO: Bytes de los
68         datos de la trama erroneos.");
69     else
70         System.out.println("\nChecksum.... OK");
71
72     if(ntramas==TramasTotal){
73         try{
74             System.out.println("\nEscribiendo en disco....\n");
75             archivoDestino = new RandomAccessFile("C:\\received\\"
76             + namexd, "rw");
77             try{
78                 byte[]escribir=new byte[contadorArchivo];
79
80                 for(int k=0;k<contadorArchivo;k++){
81                     escribir[k]=a.get(k);
82                 }
83                 archivoDestino.write(escribir);//Escribe el
84                 archivo recibido en disco
85                 System.out.println("\nArchivo recibido escrito
86                 en disco");
87                 JOptionPane.showMessageDialog(null, "Se ha
88                 recibido el archivo "+namexd);
89                 LimpiarRecibidos(lname, larchivo, j, escribir,
90                 namexd, n);
91             }
92             catch(Exception e)
93             {
94                 JOptionPane.showMessageDialog(null, "No se ha
95                 podido recibir el archivo "+namexd, "Error",
96                 JOptionPane.ERROR_MESSAGE);
97             }
98         }
99         catch(FileNotFoundException ex){
100             Logger.getLogger(VentanaProtocolo.class.getName())
101             .log(Level.SEVERE, null, ex);
102         }
103     }
104 }
105     ntramas++;
106 } // Fin if
107 }

```

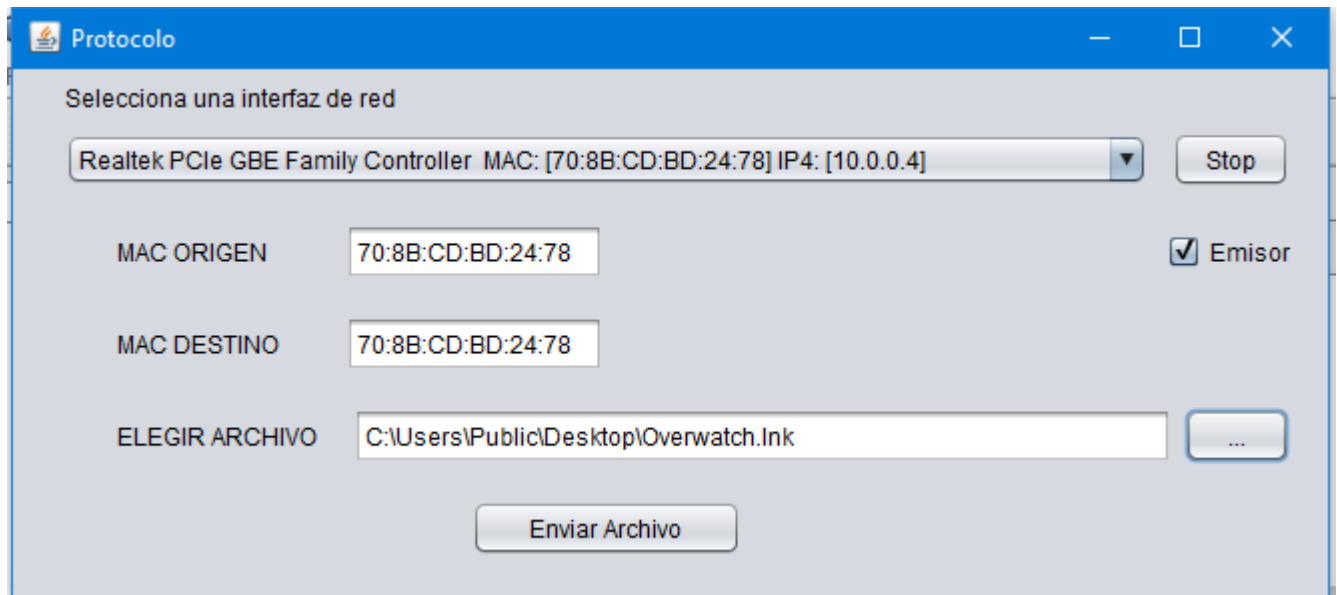
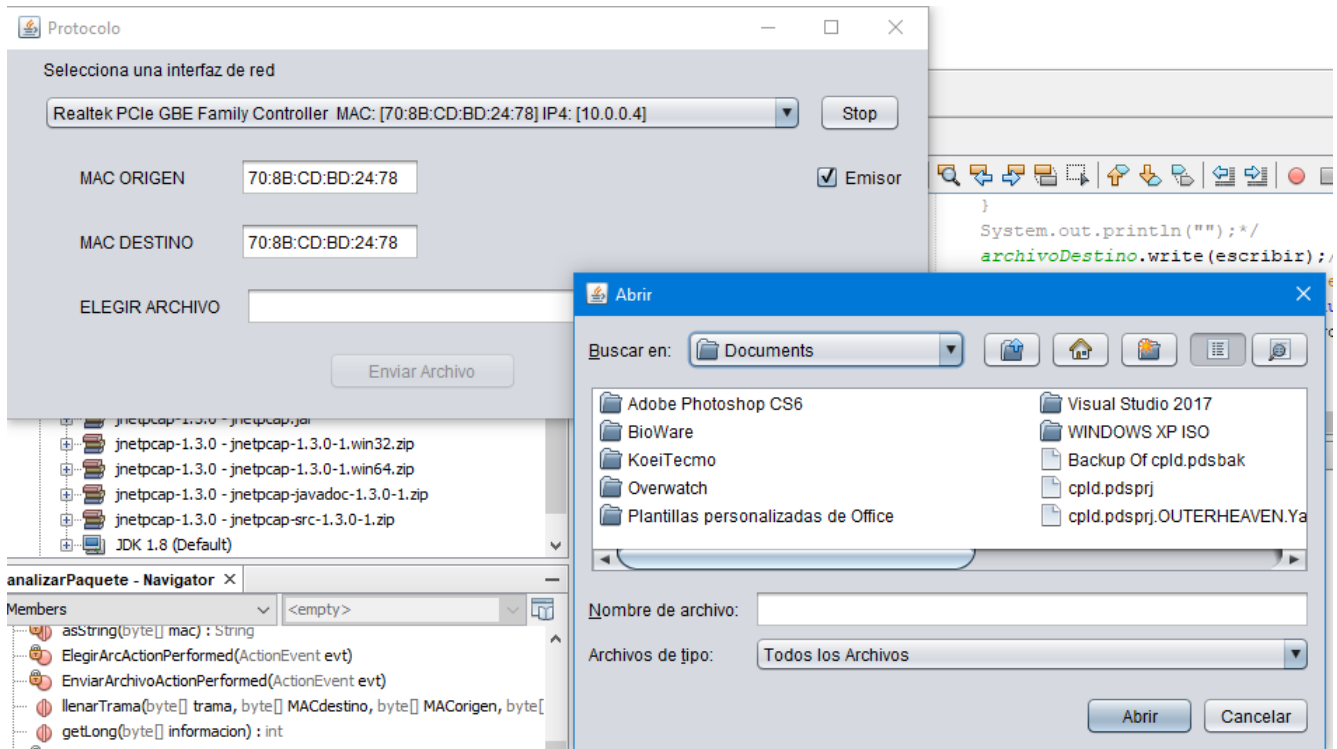
5. Funcionamiento

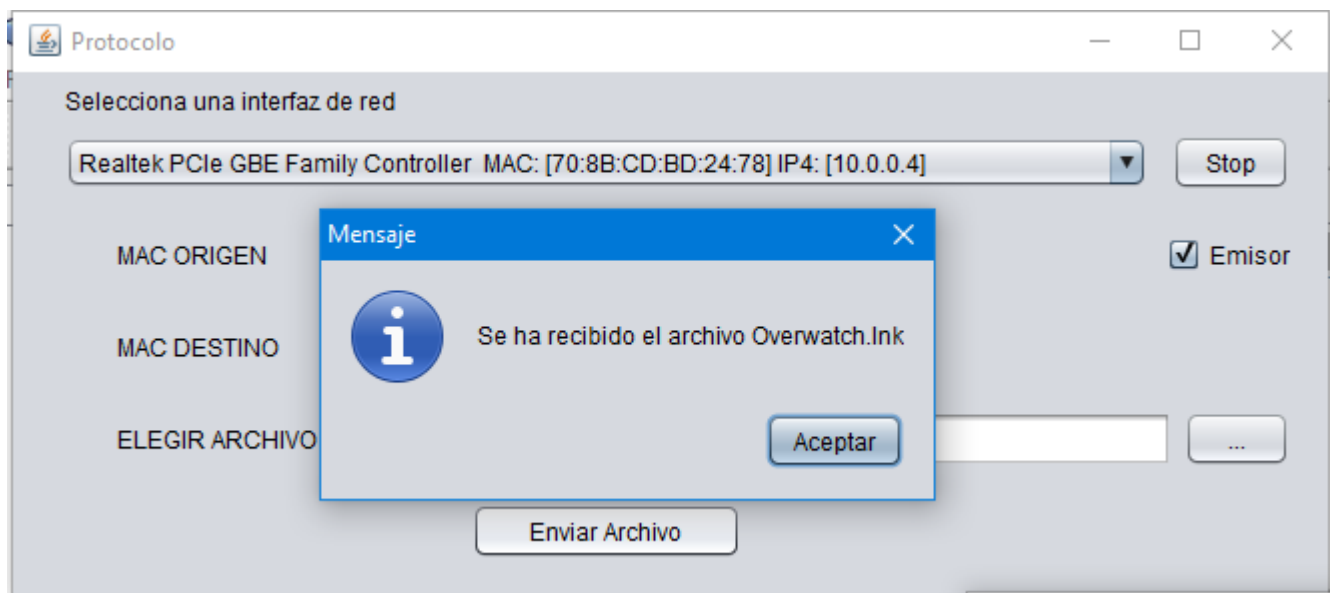
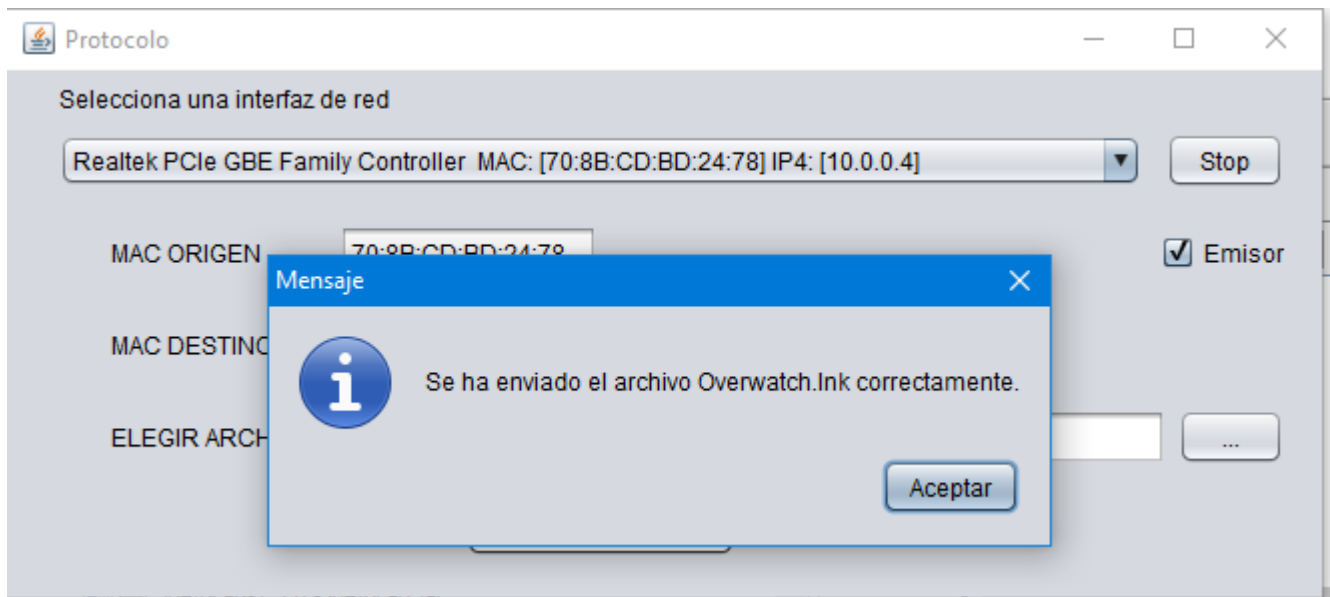


The screenshot shows a window titled "Protocolo" with a blue header bar. Below the header, the text "Selecciona una interfaz de red" is displayed. A dropdown menu shows "Realtek PCIe GBE Family Controller MAC: [70:8B:CD:BD:24:78] IP4: [10.0.0.4]". To the right of the dropdown is a "Start" button. Below the dropdown, there are three input fields: "MAC ORIGEN" (empty), "MAC DESTINO" (empty), and "ELEGIR ARCHIVO" (empty). To the right of the "MAC ORIGEN" field is a checkbox labeled "Emisor" which is unchecked. At the bottom center is a button labeled "Enviar Archivo".

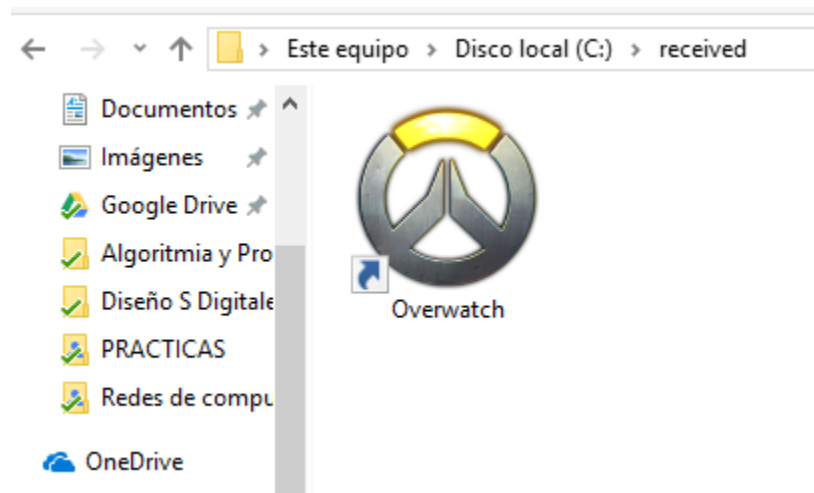


The screenshot shows the same "Protocolo" window, but with the following changes: The dropdown menu remains the same. The "Start" button has been replaced by a "Stop" button. The "MAC ORIGEN" field now contains the value "70:8B:CD:BD:24:78". The "MAC DESTINO" field now contains the value "70:8B:CD:BD:24:78". The "Emisor" checkbox is now checked. The "ELEGIR ARCHIVO" field remains empty. The "Enviar Archivo" button is still present at the bottom center.






```
NUMERO DE TRAMAS: 1  
4C 00 00 00 01 14 02 00 00 00 00 00 C0 00 00 00 00 00 46 9D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
70 8B CD BD 24 78  
SE ENVIO LA TRAMA  
  
Paquete recibido el Sun May 06 20:22:37 CDT 2018 bytes capturados=931 tam original=931  
NUMERO TOTAL DE TRAMAS: 1  
  
Trama: 1  
4C 00 00 00 01 14 02 00 00 00 00 00 C0 00 00 00 00 00 46 9D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
894  
  
Checksum.... OK  
  
Escribiendo en disco....  
  
Archivo recibido escrito en disco
```



Referencias

- [1] <https://es.ccm.net/contents/275-protocolo-de-comunicacion>.