Instituto Politécnico Nacional.

Escuela Superior De Cómputo.

Materia:

Aplicaciones Para Comunicaciones en Red.

Tema:

Reporte

(Práctica 3)

Profesor:

Axel Ernesto Moreno Cervantes.

Alumnos:

Luis Enrique Rojas Alvarado.

Mario Alberto Miranda Sandoval.

Grupo:

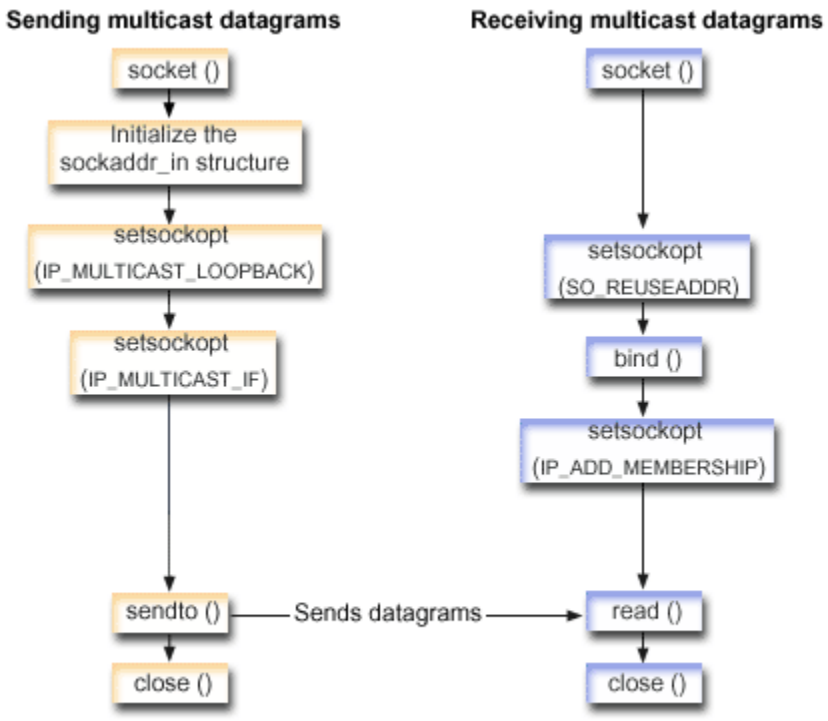
3CM5.

Introducción.

A veces nos interesa que un ordenador pueda enviar un mensaje por red y que este sea recibido por otros ordenadores simultáneamente. Para ello están las direcciones multicast.

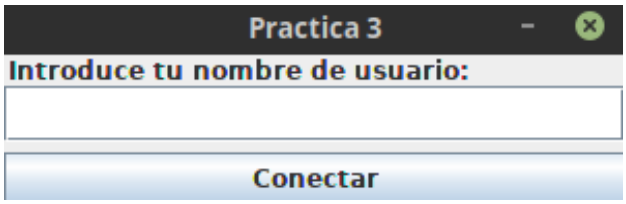
Son direcciones en el rango 224.0.0.0 a 239.255.255.255. La 224.0.0.0 está reservada y no puede usarse. Enviando mensajes por estas direcciones, cualquier otro ordenador en la red que las escuche podría leer dicho mensaje, independientemente de cuál sea la IP real de ese ordenador.

Es decir, si un ordenador quiere enviar un mensaje simultáneamente a varios, puede hacerlo enviando el mensaje a una de estas IPs, los demás ordenadores deben estar a la escucha de dichas IPs para recibir el mensaje.

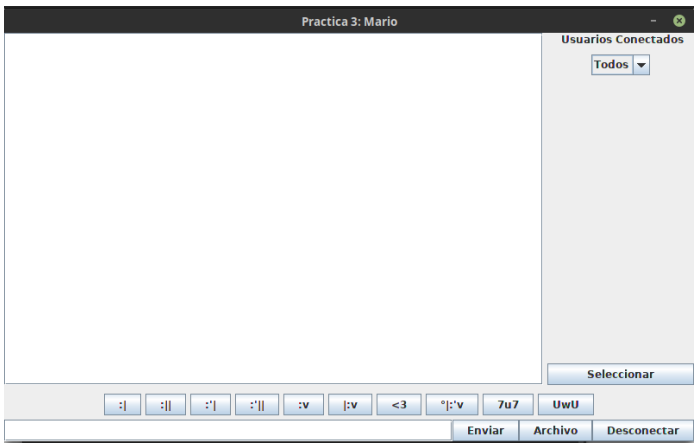


Desarrollo.

Para el desarrollo de esta práctica se genero los siguiente, primero una interfaz donde se coloca el nombre del usuario, mientras que la segunda es el chat principal.



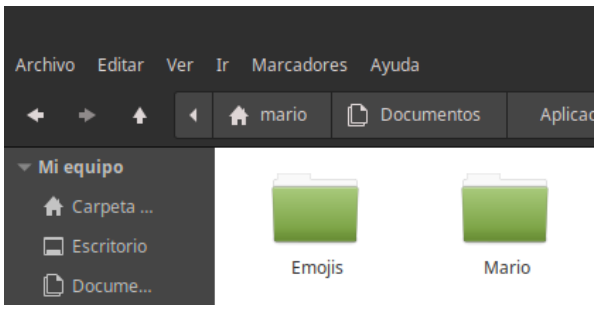
Interfaz de introducción de nombre.



Interfaz del chat.

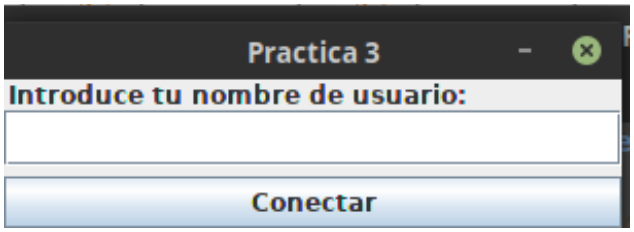
Una vez dado esto el usuario está en posición de entablar conversaciones en la sala, mandar y recibir archivos, y por último mandar mensajes privados.

Una consideración más es, debido al control de los usuarios el programa tiene la facultad de crear carpetas de usuario de acuerdo al nombre de este.

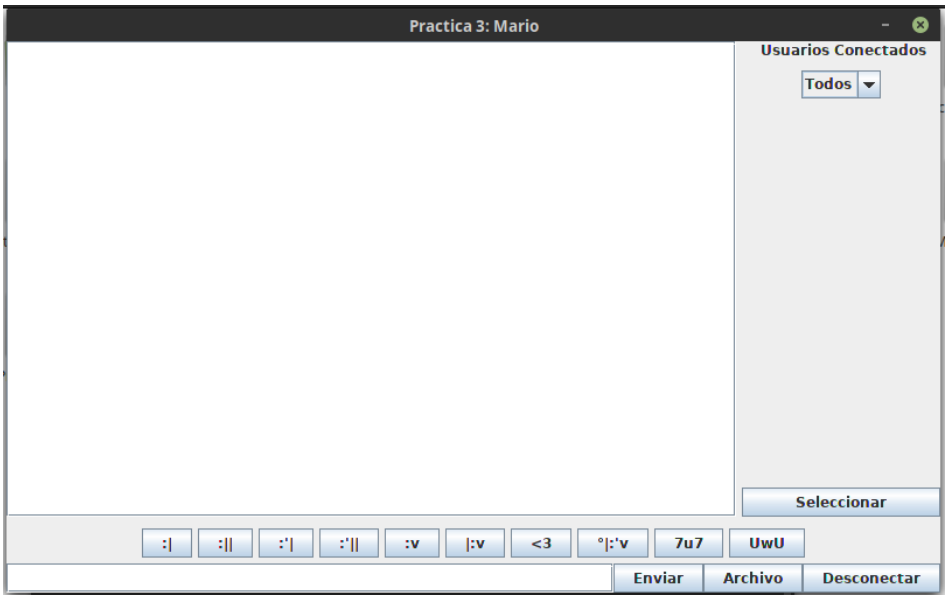


Carpeta Creada para el usuario **Mario**.

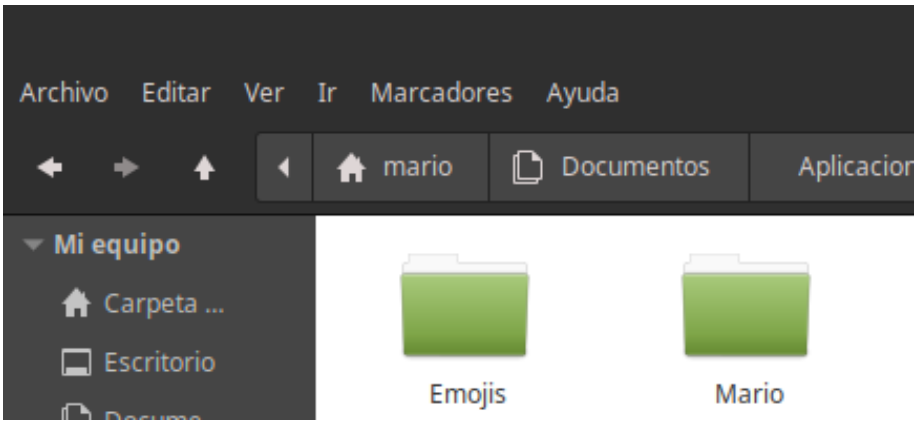
Pruebas.



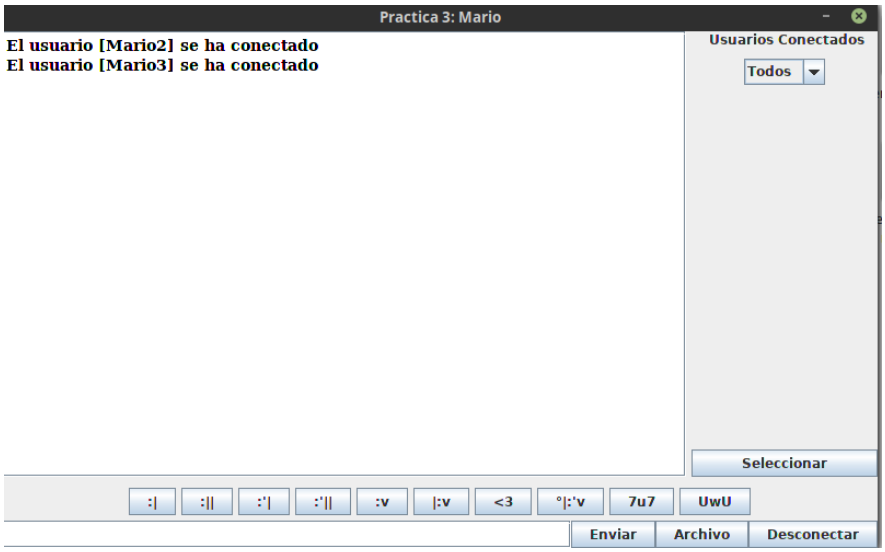
Usuario Registrado.



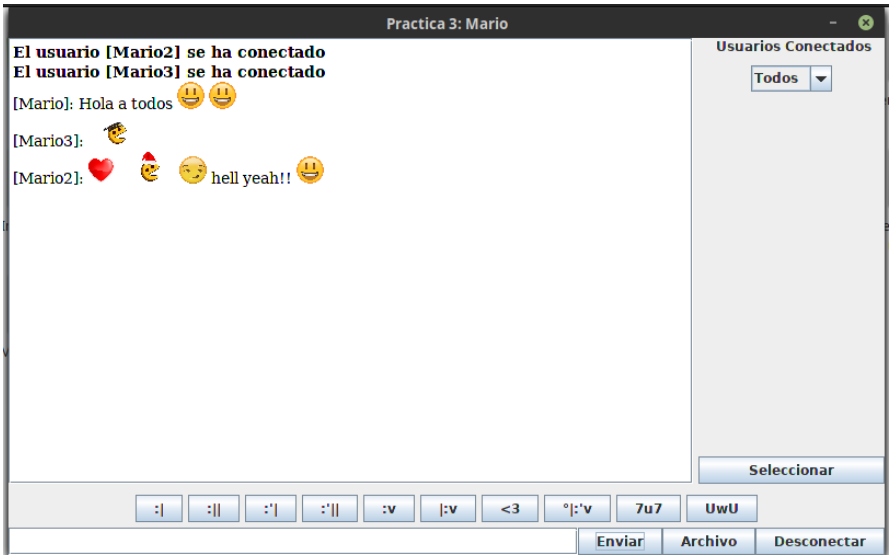
Interfaz de usuario principal.



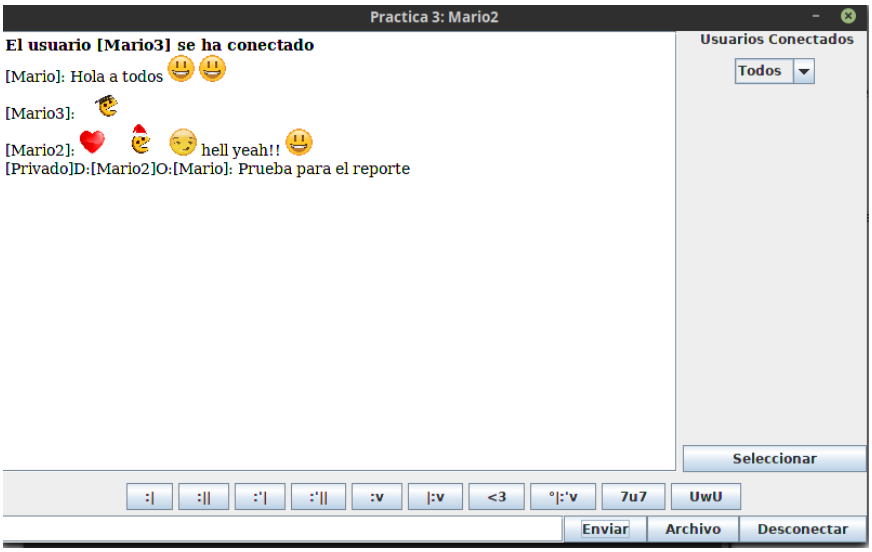
Carpeta de usuario creada.



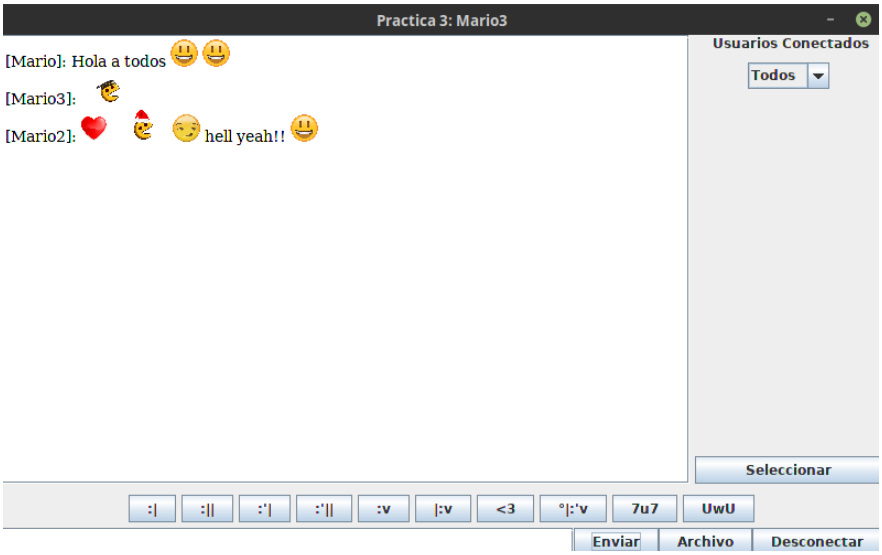
Anuncio de nuevos usuarios.



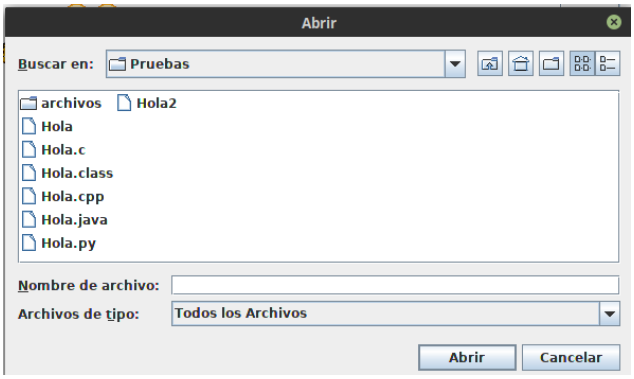
Envió de mensajes.



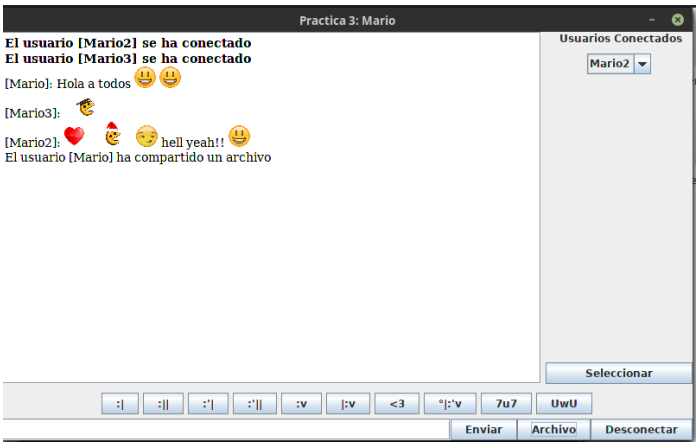
Envió de mensajes privados.



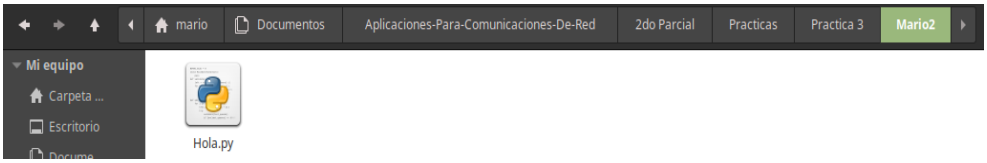
Comprobación de mensajes privados.

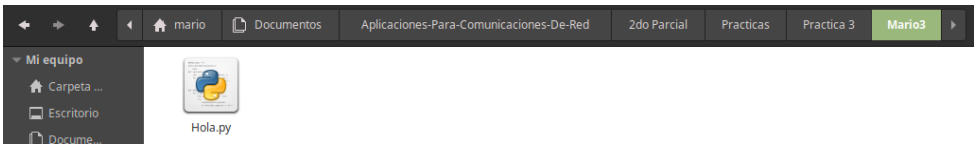


Enviando Archivos.



Anuncio de envió de archivos.





Comprobación del envío de archivos.

Código.

Principal.java

```java
import javax.swing.*;
import java.awt.BorderLayout;

public class Principal extends JFrame {
    private static final long serialVersionUID = 1L;

    public Principal() {
        setBounds(450, 150, 300, 100);
        setResizable(false);
        setTitle("Practica 3");

        panelPrincipal = new JPanel();
        panelCentral = new JPanel();
        panelPrincipal.setLayout(new BorderLayout(5, 5));
        panelCentral.setLayout(new BoxLayout(this.panelCentral, BoxLayout.Y_AXIS))
;
        nombreUsuario = new JLabel("Introduce tu nombre de usuario:");
        campoUsuario = new JTextField(30);
        botonConectar = new JButton("Conectar");

        botonConectar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent e) {
                if (!campoUsuario.getText().equals("")) {
                    setVisible(false);
                    new Interfaz(HOST, PUERTO, campoUsuario.getText().trim());
                } else
                    JOptionPane.showMessageDialog(Principal.this, "Nombre de usuar
io Vacio", "Error", JOptionPane.ERROR_MESSAGE);
            }
        });

        panelCentral.add(nombreUsuario);
        panelCentral.add(campoUsuario);
        panelPrincipal.add(panelCentral, BorderLayout.CENTER);
        panelPrincipal.add(botonConectar, BorderLayout.SOUTH);
        add(panelPrincipal);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }

    public static void main(String[] args) {
        new Principal();
    }

    private final String HOST = "230.1.1.1";
    private final int PUERTO = 9000;
    private JPanel panelPrincipal;
    private JPanel panelCentral;
    private JLabel nombreUsuario;
    private JTextField campoUsuario;
    private JButton botonConectar;
}
```

Interfaz.java

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Interfaz extends JFrame {
    private static final long serialVersionUID = 2L;

    public Interfaz(String host, int puerto, String nombre) {
        // -----------------Recibiendo Parametros------------------//
        this.host = host;
        this.puerto = puerto;
        this.nombre = nombre;
        // ----------------Creando Interfaz-------------------------//
        setBounds(325, 100, 800, 500);
        setTitle("Practica 3: " + nombre);
        setResizable(false);

        panelPrincipal = new JPanel();
        panelCentral = new JPanel();
        panelInferior = new JPanel();
        panelEmojis = new JPanel();
        panelFunciones = new JPanel();
        panelUsuarios = new JPanel();
        panelCombo = new JPanel();
        editor = new JEditorPane("text/html", null);
        editor.setEditable(false);
        areaMensaje = new JTextArea();
        areaMensaje.setLineWrap(true);
        botonesEmojis = new JButton[textoBotonesEmojis.length];
        enviar = new JButton("Enviar");
        archivo = new JButton("Archivo");
        desconectar = new JButton("Desconectar");
        seleccion = new JButton("Seleccionar");
        usuariosConectados = new JLabel("    Usuarios Conectados   ");
        escuchaEmojis = new ManejoEmojis();
        usuarioConectado = new JComboBox<>();
        usuarioConectado.addItem("Todos");

        enviar.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                miCliente.enviar(new Mensaje("[" + nombre + "]: " + areaMensaje.ge
tText(), nombre, "", 1));
                areaMensaje.setText("");
            }
        });

        archivo.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JFileChooser jf = new JFileChooser();
                jf.requestFocus();
                int r = jf.showOpenDialog(Interfaz.this);
                if (r == JFileChooser.APPROVE_OPTION) {
                    miCliente.enviarArchivo(jf.getSelectedFile());
                    String mensaje = "El usuario [" + nombre + "] ha compartido un
 archivo";
                    miCliente.enviar(new Mensaje(mensaje, nombre, "", 1));
                }
            }
```

```
57.         });
58.
59.         seleccion.addActionListener(new ActionListener() {
60.             public void actionPerformed(ActionEvent e) {
61.                 miCliente.enviar(new Mensaje("[" + nombre + "]: " + areaMensaje.ge
    tText(), nombre,
62.                 (String) usuarioConectado.getSelectedItem(), 4));
63.                 areaMensaje.setText("");
64.             }
65.         });
66.
67.         panelPrincipal.setLayout(new BorderLayout(5, 5));
68.         panelCentral.setLayout(new BorderLayout(5, 5));
69.         panelInferior.setLayout(new BoxLayout(this.panelInferior, BoxLayout.Y_AXIS
    ));
70.         panelFunciones.setLayout(new BoxLayout(this.panelFunciones, BoxLayout.X_AX
    IS));
71.         panelUsuarios.setLayout(new BorderLayout(5, 5));
72.
73.         colocarBotones();
74.         addWindowListener(new CorreCliente());
75.         panelCombo.add(usuarioConectado);
76.         panelUsuarios.add(usuariosConectados, BorderLayout.NORTH);
77.         usuariosConectados.setAlignmentX(SwingConstants.CENTER);
78.         panelUsuarios.add(panelCombo, BorderLayout.CENTER);
79.         panelUsuarios.add(seleccion, BorderLayout.SOUTH);
80.         panelCentral.add(new JScrollPane(editor), BorderLayout.CENTER);
81.         panelCentral.add(panelUsuarios, BorderLayout.EAST);
82.         panelFunciones.add(new JScrollPane(areaMensaje));
83.         panelFunciones.add(enviar);
84.         panelFunciones.add(archivo);
85.         panelFunciones.add(desconectar);
86.         panelInferior.add(panelEmojis);
87.         panelInferior.add(panelFunciones);
88.         panelPrincipal.add(panelCentral, BorderLayout.CENTER);
89.         panelPrincipal.add(panelInferior, BorderLayout.SOUTH);
90.         add(panelPrincipal);
91.
92.         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
93.         setVisible(true);
94.     }
95.
96.     private void colocarBotones() {
97.         for(int i = 0; i < botonesEmojis.length; i += 1) {
98.             botonesEmojis[i] = new JButton("" + textoBotonesEmojis[i]);
99.             botonesEmojis[i].addActionListener(escuchaEmojis);
100.                panelEmojis.add(botonesEmojis[i]);
101.            }
102.        }
103.
104.        private class ManejoEmojis implements ActionListener {
105.            public void actionPerformed(ActionEvent e) {
106.                areaMensaje.append(" " + e.getActionCommand() + " ");
107.            }
108.        }
109.
110.        private class CorreCliente extends WindowAdapter {
111.            public void windowOpened(WindowEvent we) {
112.                System.out.println("Ventana abierta");
113.                miCliente = new Cliente(nombre, host, puerto, editor, usuarioC
    onectado);
```

```
                    miCliente.saludo(nombre);
                }
            }

            private String host;
            private int puerto;
            private String nombre;
            private JPanel panelPrincipal;
            private JPanel panelCentral;
            private JPanel panelInferior;
            private JPanel panelEmojis;
            private JPanel panelFunciones;
            private JPanel panelUsuarios;
            private JEditorPane editor;
            private JButton[] botonesEmojis;
            private String[] textoBotonesEmojis = {
                ":|",
                ":||",
                ":'|",
                ":'||",
                ":v",
                "|:v",
                "<3",
                "°|:'v",
                "7u7",
                "UwU"
            };
            private JTextArea areaMensaje;
            private JButton enviar;
            private JButton archivo;
            private JButton desconectar;
            public static JComboBox<String> usuarioConectado;
            private JButton seleccion;
            private JLabel usuariosConectados;
            private ActionListener escuchaEmojis;
            private Cliente miCliente;
            private JPanel panelCombo;
        }
```

Mensaje.java

```
1.  import java.io.*;
2.
3.  public class Mensaje implements Serializable {
4.      private static final long serialVersionUID = 3L;
5.      public Mensaje(String mensaje, String usuarioOrigen, String usuarioDestino, in
    t tipo) {
6.          this.mensaje = mensaje;
7.          this.usuarioOrigen = usuarioOrigen;
8.          this.usuarioDestino = usuarioDestino;
9.          this.tipo = tipo;
10.     }
11.
12.     public Mensaje(String nombreArchivo, String usuarioOrigen, String usuarioDesti
    no, int tipo, long tamanio, String ruta, int np) {
13.         this.nombreArchivo = nombreArchivo;
14.         this.usuarioOrigen = usuarioOrigen;
15.         this.usuarioDestino = usuarioDestino;
16.         this.tamanio = tamanio;
17.         this.ruta = ruta;
```

```
        this.np = np;
        this.tipo = tipo;
    }

    public String getMensaje() { return mensaje; }
    public String getUsuarioOrigen() { return usuarioOrigen; }
    public String getUsuarioDestino() { return usuarioDestino; }
    public int getTipo() { return tipo; }

    public void setMensaje(String mensaje) { this.mensaje = mensaje; }
    public void setUsuarioOrigen(String usuarioOrigen) { this.usuarioOrigen = usua
rioOrigen; }
    public void setUsuarioDestino(String usuarioDestino) { this.usuarioDestino = u
suarioDestino; }
    public void setTipo(int tipo) { this.tipo = tipo; }

    public String getNombre() { return nombreArchivo; }
    public long getTamanio() { return tamanio; }
    public String ruta() { return ruta; }
    public int getNp() { return np; }
    public byte[] getDatos() { return datos; }
    public int getBytesEnviados() { return bytesEnviados; }

    public void setDatos(byte[] datos) { this.datos = datos; }
    public void setBytesEnviados(int bytesEnviados) { this.bytesEnviados = bytesEn
viados; }

    private String mensaje;
    private String usuarioOrigen;
    private String usuarioDestino;
    private int tipo;
    private String nombreArchivo;
    private long tamanio;
    private String ruta;
    private int np;
    private byte[] datos;
    private int bytesEnviados;
}
```

Cliente.java

```
import java.net.*;
import java.io.*;
import javax.swing.*;
import javax.swing.text.html.HTMLEditorKit;
import java.util.*;
import java.util.regex.*;

public class Cliente {
    public Cliente(String nombre, String host, int puerto, JEditorPane editor, JCo
mboBox<String> usuarioConectado) {
        this.nombre = nombre;
        this.host = host;
        this.puerto = puerto;
        this.editor = editor;
        this.usuarioConectado = usuarioConectado;

        try {
            cliente = new MulticastSocket(puerto);
            grupo = InetAddress.getByName(host);
```

```
19.             cliente.joinGroup(grupo);
20.         } catch (Exception e) {
21.             e.printStackTrace();
22.         }
23.
24.         hiloEscucha = new EscuchaMensajes();
25.         escucha = new Thread(hiloEscucha);
26.         escucha.start();
27.         carpeta();
28.     }
29.
30.     private class EscuchaMensajes implements Runnable {
31.         public void run() {
32.             System.out.println("Escuchando Mensajes");
33.                 try {
34.                     DatagramPacket recibido = new DatagramPacket(new byte[6500], 6
    500);
35.                     while (true) {
36.                         cliente.receive(recibido);
37.                         ObjectInputStream ois = new ObjectInputStream(new ByteArra
    yInputStream(recibido.getData()));
38.                         Mensaje msj = (Mensaje) ois.readObject();
39.                         if(msj.getTipo() == 0 && !msj.getUsuarioOrigen().equals(no
    mbre)) {
40.                             byte[] bmsj = msj.getMensaje().getBytes();
41.                             byte[] busuario = msj.getUsuarioOrigen().getBytes();
42.                             String mensaje = new String(bmsj, 0, msj.getMensaje().
    length());
43.                             String usuario = new String(busuario, 0, msj.getUsuari
    oOrigen().length());
44.                             HTMLEditorKit kit = (HTMLEditorKit) editor.getEditorKi
    t();
45.                             StringReader reader = new StringReader(mensaje);
46.                             kit.read(reader, editor.getDocument(), editor.getDocum
    ent().getLength());
47.                             usuarioConectado.addItem(usuario);
48.                             ByteArrayOutputStream baos = new ByteArrayOutputStream
    ();
49.                             ObjectOutputStream oos = new ObjectOutputStream(baos);

50.                             msj.setTipo(3);
51.                             msj.setUsuarioDestino(msj.getUsuarioOrigen());
52.                             msj.setUsuarioOrigen(nombre);
53.                             oos.writeObject(msj);
54.                             oos.flush();
55.                             byte[] b = baos.toByteArray();
56.                             DatagramPacket re = new DatagramPacket(b, b.length, gr
    upo, puerto);
57.                             cliente.send(re);
58.                         } else if(msj.getTipo() == 1) {
59.                             byte[] bmsj = msj.getMensaje().getBytes();
60.                             String mensaje = new String(bmsj, 0, msj.getMensaje().
    length());
61.                             HTMLEditorKit kit = (HTMLEditorKit) editor.getEditorKi
    t();
62.                             StringReader reader = new StringReader(mensaje);
63.                             kit.read(reader, editor.getDocument(), editor.getDocum
    ent().getLength());
64.                         } else if(msj.getTipo() == 2) {
65.                             if(!msj.getUsuarioOrigen().equals(nombre))
66.                                 recibirArchivo(msj);
```

```
67.                         }else if(msj.getTipo() == 3 && !msj.getUsuarioOrigen().equ
    als(nombre) && msj.getUsuarioDestino().equals(nombre)) {
68.                             byte[] busuario = msj.getUsuarioOrigen().getBytes();
69.                             String usuario = new String(busuario, 0, msj.getUsuari
    oOrigen().length());
70.                             usuarioConectado.addItem(usuario);
71.                         } else if(msj.getTipo() == 4 && msj.getUsuarioDestino().eq
    uals(nombre) && !msj.getUsuarioOrigen().equals(nombre)) {
72.                             byte[] bmsj = msj.getMensaje().getBytes();
73.                             String mensaje = new String(bmsj, 0, msj.getMensaje().
    length());
74.                             mensaje = "[Privado]D:[" + nombre + "]O:" + mensaje;
75.                             HTMLEditorKit kit = (HTMLEditorKit) editor.getEditorKi
    t();
76.                             StringReader reader = new StringReader(mensaje);
77.                             kit.read(reader, editor.getDocument(), editor.getDocum
    ent().getLength());
78.                         }
79.                         ois.close();
80.                     }
81.                 } catch (Exception e) {
82.                     e.printStackTrace();
83.                 }
84.         }
85.     }
86.
87.     private class EnviaMensajes implements Runnable {
88.         private EnviaMensajes(Mensaje msj) {
89.             this.msj = msj;
90.         }
91.
92.         public void run() {
93.             // editor.setText("<b>Enviando Mensajes</b>" + (++i));
94.             try {
95.                 ByteArrayOutputStream baos = new ByteArrayOutputStream();
96.                 ObjectOutputStream oos = new ObjectOutputStream(baos);
97.                 oos.writeObject(msj);
98.                 oos.flush();
99.                 byte[] msj = baos.toByteArray();
100.                        DatagramPacket p = new DatagramPacket(msj, msj.length, gru
    po, puerto);
101.                        cliente.send(p);
102.                        oos.close();
103.                        baos.close();
104.                    } catch (IOException e) {
105.                        e.printStackTrace();
106.                    }
107.                }
108.
109.                private Mensaje msj;
110.            }
111.
112.            public void enviar(Mensaje msj) {
113.                p = Pattern.compile("UwU");
114.                m = p.matcher(msj.getMensaje());
115.
116.                if(m.find()) {
117.                    System.out.println("Se encontro UwU");
118.                    String imgsrc = Cliente.class.getClassLoader().getSystemResour
    ce("./Emojis/uwu.png").toString();
```

```
119.                     String img = "<img src = '" + imgsrc + "' width = 50 height =
    50>";
120.                     String cad = msj.getMensaje().replace("UwU", img);
121.                     msj.setMensaje(cad);
122.                 }
123.
124.                 p = Pattern.compile("7u7");
125.                 m = p.matcher(msj.getMensaje());
126.
127.                 if(m.find()) {
128.                     System.out.println("Se encontro 7u7");
129.                     String imgsrc = Cliente.class.getClassLoader().getSystemResour
    ce("./Emojis/7u7.png").toString();
130.                     String img = "<img src = '" + imgsrc + "' width = 25 height =
    25>";
131.                     String cad = msj.getMensaje().replace("7u7", img);
132.                     msj.setMensaje(cad);
133.                 }
134.
135.                 p = Pattern.compile("°|:'v");
136.                 m = p.matcher(msj.getMensaje());
137.
138.                 if(m.find()) {
139.                     System.out.println("Se encontro °<:{v");
140.                     String imgsrc = Cliente.class.getClassLoader().getSystemResour
    ce("./Emojis/navidad.png").toString();
141.                     String img = "<img src = '" + imgsrc + "' width = 50 height =
    30>";
142.                     String cad = msj.getMensaje().replace("°|:'v", img);
143.                     msj.setMensaje(cad);
144.                 }
145.
146.                 p = Pattern.compile("<3");
147.                 m = p.matcher(msj.getMensaje());
148.
149.                 if(m.find()) {
150.                     System.out.println("Se encontro <3");
151.                     String imgsrc = Cliente.class.getClassLoader().getSystemResour
    ce("./Emojis/cora.png").toString();
152.                     String img = "<img src = '" + imgsrc + "' width = 25 height =
    25>";
153.                     String cad = msj.getMensaje().replace("<3", img);
154.                     msj.setMensaje(cad);
155.                 }
156.
157.                 p = Pattern.compile("|:v");
158.                 m = p.matcher(msj.getMensaje());
159.
160.                 if(m.find()) {
161.                     System.out.println("Se encontro {:v");
162.                     String imgsrc = Cliente.class.getClassLoader().getSystemResour
    ce("./Emojis/jackie.png").toString();
163.                     String img = "<img src = '" + imgsrc + "' width = 50 height =
    30>";
164.                     String cad = msj.getMensaje().replace("|:v", img);
165.                     msj.setMensaje(cad);
166.                 }
167.
168.                 p = Pattern.compile(":v");
169.                 m = p.matcher(msj.getMensaje());
170.
```

```
171.                if(m.find()) {
172.                    System.out.println("Se encontro :v");
173.                    String imgsrc = Cliente.class.getClassLoader().getSystemResour
    ce("./Emojis/pacman.png").toString();
174.                    String img = "<img src = '" + imgsrc + "' width = 25 height =
    25>";
175.                    String cad = msj.getMensaje().replace(":v", img);
176.                    msj.setMensaje(cad);
177.                }
178.
179.                p = Pattern.compile(":'||");
180.                m = p.matcher(msj.getMensaje());
181.
182.                if(m.find()) {
183.                    System.out.println("Se encontro :'(");
184.                    String imgsrc = Cliente.class.getClassLoader().getSystemResour
    ce("./Emojis/lagrima.png").toString();
185.                    String img = "<img src = '" + imgsrc + "' width = 25 height =
    25>";
186.                    String cad = msj.getMensaje().replace(":'||", img);
187.                    msj.setMensaje(cad);
188.                }
189.
190.                p = Pattern.compile(":'|");
191.                m = p.matcher(msj.getMensaje());
192.
193.                if(m.find()) {
194.                    System.out.println("Se encontro :')");
195.                    String imgsrc = Cliente.class.getClassLoader().getSystemResour
    ce("./Emojis/risas.png").toString();
196.                    String img = "<img src = '" + imgsrc + "' width = 25 height =
    25>";
197.                    String cad = msj.getMensaje().replace(":'|", img);
198.                    msj.setMensaje(cad);
199.                }
200.
201.                p = Pattern.compile(":||");
202.                m = p.matcher(msj.getMensaje());
203.
204.                if(m.find()) {
205.                    System.out.println("Se encontro :(");
206.                    String imgsrc = Cliente.class.getClassLoader().getSystemResour
    ce("./Emojis/triste.png").toString();
207.                    String img = "<img src = '" + imgsrc + "' width = 25 height =
    25>";
208.                    String cad = msj.getMensaje().replace(":||", img);
209.                    msj.setMensaje(cad);
210.                }
211.
212.                p = Pattern.compile(":|");
213.                m = p.matcher(msj.getMensaje());
214.
215.                if(m.find()) {
216.                    System.out.println("Se encontro :)");
217.                    String imgsrc = Cliente.class.getClassLoader().getSystemResour
    ce("./Emojis/feliz.png").toString();
218.                    String img = "<img src = '" + imgsrc + "' width = 25 height =
    25>";
219.                    String cad = msj.getMensaje().replace(":|", img);
220.                    msj.setMensaje(cad);
221.                }
```

```
222.
223.                 new Thread(new EnviaMensajes(msj)).start();
224.             }
225.
226.             public void saludo(String nombre) {
227.                 String a = " <b>El usuario [" + nombre + "] se ha conectado</b>";

228.                 Mensaje m = new Mensaje(a, nombre, "", 0);
229.                 new Thread(new EnviaMensajes(m)).start();
230.             }
231.
232.             public void carpeta() {
233.                 File carpeta = new File("./" + nombre);
234.                 if(!carpeta.exists()) {
235.                     try {
236.                         if(carpeta.mkdir())
237.                             System.out.println("Se creo la carpeta");
238.                         else
239.                             System.out.println("No se creo la carpeta");
240.                     }catch(SecurityException se) {
241.                         se.printStackTrace();
242.                     }
243.                 }
244.             }
245.
246.             private class EnvioArchivos implements Runnable {
247.                 private EnvioArchivos(File file) {
248.                     this.file = file;
249.                 }
250.
251.                 public void run() {
252.                     try {
253.                         DataInputStream dis = new DataInputStream(new FileInputStr
    eam(file));
254.                         long tamanio = dis.available();
255.                         long enviado = 0;
256.                         int n = 0;
257.                         int i = 0;
258.                         while (enviado < tamanio) {
259.                             Mensaje datos = new Mensaje(file.getName(), nombre, ""
    , 2, file.length(), "", ++i);
260.                             ByteArrayOutputStream baos = new ByteArrayOutputStream
    (6400);
261.                             ObjectOutputStream oos = new ObjectOutputStream(new Bu
    fferedOutputStream(baos));
262.                             oos.flush();
263.                             byte[] b = new byte[4000];
264.                             n = dis.read(b);
265.                             byte[] b2 = new byte[n];
266.                             System.arraycopy(b, 0, b2, 0, n);
267.                             datos.setDatos(b2);
268.                             datos.setBytesEnviados(n);
269.                             oos.writeObject(datos);
270.                             oos.flush();
271.                             byte[] d = baos.toByteArray();
272.                             DatagramPacket paqueteEnvio = new DatagramPacket(d, d.
    length, grupo, puerto);
273.                             cliente.send(paqueteEnvio);
274.                             try {
275.                                 Thread.sleep(500);
276.                             }catch (Exception e) { e.printStackTrace(); }
```

```
277.                            System.out.println("Numero paquete:" + i);
278.                            enviado += n;
279.                            oos.close();
280.                            baos.close();
281.                        }
282.                        byte[] bFinal = {0x02};
283.                        Mensaje paqueteFinal = new Mensaje(file.getName(), nombre,
    "", 2, file.length(), "", 0);
284.
285.                        paqueteFinal.setDatos(bFinal);
286.                        ByteArrayOutputStream baos = new ByteArrayOutputStream();

287.                        ObjectOutputStream oos = new ObjectOutputStream(baos);
288.
289.                        oos.writeObject(paqueteFinal);
290.                        oos.flush();
291.
292.                        byte[] mnsj = baos.toByteArray();
293.
294.                        DatagramPacket dp = new DatagramPacket(mnsj,mnsj.length, g
   rupo, puerto);
295.                        cliente.send(dp);
296.
297.                        System.out.println("Archivo Enviado");
298.
299.                        oos.close();
300.                        baos.close();
301.                        //cliente.close();
302.                        dis.close();
303.                } catch(Exception e) {
304.                    e.printStackTrace();
305.                }//try/catch
306.                }
307.
308.                private File file;
309.            }
310.
311.            public void enviarArchivo(File file) {
312.                new Thread(new EnvioArchivos(file)).start();
313.            }
314.
315.            private void recibirArchivo(Mensaje datos) {
316.                try{
317.                        System.out.println("Numero de paquete: " + datos.getNp());

318.                        if(datos.getNp() == 0) {
319.                            dos = new DataOutputStream(new FileOutputStream("./" +
    nombre + "/" + nombreArchivo));
320.                            for(int i = 0; i < lista.size(); i++) {
321.                                dos.write(lista.get(i));
322.                            }
323.                            dos.close();
324.                            lista.clear();
325.                        } else if(datos.getNp() == 1) {
326.                            lista = new ArrayList<>();
327.                            nombreArchivo = datos.getNombre();
328.                            lista.add(datos.getDatos());
329.                        } else {
330.                            if(nombreArchivo.equals(datos.getNombre()))
331.                                lista.add(datos.getDatos());
332.                        }
```

```
333.                     Thread.sleep(500);
334.                 }catch(Exception e) {
335.                     e.printStackTrace();
336.                 }
337.
338.             }
339.
340.             private String nombre;
341.             private String nombreArchivo;
342.             private String host;
343.             private int puerto;
344.             private JEditorPane editor;
345.             private Thread escucha;
346.             private Runnable hiloEscucha;
347.             private MulticastSocket cliente;
348.             private InetAddress grupo;
349.             private JComboBox<String> usuarioConectado;
350.             private DataOutputStream dos;
351.             private ArrayList <byte[]> lista = null;
352.             private Pattern p;
353.             private Matcher m;
354.         }
```

Dificultades Encontradas.

El primer punto a superar fue el comprender de buena manera como funcionaban los sockets multicast, ya que en un principio se tenia la vaga idea de usar un cliente y servidor, pero eso no era fácil de usar.

Una vez pasado este punto el siguiente, fue como actualizar la lista de contactos ya que en un punto pasaba de que se agregaban a modo que llegaban, es decir el primer usuario registrado es el que tenia la lista completa de todos los que se conectaban, el truco estuvo en poner un segundo anuncio que mandaba el nombre de usuario ya existente al recibir uno nuevo.

Otro punto a resaltar, fue el hecho de sobrecargar el constructor del objeto Mensaje, con el propósito de poder mandar mensajes y archivos sin problemas entre el envío de mensajes y archivos.

Por último, el uso de hilos fue imprescindible para evitar que la interfaz se quedara a la espera del final del proceso para poder seguir funcionando, el uso de clases internas fue esencial para poder usar diversos hilos con diferentes usos.

Como posibles mejoras existen demasiadas, como el hecho de hacer anuncios de los usuarios ya conectados aparte de añadirlos a la lista de contactos, validaciones de cuando el usuario ya esta conectado no se pueda registrar otro con el

mismo nombre y quitar los usuarios de la lista de contactos cuando se desconecta.

Conclusiones.

Luis Enrique Rojas Alvarado.

En esta práctica pudimos observar las diferencias entre los sockets multicas y los de flujo. Pero la versatilidad que nos dan es amplia para hacer aplicaciones que deben tratar varios elementos interactuando entre sí, pero cuidando la integridad de los datos que se van a mandar, puesto que con UDP hay que buscar la forma de garantizar el envío de paquetes y el uso de hilos para soportar la concurrencia de múltiples usuarios al mismo tiempo.

Miranda Sandoval Mario Alberto.

Esta práctica fue interesante por el hecho de tener que usar los hilos y usar sockets multicast, el uso de clases privadas que implementaban la interfaz Runnable permitieron hacer que cada hilo actuara de manera diferente, mientras que usar distintos métodos para las llamadas dieron el éxito.

Para los emojis hubo que cambiar algunos caracteres del texto del botón, ya que el analizador de expresiones regulares de Java devolvía errores al usar paréntesis o llaves.