ALUMNO:

DÍAZ MEDINA JESÚS KAIMORTS

ESQUIVEL VALDEZ ALBERTO

VARGAR ROMERO ERICK EFRÁIN

MATERÍA:

REDES COMPUTACIONALES

PROFESOR:

MORENO CERVANTES AXEL ERNESTO  
MR. PACKS

GRUPO:

3CM9

PRACTICA 1

**Introducción**

Una de las funciones más utilizadas en las aplicaciones actuales es la posibilidad de enviar archivos a través de la red.

La problemática a la que nos enfrentamos frecuentemente en este tipo de aplicaciones es que existen veces en donde el tamaño del archivo suele ser un impedimento, ya sea porque es demasiado grande o porque la aplicación no va dirigida directamente al envío de archivos.

En esta práctica se desarrollará una aplicación que nos permitirá enviar múltiples archivos, independientemente del tamaño de estos.

**Desarrollo**

Para el desarrollo de la práctica se utilizaron los códigos mostrados en clase, tanto el Servidor(Receptor) como el Cliente(Emisor).

Además, se implementó un sistema de Drag and Drop que nos permite seleccionar los archivos o carpetas arrastrándolos sobre la interfaz de la aplicación.

SERVIDOR:

1. **package** practica1;
3. **import** java.io.DataInputStream;
4. **import** java.io.DataOutputStream;
5. **import** java.io.File;
6. **import** java.io.FileOutputStream;
7. **import** java.net.ServerSocket;
8. **import** java.net.Socket;

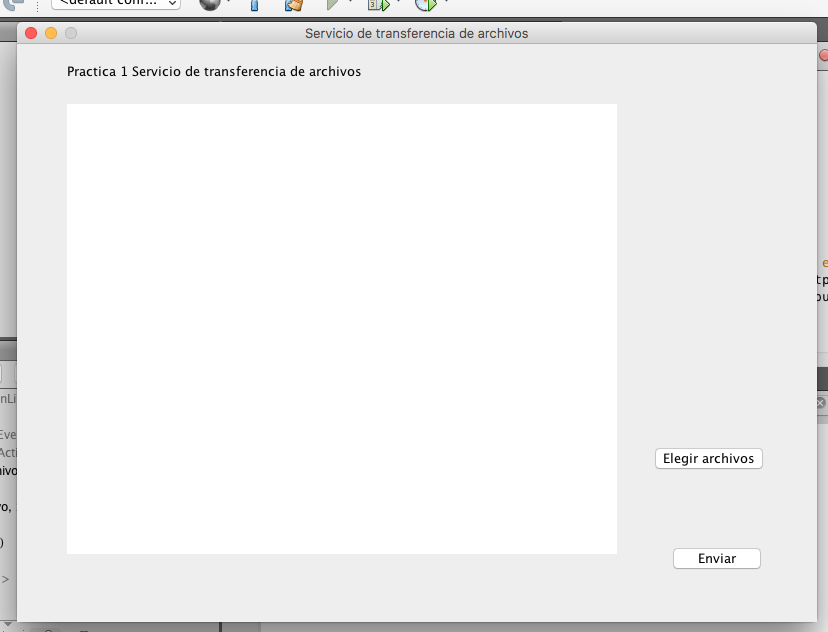
11. **public** **class** Practica1 {
12. **public** **static** **void** main(String [] args){
14. **try**{
15. **int** puerto=9999;
16. **int** n=0;
17. ServerSocket s= **new** ServerSocket(puerto);
18. System.out.println("Servidor iniciado");
19. **for**(;;){
20. Socket cl= s.accept();
21. System.out.println("Cliente conectado desde "+ cl.getInetAddress() + ":" + cl.getPort());
22. DataInputStream d= **new** DataInputStream(cl.getInputStream());
23. **boolean** directory=d.readBoolean();
24. **if**(directory){
25. String nombre=d.readUTF();
26. System.out.println("Recibi la carpeta "+nombre);
27. File mkd=**new** File(nombre);
28. **if** (!mkd.exists()){
29. mkd.mkdir();
30. System.out.println("Creando directorio " + mkd.toString());
31. }
32. }**else**{
33. String nombre=d.readUTF();
34. **long** tam=d.readLong();
35. DataOutputStream a= **new** DataOutputStream(**new** FileOutputStream(nombre));
36. **long** recibidos=0;
37. **while**(recibidos<tam){
38. **byte**[] b= **new** **byte**[1500];
39. n=d.read(b);
40. a.write(b,0,n);
41. a.flush();
42. recibidos=recibidos+n;
43. }
44. System.out.println("Archivo recibido");
45. a.close();
46. }
47. d.close();
48. cl.close();
49. }
50. }**catch**(Exception e){
51. e.printStackTrace();
52. }
53. }

}

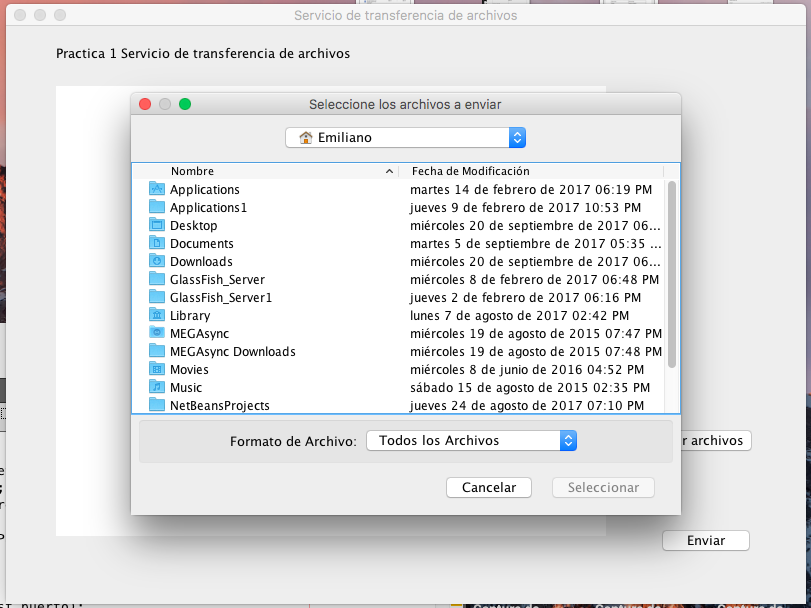
CLIENTE:

1. **package** main;
2. **import** java.awt.event.ActionEvent;
3. **import** java.awt.event.ActionListener;
4. **import** java.io.DataInputStream;
5. **import** java.io.DataOutputStream;
6. **import** java.io.File;
7. **import** java.io.FileInputStream;
8. **import** java.net.Socket;
9. **import** java.util.ArrayList;
10. **import** javax.swing.JButton;
11. **import** javax.swing.JFileChooser;
12. **import** javax.swing.JFrame;
13. **import** javax.swing.JLabel;
14. **import** javax.swing.JOptionPane;
15. **import** javax.swing.JScrollPane;
16. **import** javax.swing.JTextArea;
18. **public** **class** Practica1C **extends** JFrame **implements** ActionListener {
20. **private** JLabel texto;
21. **private** JTextArea area;
22. **private** JButton boton;
23. **private** JButton botonFC;
24. **private** JScrollPane scroll;
25. **private** ArrayList<File> archivos=**new** ArrayList<File>();
26. **private** **boolean** error=**false**;
28. **public** Practica1C() {
29. **super**();
30. configurarVentana();
31. inicializarComponentes();
32. }
34. **private** **void** configurarVentana() {
35. **this**.setTitle("Servicio de transferencia de archivos");
36. **this**.setSize(800, 600);
37. **this**.setLocationRelativeTo(**null**);
38. **this**.setLayout(**null**);
39. **this**.setResizable(**false**);
40. **this**.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
41. }
43. **private** **void** inicializarComponentes() {
44. texto = **new** JLabel();
45. boton = **new** JButton();
46. botonFC = **new** JButton();
47. area = **new** JTextArea();
48. texto.setText("Practica 1 Servicio de transferencia de archivos");
49. texto.setBounds(50, 15, 300, 25);
50. boton.setText("Enviar");
51. boton.setBounds(650, 500, 100, 30);
52. boton.addActionListener(**new** java.awt.event.ActionListener() {
53. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) {
54. actionPerformedEnviar(evt);
55. }
56. });
57. botonFC.setText("Elegir archivos");
58. botonFC.setBounds(632, 400, 120, 30);  // (x, y, ancho, alto)
59. botonFC.addActionListener(**this**);
60. area.setBounds(50, 60, 550, 450);
61. area.setEditable(**false**);
62. scroll=**new** JScrollPane(area);
63. **this**.add(area);
64. **this**.add(texto);
65. **this**.add(boton);
66. **this**.add(botonFC);
67. **new** FileDrop( area, **new** FileDrop.Listener(){
68. **public** **void** filesDropped( java.io.File[] files ){
69. **for**( **int** i=0; i < files.length; i++ ){
70. **if**(!files[i].isDirectory()){
71. archivos.add(files[i]);
72. area.append(archivos.get(archivos.size()-1).getName() + " (Archivo)\n" );
73. }**else**{
74. archivos.add(files[i]);
75. area.append(archivos.get(archivos.size()-1).getName() + " (Directorio)\n" );
76. }
77. }
78. }
79. });
81. }
83. **public** **void** actionPerformedEnviar(ActionEvent e) {
84. enviarArchivos();
85. }
87. **public** **static** **void** main(String[] args) {
88. Practica1C V = **new** Practica1C();
89. V.setVisible(**true**);
90. }
92. **private** **void** enviarArchivos() {
93. **while**(!archivos.isEmpty()){
94. enviarArchivo(archivos.get(0),"");
95. archivos.remove(0);
96. }
97. area.setText("");
98. **if**(error){
99. JOptionPane.showMessageDialog(**null**, "Error al enviar los archivos");
100. }**else**{
101. JOptionPane.showMessageDialog(**null**, "Archivos enviados");
102. }
103. }
105. **private** **void** enviarArchivo(File archivo,String rutad) {
106. **try** {
107. **if**(archivo.isDirectory()){
108. rutad+= archivo.getName() +"/";
109. String host="127.0.0.1";
110. **int** puerto=9999;
111. Socket cl = **new** Socket(host,puerto);
112. System.out.println("Conexion establecida...Comienza el envio de la carpeta: "+rutad);
113. DataOutputStream dos= **new** DataOutputStream(cl.getOutputStream());
114. dos.writeBoolean(**true**);
115. dos.writeUTF(rutad);
116. File[] archivosd=archivo.listFiles();
117. **for** (**int** i = 0; i < archivosd.length; i++) {
118. enviarArchivo(archivosd[i],rutad);
119. }
120. }**else**{
121. File f=**new** File(archivo.getCanonicalPath());
122. String nombre=f.getName();
123. String enviarA=rutad+nombre;
124. **long** tam=f.length();
125. String ruta=f.getAbsolutePath();
126. String host="127.0.0.1";
127. **int** puerto=9999;
128. **int** n=0, porcentaje=0;
129. Socket cl = **new** Socket(host,puerto);
130. System.out.println("Conexion establecida...Comienza el envio del archivo: "+ruta);
131. DataOutputStream dos= **new** DataOutputStream(cl.getOutputStream());
132. DataInputStream dis= **new** DataInputStream(**new** FileInputStream(ruta));
133. dos.writeBoolean(**false**);
134. dos.writeUTF(enviarA);
135. dos.flush();
136. dos.writeLong(tam);
137. **int** enviados=0;
138. **while**(enviados<tam){
139. **byte**[] b= **new** **byte**[1500];
140. n=dis.read(b);
141. dos.write(b,0,n);
142. dos.flush();
143. enviados=enviados+n;
144. porcentaje=(**int**)((enviados\*100)/tam);
145. System.out.println("Se transmitio el "+ porcentaje + " %");
146. }
147. System.out.println("Archivo enviado");
148. }
149. } **catch** (Exception e) {
150. error=**true**;
151. e.printStackTrace();
152. }
154. }
156. @Override
157. **public** **void** actionPerformed(ActionEvent ae) {
158. JFileChooser chooser=**new** JFileChooser();
159. **int** sel;
160. chooser.setMultiSelectionEnabled(**true**);
161. chooser.setFileSelectionMode(JFileChooser.FILES\_AND\_DIRECTORIES);
162. chooser.setDialogTitle("Seleccione los archivos a enviar");
163. sel=chooser.showOpenDialog(**this**);
164. **if**(sel==JFileChooser.APPROVE\_OPTION){
165. File[] archivosSel=chooser.getSelectedFiles();
166. agregarALista(archivosSel);
167. }
168. }
170. **private** **void** agregarALista(File[] archivosSel) {
171. **int** i=0;
172. **while**(i<archivosSel.length){
173. archivos.add(archivosSel[i]);
174. **if**(archivosSel[i].isDirectory()){
175. area.append(archivosSel[i].getName() + "(Directorio)\n");
176. }**else**{
177. area.append(archivosSel[i].getName() + "(Archivo)\n");
178. }
179. i++;
180. }
181. }
182. }

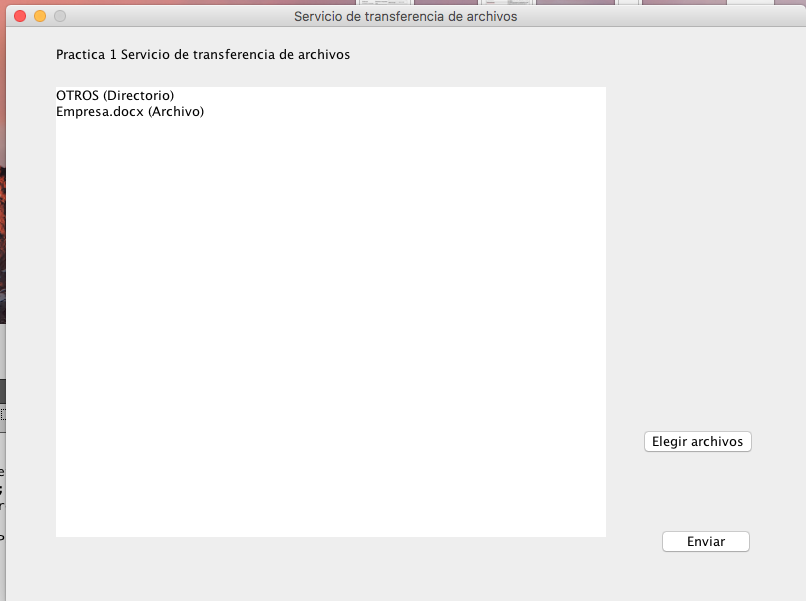
Se muestra la zona de drag and drop en pantalla.

****

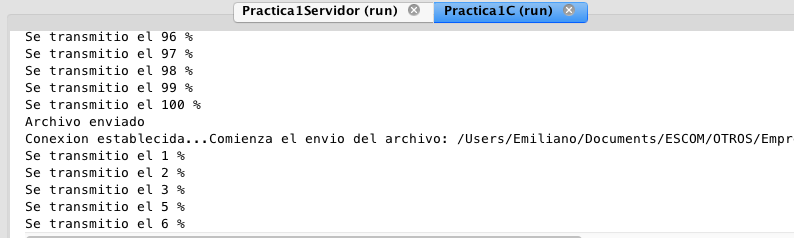
También se pueden seleccionar los archivos mediante un FileChooser.

****

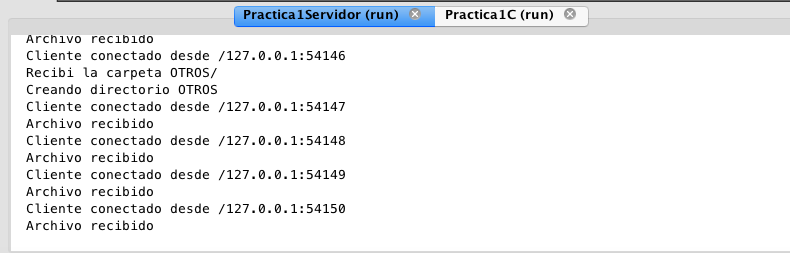
Una vez seleccionados los archivos se presiona el botón de enviar.

****

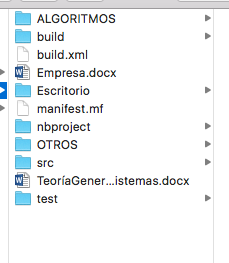
En el cliente se muestran los archivos enviados y el porcentaje de datos transmitidos.

****

En el servidor se muestra cuando se recibe el archivo y el origen de este.

****

Finalmente podemos comprobar que los archivos se transmitieron.

****

Conclusiones:

En esta práctica implementamos una aplicación que envía múltiples archivos a través de la red. Esto nos sirvió para darnos una primera idea de cómo funciona este proceso de transferencia, y de igual manera pudimos observar y aplicar los conceptos vistos en clase como los sockets, sockets bloqueantes, etc. También nos dimos cuenta que los archivos deben de ser enviados en orden y que puede que se pierda un pedazo de estos y es por ello que debemos verificar si llegaron en orden y completos.