Instituto Politécnico Nacional

Escuela Superior de Cómputo

Práctica 7: Aplicación P2P

Estudiantes:

Morales Hernández Carlos Jesús

Ramirez Hidalgo Marco Antonio

Grupo: 3CM16

Materia: Aplicaciones para comunicaciones en red

Docente: Axel Ernesto Moreno Cervantes

Carrera: Ingeniería en Sistemas Computacionales

Miércoles, 15 de junio del 2022

Índice

1. Introducción 1
1.1. Comunicación P2P 1
1.1.1. Preocupaciones en las comunicaciones P2P 1
1.2. Tráfico Multicast 2
1.2.1. Direccionamiento IP para Multicast 2
1.3. RMI en Java 3
1.3.1. Invocación remota de objetos 3
1.4. MD5 4
1.4.1. Algoritmo del MD5 5
2. Desarrollo experimental 5
2.1. Código 5
2.1.1. Clase Archivo 5
2.1.2. Clase ServidorMulticast 6
2.1.3. Clase ServidorFlujo 7
2.1.4. Clase ServicioRMI 8
2.1.5. Clase ListaRespuesta 11
2.1.6. Clase ListaArchivos 11
2.1.7. Clase Interfaz 11
2.1.8. Clase DefalutTablerModel.java 19
2.1.9. Clase Datos 20
2.1.10. Clase ClienteRMI 20
2.1.11. Clase ClienteMulticast 21
2.1.12. Clase ClienteFlujo 23
2.1.13. Clase InterfazBusc 24
2.2. Capturas de la ejecución 24
3. Conclusiones 29
Bibliografía 30

Práctica 7: Aplicación P2P

1. Introducción

1.1. Comunicación P2P

Las tecnologías 'peer to peer' (P2P) hacen referencia a un tipo de arquitectura para la comunicación entre aplicaciones que permite a individuos comunicarse y compartir información con otros individuos sin necesidad de un servidor central que facilite la comunicación. Es importante destacar que el término "P2P" se refiere a un tipo de arquitectura de aplicaciones y no a la funcionalidad específica de una aplicación final; es decir, la tecnología P2P es un medio para alcanzar un fin superior. [1]

Sin embargo, a menudo se utiliza el término "P2P" como sinónimo de "intercambio de archivos", ya que éste es uno de los usos más populares de dicha tecnología. No obstante, existen muchos otros usos de la tecnología P2P, por ejemplo Skype utiliza una arquitectura P2P híbrida para ofrecer servicios VoIP, mientras que Tor utiliza una arquitectura P2P para ofrecer una funcionalidad de enrutamiento anónimo. [1]

La ventaja principal de la tecnología P2P es que saca el máximo partido de los recursos (ancho de banda, capacidad de almacenamiento, etc.) de los muchos clientes/peers para ofrecer servicios de aplicación y red, sin tener que confiar en los recursos de uno o más servidores centrales. De este modo se evita que tales servidores se conviertan en un cuello de botella para toda la red. [1]

Otra ventaja de la tecnología P2P es que no existe una autoridad central única que se pueda eliminar o bloquear y colapsar toda la red P2P. Esto dota a la red de la capacidad de sobrevivir por sí misma. [1]

1.1.1. Preocupaciones en las comunicaciones P2P

Las aplicaciones P2P empleadas en la red de una empresa pueden suponer una amenaza y una fuente de preocupaciones [1]:

- **Fuga de datos:** Publicación de información o archivos de la empresa de forma consciente o inconsciente.
- **Violación de derechos de propiedad intelectual:** Descarga por parte de los usuarios de contenidos ilegales/protegidos por derechos de propiedad intelectual.
- **Consumo de recursos:** Consumo excesivo de ancho de banda, incluyendo un consumo de ancho de banda adicional por el servicio prestado a otros peers en lugar de para usos directamente relacionados con la actividad del usuario.
- **Control de acceso:** La naturaleza descentralizada de las tecnologías P2P hacen que sea difícil prevenir su uso mediante el empleo de mecanismos tradicionales para el control del acceso a la red.
- **Retención de datos:** Registrar y auditar de forma correcta los datos de las comunicaciones P2P es una tarea difícil y en muchos casos imposible.
- Malware: Los usuarios pueden descargar virus, troyanos u otros tipos de malware.

- **Pérdida de tiempo:** El tiempo que se emplea utilizando las aplicaciones P2P es tiempo que no se dedica a trabajar.

1.2. Tráfico Multicast

En las redes IPv4 existen diferentes tipos de comunicaciones entre los diferentes hosts que hay en una misma red, habitualmente se utiliza tráfico Unicast (tráfico de un equipo a otro equipo) para la comunicación y la transferencia de datos, sin embargo, también existe el tráfico IP Multicast, o también conocido como multidifusión, cuyo objetivo es enviar solamente información a aquellos equipos (clientes) que están configurados específicamente para recibir este tráfico de red. [2]

El tráfico IP Multicast, o también conocido como multidifusión IP, es un método para transmitir información a un grupo de receptores (clientes) que están configurados para tal fin. Los equipos que no están configurados específicamente no recibirán este tráfico de red y podrán dedicarse a enviar y recibir otro tipo de tráfico. [2]

El tráfico Multidifusión está asociado específicamente a un grupo de «clientes» interesados en recibir ese tráfico de red, si no está en el grupo de Multicast, no recibirán la información, esto es algo ideal para no colapsar las redes, ni tener que enviar copias de todos los paquetes a todos los clientes. La comunicación se realiza una vez desde la dirección IP Unicast de origen hasta la dirección de IP Multicast elegida, independientemente de cuántos clientes haya en esta dirección de Multicast, todos los que estén en el grupo recibirán los datagramas. [2]

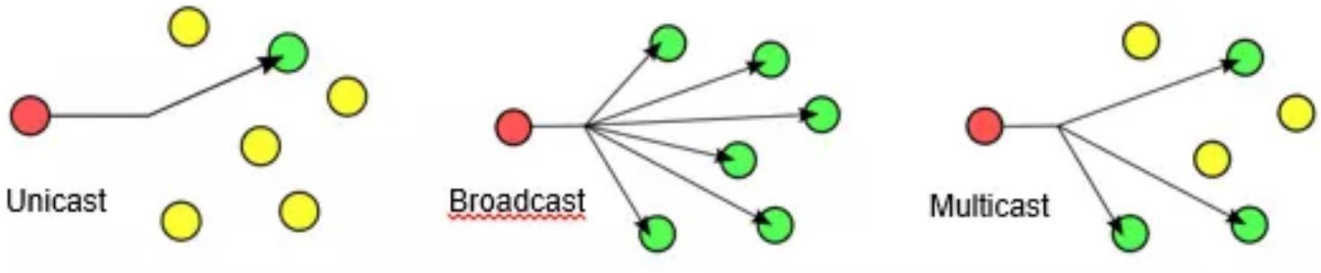

Figura 1. Comunicación unicast, broadcast y multicast

1.2.1. Direccionamiento IP para Multicast

El direccionamiento IP del tráfico multicast tiene un rango específico, este rango va desde la dirección IP 224.0.0.0 a la 239.255.255.255 están destinadas para ser direcciones de multidifusión explícitamente, a este rango se le suele llamar clase D. Estas direcciones IP no se asignan a direcciones unicast tradicionales, es un rango reservado, además, dentro del rango especificado hay otros subrangos que no se deben usar por todas las aplicaciones, estos rangos son las siguientes [2]:

224.0.0.0 – 224.0.0.255 (224.0.0/24) Local Network Control Block: este rango de direccionamiento IP es usado habitualmente por protocolos de enrutamiento de pasarela interior que hagan uso de comunicación Multicast, como RIP o OSPF. Este direccionamiento es solo para multidifusión local, por lo que no deberían ser reenviadas por los routers. [2]

1.3. RMI en Java

RMI es una tecnología desarrollada por Sun para permitir la colaboración de objetos que están localizados remotamente. Esta tecnología se enmarca en la idea de permitir colaboración entre Objetos Remotos. La idea no es que los objetos se comuniquen a través de la programación del usuario de protocolos estándares de red. La idea es tener un objeto cliente, donde podamos podamos completar un requerimiento de datos. El cliente luego prepara el requerimiento que envía a un objeto ubicado en un servidor. El objeto remoto prepara la información requerida (accediendo a bases de datos, otros objetos, etc). Finalmente el objeto remoto envía la respuesta al cliente. En lo posible esta interacción debería ser lo más semejante posible a requerimientos hechos localmente. [3]

En principio se puede anhelar la colaboración de objetos escritos en cualquier lenguaje (no es el caso de RMI). Esta idea no es simple de lograr, corresponde al esfuerzo del grupo OMG (Object Management Group, www.omg.org) los cuales propusieron CORBA (Common Object Request Broker Architecture), el cual define un mecanismo común para descubrir servicios e intercambiar datos. CORBA usa Object Request Broker (ORB) como traductores universales para la comunicación entre objetos. Los objetos remotos hablan a través de estos ORB. El protocolo de comunicación entre objetos y ORB es llamado Internet Inter-ORB Protocol o IIOP. [3]

La opción propuesta por Microsoft para comunicar objetos remotos es COM (Component Object Model). Hoy este modelo parece haber sido superado por la tecnología .NET. [3]

Cuando el cliente y servidor son escritos en Java, la generalidad y complejidad de CORBA no es requerida. En este caso Sun desarrolló RMI, un mecanismo más simple especialmente pensado para comunicación entre aplicaciones Java. [3]

1.3.1. Invocación remota de objetos

La idea suena simple, si tenemos acceso a objetos en otras máquinas, podemos llamar a métodos de ese objeto remoto. RMI maneja los detalles de enviar los parámetros, el objeto remoto debe ser activado para ejecutar el método y los valores deben ser retornados de regreso al llamador, ver la siguiente Figura. [3]

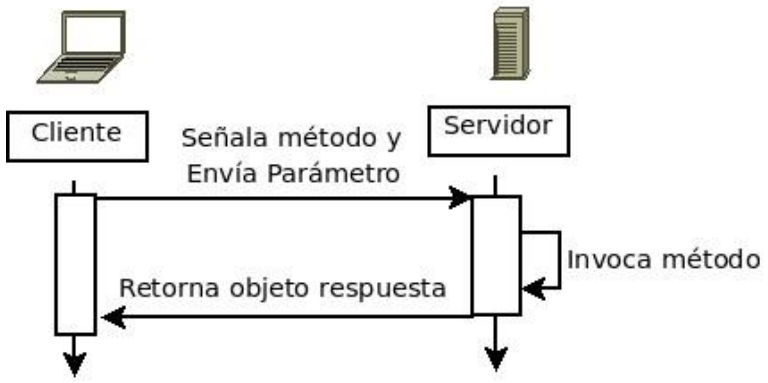

Figura 2. Invocación remota de objetos.

La terminología de RMI se aprecia en la Figura debajo. [3]

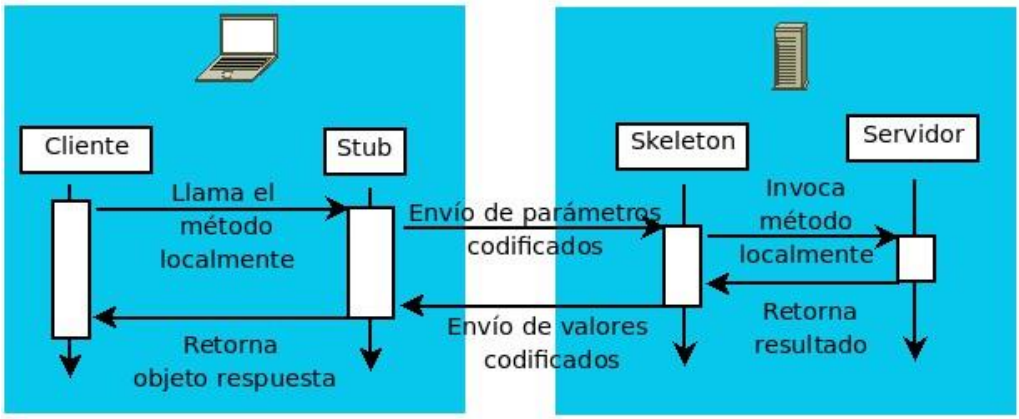

Figura 3. Stub, Skeleton y codificación de parámetros y valores retornados.

De los términos se destacan los siguientes [3]:

- **Objeto cliente:** objeto cuyo método hace el llamado remoto.
- **Objeto servidor:** Objeto remoto llamado
- **Marshalling:** es el proceso de codificación de los parámetros.
- **Stub:** es un objeto que encapsula el método que deseamos invocar remotamente. Así el llamado remoto es semejante a un llamado local. Éste prepara información con la identificación el objeto remoto a invocar, el método a invocar y codificación de los parámetros (Marshalling).
- **Skeleton:** es el objeto del lado del servidor que decodifica los parámetros, ubica el objeto llamado, llama el método deseado, codifica el valor retornado, y envía la información de regreso al stub.

Notar que los roles de cliente y servidor aplican sólo a un llamado. Un objeto servidor luego puede ser cliente al hacer un llamado remoto. [3]

Aun cuando el proceso es completo, RMI lo hace en gran medida automático y en gran media transparente para el programador. La sintaxis de llamados remotos es la misma de los llamados locales. [3]

RMI posee posee un mecanismo para cargar clases dinámicamente desde otro lugar. Esto es requerido, por ejemplo, cuando el valor retornado corresponde a una instancia de una clase derivada de la clase conocida por el cliente. Aquí se ocupa un mecanismo similar al usado por applets. [3]

1.4. MD5

En Criptografía, MD5 (acrónimo de Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje 5) es un Algoritmo de reducción criptográfico de 128 Bits ampliamente usado. El código MD5 fue diseñado por Ronald Rivest en 1991. Durante el año 2004 fueron divulgados ciertos defectos de seguridad, lo que hará que en un futuro cercano se cambie de este sistema a otro más seguro. [4]

MD5 es uno de los algoritmos de reducción criptográficos diseñados por el profesor Ronald Rivest del MIT (Massachussets Institute of Technology, Instituto Tecnológico de Masachusets). Cuando un análisis analí•tico indicó que el algoritmo MD4 era inseguro, se decidió a programar el MD5 para sustituirlo en 1991. Las debilidades en MD4 fueron descubiertas por Hans Dobbertin. [4]

En 1996 Dobbertin anunció una colisión de Hash de la función de compresión del MD5. Esto no era una ataque contra la función de hash del MD5, pero hizo que los criptógrafos empezasen a recomendar el reemplazo de la codificación MD5 a otras como SHA-1 o RIPEMD-160. En Agosto de 2004 unos investigadores chinos encontraron también colisiones hash en el MD5. Actualmente el uso de MD5 es muy amplio y se desconoce cómo afectarán estos problemas a su uso y a su futuro. [4]

1.4.1. Algoritmo del MD5

En este documento "palabra" es una entidad de 32 bits y byte es una entidad de 8 bits. Una secuencia de bits puede ser interpretada de manera natural como una secuencia de bytes, donde cada grupo consecutivo de ocho bits se interpreta como un byte con el bit más significativo al principio. Similarmente, una secuencia de bytes puede ser interpretada como una secuencia de 32 bits (palabra), donde cada grupo consecutivo de cuatro bytes se interpreta como una palabra en la que el bit menos significativo está al principio. [4]

La codificación del MD5 de 128 bits es representada tí•picamente como un número de 32 dí•gitos hexadecimal. El siguiente código de 28 bytes ASCII será tratado con MD5 y con eso se tiene la salida. [4]

2. Desarrollo experimental

Primero, se va a mostrar el código y después capturas del funcionamiento de este.

2.1. Código

El código que se desarrolló en esta práctica se dividió en doce clases y una interfaz, las cuales se agregan a continuación:

2.1.1. Clase Archivo

```
import java.io.Serializable;

public class Archivo implements Serializable {

    private String archivo;
    private String hash;
    private long tam;

    public Archivo(){
```

```
    }

    public Archivo(String archivo, String hash){
        this.archivo = archivo;
        this.hash = hash;
    }

    public long getTam() {
        return tam;
    }

    public void setTam(long tam) {
        this.tam = tam;
    }

    public String getArchivo() {
        return archivo;
    }

    public void setArchivo(String archivo) {
        this.archivo = archivo;
    }

    public String getHash() {
        return hash;
    }

    public void setHash(String hash) {
        this.hash = hash;
    }

}
```

2.1.2. Clase ServidorMulticast

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.InetAddress;
import java.net.MulticastSocket;


public class ServidorMulticast extends Thread{
    private final MulticastSocket ms;
    private final String host = "228.1.1.1";
    private final int PuertoRMI;
    private final int PuertoEnvio;
    private final String ID;
    private final String direccion;

    public ServidorMulticast(MulticastSocket ms, int PuertoRMI, int PuertoEnvio, String ID, String
direccion){
        this.ms = ms;
        this.PuertoRMI = PuertoRMI;
        this.PuertoEnvio = PuertoEnvio;
```

```
        this.ID = ID;
        this.direccion = direccion;
    }

    @Override
    public void run()
    {
        try
        {
            InetAddress gpo = InetAddress.getByName(host);
            String mensaje = "" + PuertoRMI + " " + PuertoEnvio + " " + ID + " " + direccion;
            EnviarMensaje(mensaje, gpo);

            try{
                //Lo envia cada 5 segundos de que esta ahi
                Thread.sleep(5000);
            }catch(InterruptedException ie){}
        }
        catch(IOException ex){
            ex.printStackTrace();
        }
    }

    private void EnviarMensaje(String mensaje, InetAddress gpo) throws IOException
    {
        byte[] buffer = mensaje.getBytes();
        DatagramPacket dp = new DatagramPacket(buffer, buffer.length, gpo,7777);
        ms.send(dp);
    }

}
```

2.1.3. Clase ServidorFlujo

```
import java.io.BufferedInputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.RandomAccessFile;
import java.net.ServerSocket;
import java.net.Socket;


public class ServidorFlujo extends Thread {

    private int puerto;
    private String rutaDescarga;
```

```
    public ServidorFlujo(int puerto, String rutaDescarga)
    {
        this.puerto = puerto;
        this.rutaDescarga = rutaDescarga;
    }



    @Override
    public void run()
    {
        try
        {
            ServerSocket ss = new ServerSocket(puerto);
            ss.setReuseAddress(true);

            while(true)
            {
                Socket servidor = ss.accept();
                DataInputStream dis = new DataInputStream(servidor.getInputStream());
                String nombre = dis.readUTF();
                long tam = dis.readLong();
                DataOutputStream dos =  new DataOutputStream(new FileOutputStream(rutaDescarga +
nombre));


                long recibidos = 0;
                int l = 0, porcentaje = 0;

                while (recibidos < tam)
                {
                    byte[] b = new byte[8192];
                    l = dis.read(b);
                    if(l == -1)
                        break;
                    dos.write(b, 0, l);
                    dos.flush();
                    recibidos = recibidos + l;
                    porcentaje = (int) ((recibidos * 100) / tam);
                    //System.out.print("\rEnviando el " + porcentaje + " % del archivo");
                }
                dos.close();
                dis.close();
                servidor.close();
            }
        }catch(Exception ex)
        {
            ex.printStackTrace();
        }


    }
}
```

2.1.4. Clase ServicioRMI

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FilenameFilter;
```

```
import java.io.IOException;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;


public class ServicioRMI extends Thread implements ListaArchivos{

    private int puerto;
    private String ruta;

    public ServicioRMI(){}

    public ServicioRMI(int puerto)
    {
        this.puerto = puerto;
    }

    public ServicioRMI(int puerto, String ruta)
    {
        this.puerto = puerto;
        this.ruta = ruta;
    }

    private String obtenerHash(File archivo, MessageDigest md) throws
FileNotFoundException, IOException
    {
        FileInputStream fis = new FileInputStream(archivo);
        byte[] bytes = new byte[1024];
        int bcount = 0;

        while((bcount = fis.read(bytes)) != -1)
        {
            md.update(bytes,0,bcount);
        }
        fis.close();

        byte[] bytesMes = md.digest();

        StringBuilder sb = new StringBuilder();
        for(int i = 0; i<bytesMes.length; i++)
        {
            sb.append(Integer.toString((bytesMes[i] & 0xff) + 0x100, 16).substring(1));
        }
        return sb.toString();
    }

    public  ArrayList <Archivo> busquedaArchivos(String archivo, String dir) throws
RemoteException
    {
```

```
        ArrayList<Archivo> encontrados = new ArrayList();
        File path = new File(dir);
        System.out.println("Buscando " +archivo + " en la direccion: "+dir);
        int bandera = 0;

        File[] coincide = path.listFiles(new FilenameFilter()
        {
            public boolean accept(File dir, String nombre){
                return nombre.startsWith(archivo);
            }
        });
        try
        {
            MessageDigest md = MessageDigest.getInstance("MD5");
            for(File match: coincide)
            {
                String rutaArchivo = match.getAbsolutePath();
                String hash = obtenerHash(match, md);
                long tam = match.length();

                Archivo arch = new Archivo();
                arch.setArchivo(rutaArchivo);
                arch.setHash(hash);
                arch.setTam(tam);

                encontrados.add(arch);

            }
        }catch(NoSuchAlgorithmException ex) {
            Logger.getLogger(ServicioRMI.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IOException ex) {
            Logger.getLogger(ServicioRMI.class.getName()).log(Level.SEVERE, null, ex);
        }
        return encontrados;
    }

    @Override
    public void run()
    {
        try {
            java.rmi.registry.LocateRegistry.createRegistry(this.puerto)
            ; System.out.println("RMI registry ready");
        } catch (RemoteException ex) {
            Logger.getLogger(ServicioRMI.class.getName()).log(Level.SEVERE, null, ex);
        }
        try
        {
            //Para que jale necesitas cambiarle estas rutas y poner las rutas que quieras
que tenga el servidor y encontrar ese archivo
            System.setProperty("java.rmi.server.codebase","file:G:\\permisos.policy");
        ServicioRMI obj = new ServicioRMI();
        ListaArchivos stub = (ListaArchivos) UnicastRemoteObject.exportObject((Remote)
this, 0);

        // Bind the remote object's stub in the registry
```

```
        Registry registry = LocateRegistry.getRegistry(this.puerto);
            System.out.println("Servidor rmi en el puerto: "+this.puerto);
        registry.bind("ListasArchivos", stub);

        System.err.println("Servidor listo...");
        } catch (Exception ex) {
            ex.printStackTrace();
        }


    }


}
```

2.1.5. Clase ListaRespuesta

```
import java.util.ArrayList;

public class ListaRespuesta {
    private ArrayList <Archivo> archivos;

    public ListaRespuesta(ArrayList <Archivo> archivos){
        this.archivos = archivos;
    }

    public ArrayList<Archivo> getArchivos() {
        return archivos;
    }

    public void setArchivos(ArrayList<Archivo> archivos) {
        this.archivos = archivos;
    }
}
```

2.1.6. Clase ListaArchivos

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.ArrayList;

public interface ListaArchivos extends Remote{
   ArrayList <Archivo> busquedaArchivos(String archivo, String direccion) throws RemoteException;
}
```

2.1.7. Clase Interfaz

```
import java.io.File;
import java.io.IOException;
import java.net.MulticastSocket;
import java.net.ServerSocket;
import java.util.ArrayList;
import java.util.UUID;
import java.util.logging.Level;
```

```
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import java.io.RandomAccessFile;
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
 */

/**
 *
 * @author tecni
 */
public class Interfaz extends javax.swing.JFrame {

    /**
     * Creates new form Interfaz
     */
    //Para que jale necesitas cambiarle estas rutas y poner las rutas que quieras que tenga el
servidor
    ArrayList <Datos> nodos = new ArrayList();
    ArrayList <Archivo> archivos = new ArrayList();
    ArrayList <Archivo> respuesta = new ArrayList();
    ArrayList <ListaRespuesta> resultado = new ArrayList();
    DefaultTableModel model = new DefaultTableModel();
    DefaultTableModel modelSer = new DefaultTableModel();
    String rutaDescarga = "G:\\Otros\\Practica7\\Descargas\\";

    public Interfaz() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel(); jLabel2 =
        new javax.swing.JLabel(); jScrollPane1 = new
        javax.swing.JScrollPane(); jTxtBuscar = new
        javax.swing.JTextArea(); jButton1 = new
        javax.swing.JButton(); jScrollPane2 = new
        javax.swing.JScrollPane(); jTextArchivos =
        new javax.swing.JTextPane(); jLabel3 = new
        javax.swing.JLabel(); jScrollPane4 = new
        javax.swing.JScrollPane(); jTablaServidores =
        new javax.swing.JTable(); jBtnInicio = new
        javax.swing.JButton(); jScrollPane3 = new
        javax.swing.JScrollPane(); jTableEncontrados
        = new javax.swing.JTable(); jButton2 = new
        javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Practica 7. RMI");

        jLabel1.setText("Servidores activos");

        jLabel2.setText("Archivos descargados");
```

```
        jTxtBuscar.setColumns(20);
        jTxtBuscar.setRows(5);
        jTxtBuscar.setName("Jbuscar"); // NOI18N
        jScrollPane1.setViewportView(jTxtBuscar);

        jButton1.setText("Buscar");
        jButton1.setEnabled(false);
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jScrollPane2.setViewportView(jTextArchivos);

        jLabel3.setText("Resultado de busqueda");

        jTablaServidores.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null}
            },
            new String [] {
                "ID", "RMI", "Servidor Flujo", "Temporizador"
            }
        ));
        jScrollPane4.setViewportView(jTablaServidores);

        jBtnInicio.setText("Inicio");
        jBtnInicio.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jBtnInicioActionPerformed(evt);
            }
        });

        jTableEncontrados.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null}
            },
            new String [] {
                "Title 1", "Title 2", "Title 3", "Title 4"
            }
        ));
        jScrollPane3.setViewportView(jTableEncontrados);

        jButton2.setText("Descargar");
        jButton2.setEnabled(false);
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                            .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED_SIZE, 270,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel3))
                        .addGap(33, 33, 33)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 107,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jScrollPane4, javax.swing.GroupLayout.PREFERRED_SIZE, 266,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                            .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 154,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel2)))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 386,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(18, 18, 18)
                        .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 109,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jBtnInicio)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jButton2)))
                .addContainerGap(17, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addContainerGap(21, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(jLabel1)
                    .addComponent(jLabel3))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                    .addComponent(jScrollPane4, javax.swing.GroupLayout.DEFAULT_SIZE, 289,
Short.MAX_VALUE)
                    .addComponent(jScrollPane2)
                    .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED_SIZE, 0,
Short.MAX_VALUE))

                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 34,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jBtnInicio, javax.swing.GroupLayout.PREFERRED_SIZE, 34,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 0,
Short.MAX_VALUE)
                    .addComponent(jButton2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
        );
```

```
        pack();
    }// </editor-fold>//GEN-END:initComponents
    //Boton Buscar
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jButton1ActionPerformed
        // TODO add your handling code here: String
        archivo = jTxtBuscar.getText();
        System.out.println("Buscando archivo: "+archivo);
        ArrayList <String> diccionario = new ArrayList();
        if(archivos.size()>0)
        {
            archivos.clear();
        }


        resultado.clear();
        respuesta.clear();

        while(model.getRowCount() > 0)
        {
            model.removeRow(0);
        }
        jTableEncontrados.setModel(model);


        int bandera =0;

        for(Datos datos: nodos)
        {
            //lienteRMI(String archivo, int puertoRMI, ArrayList <Archivo> Lista)
            System.out.println("Buscando en el puerto: "+ datos.getPuertoRMI());
            new ClienteRMI(archivo, Integer.parseInt(datos.getPuertoRMI()), respuesta,
datos.getDirrecion()).start();
            resultado.add(new ListaRespuesta(respuesta));
            try {
                Thread.sleep(1000);
            } catch (InterruptedException ex) {
                Logger.getLogger(Interfaz.class.getName()).log(Level.SEVERE, null, ex);
            }

        }

        try {
            Thread.sleep(1000);

            for(ListaRespuesta res: resultado)
            {
                archivos.addAll(res.getArchivos());
            }

            if(archivos.size() > 0)
            {
                for(int i = 0; i<archivos.size(); i++)
                {
                    //Vemos si se repiten archivos para evitar que muestre muchos archivos
                    String nombre = archivos.get(i).getArchivo();
                    if(!diccionario.contains(nombre))
                    {
                        diccionario.add(nombre);
                        long tam = archivos.get(i).getTam();
                        String md5 = archivos.get(i).getHash();
                        model.addRow(new Object[]{md5,nombre,tam});
```

```
                    }
                }
                JOptionPane.showMessageDialog(rootPane, "Se encontraron "+diccionario.size()+"
archivos con ese nombre");
                jTableEncontrados.setModel(model);
                jButton2.setEnabled(true);
            }
            else
            {
                JOptionPane.showMessageDialog(rootPane, "Se encontraron "+diccionario.size()+"
archivos con ese nombre");
                jButton2.setEnabled(false);
            }


        } catch (InterruptedException ex) {
            Logger.getLogger(Interfaz.class.getName()).log(Level.SEVERE, null, ex);
        }
    }//GEN-LAST:event_jButton1ActionPerformed

    public String generarId(){
        UUID uuid = UUID.randomUUID();
        return uuid.toString();
    }

    public void crearServers() throws IOException, InterruptedException
    {
        //Para que jale necesitas cambiarle estas rutas y poner las rutas que quieras que tenga el
servidor
        String[] direcciones = new String[2];
        direcciones[0] = "G:\\Otros\\Practica7\\Server2\\";
        direcciones[1] = "G:\\Otros\\Practica7\\Server3\\";

        for(int i=0; i<2; i++)
        {
            int puertoRMI = puertos(1999,4000);
            int puertoEnvio = puertos(7778,10000);
            String ID = generarId();
            MulticastSocket ms = new MulticastSocket(7777);
            ms.setReuseAddress(true);
            ms.setTimeToLive(225);

            String ruta = direcciones[i];

            System.out.println("Ruta de la carpeta: "+ ruta);
            new ServidorMulticast(ms,puertoRMI,puertoEnvio, ID, direcciones[i]).start();
            new ClienteMulticast(ms, puertoRMI, puertoEnvio, ID, nodos,direcciones[i]).start();

            modelSer.addRow(new Object[]{ID, puertoRMI, puertoEnvio, "6" });
            new ServicioRMI(puertoRMI).start();
            Thread.sleep(1000);

            new ServidorFlujo(puertoEnvio, rutaDescarga).start();
        }
        jTablaServidores.setModel(modelSer);

    }

    //Boton Inicio

    private void jBtnInicioActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jBtnInicioActionPerformed
```

```
        int puertoRMI = puertos(1999,4000);
        int puertoEnvio = puertos(7778,10000);
        String ID = generarId();
        try
        {
            MulticastSocket ms = new MulticastSocket(7777);
            ms.setReuseAddress(true);
            ms.setTimeToLive(225);

            //Para que jale necesitas cambiarle estas rutas y poner las rutas que quieras que tenga
el servidor
            String ruta = "G:\\Otros\\Practica7\\Server1\\";

            System.out.println("Ruta de la carpeta: "+ ruta);
            //ServidorMulticast(MulticastSocket ms, int PuertoRMI, int PuertoEnvio, String ID,
String direccion)
            //ClienteMulticast(MulticastSocket ms, int PuertoRMI, int PuertoFlujo, String ID,
ArrayList <Datos> servidores, String direccion)
            new ServidorMulticast(ms,puertoRMI,puertoEnvio, ID,ruta).start();
            new ClienteMulticast(ms, puertoRMI, puertoEnvio, ID, nodos,ruta).start();

            //Agregamos las columnas y el primer servidor
            modelSer.addColumn("Id");
            modelSer.addColumn("RMI");
            modelSer.addColumn("Envio");
            modelSer.addColumn("Temp");
            modelSer.addRow(new Object[]{ID, puertoRMI, puertoEnvio, "5" });
            //jTablaServidores.setModel(model);


            System.out.println("Server RMI iniciado");
            //ServicioRMI(int puerto, String ruta)
            new ServicioRMI(puertoRMI).start();
            Thread.sleep(1000);

            new ServidorFlujo(puertoEnvio,rutaDescarga).start();
            model.addColumn("MD5");
            model.addColumn("Nombre");
            model.addColumn("Tamaño");
            crearServers();

            jButton1.setEnabled(true);
            jBtnInicio.setEnabled(false);

        } catch (IOException ex) {
            Logger.getLogger(Interfaz.class.getName()).log(Level.SEVERE, null, ex);
        } catch (InterruptedException ex) {
            Logger.getLogger(Interfaz.class.getName()).log(Level.SEVERE, null, ex);
        }
    }//GEN-LAST:event_jBtnInicioActionPerformed

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jButton2ActionPerformed
        // TODO add your handling code here:
        int row = jTableEncontrados.getSelectedRow();
        String hash = jTableEncontrados.getValueAt(row, 0).toString();
        String nombre = jTableEncontrados.getValueAt(row, 1).toString();
        String tam = jTableEncontrados.getValueAt(row,2).toString();
        long tama = Long.parseLong(tam);
        int resultados = jTableEncontrados.getRowCount();

        ArrayList <Integer> indi = new ArrayList();
```

```
        ArrayList <String> direcciones = new ArrayList();
        for(int i=0; i<resultado.size(); i++)
        {
            ListaRespuesta lista = resultado.get(i);
            for(int j = 0; j<lista.getArchivos().size(); j++)
            {
                Archivo elem = lista.getArchivos().get(j);
                if(hash.equals(elem.getHash()))
                {
                    indi.add(i);
                    direcciones.add(elem.getArchivo());
                    break;
                }
            }
        }

        JOptionPane.showMessageDialog(rootPane, "La descarga se dividira en "+resultados+"
servidores");
        String fileName = archivos.get(row).getArchivo();
        String [] fileParts = fileName.split("\\\\");
        fileName = fileParts[fileParts.length - 1];

        try
        {

            int puerto = Integer.parseInt(nodos.get(indi.get(0)).getPuertoEnvio());
            String path = nodos.get(indi.get(0)).getDirrecion();

            //ClienteFlujo(String path, int puerto, int tam)
            //new ServidorFlujo(puerto, rutaDescarga).start();
            new ClienteFlujo(nombre,fileName,puerto,tama).start();
            Thread.sleep(1000);


            Thread.sleep(1000);
            archivos.clear();
            JOptionPane.showMessageDialog(rootPane, "Descarga completada");
            String texto = jTextArchivos.getText();
            texto = texto + "\n" + nombre;
            jTextArchivos.setText(texto);

        }catch(Exception ex){
            ex.printStackTrace();
        }

    }//GEN-LAST:event_jButton2ActionPerformed

    public int puertos(int lim_menor, int lim_sup)
    {
        for(int puerto = lim_menor; puerto < lim_sup; puerto++)
        {
            try{
                ServerSocket ss = new ServerSocket(puerto);
                ss.close();
                return puerto;
            }catch(Exception ex){}

        }
        return 0;
    }


    /**
```

```
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Interfaz().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jBtnInicio;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JScrollPane jScrollPane4;
    private javax.swing.JTable jTablaServidores;
    private javax.swing.JTable jTableEncontrados;
    private javax.swing.JTextPane jTextArchivos;
    private javax.swing.JTextArea jTxtBuscar;
}
```

2.1.8. Clase DefalutTablerModel.java

```
public class DefalutTableModel {
}
```

2.1.9. Clase Datos

```
public class Datos {
    public String PuertoRMI;
    public String PuertoEnvio;
    public String ID;
    public String Dirrecion;

    public Datos(String PuertoRMI, String PuertoEnvio, String ID, String Dirrecion)
    {
        this.PuertoEnvio = PuertoEnvio;
        this.PuertoRMI = PuertoRMI;
        this.ID = ID;
        this.Dirrecion = Dirrecion;
    }

    public String getPuertoRMI()
    {
        return PuertoRMI;
    }

    public void setPuertoRMI(String PuertoRMI){
        this.PuertoRMI = PuertoRMI;
    }

    public void setPuertoEnvio(String PuertoEnvio){
        this.PuertoEnvio = PuertoEnvio;
    }

    public String getID()
    {
        return ID;
    }

    public void setID(String ID){
        this.ID = ID;
    }

    public String getPuertoEnvio()
    {
        return PuertoEnvio;
    }

     public String getDirrecion() {
        return Dirrecion;
    }

    public void setDirrecion(String Dirrecion) {
        this.Dirrecion = Dirrecion;
    }
}
```

2.1.10. Clase ClienteRMI

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.ArrayList;
```

```
public class ClienteRMI extends Thread {
    private String archivo;
    private int puertoRMI;
    private ArrayList <Archivo> Lista;
    private String direccion;


    public ClienteRMI(String archivo, int puertoRMI, ArrayList <Archivo> Lista)
    {
        this.archivo = archivo;
        this.puertoRMI = puertoRMI;
        this.Lista = Lista;
    }

    public ClienteRMI(String archivo, int puertoRMI, ArrayList <Archivo> Lista, String direccion)
    {
        this.archivo = archivo;
        this.puertoRMI = puertoRMI;
        this.Lista = Lista;
        this.direccion = direccion;
    }


    public void run()
    {
        String host = "127.0.0.1";
        try
        {
            Registry registry = LocateRegistry.getRegistry(host, puertoRMI);
            System.out.println("Puerto cliente RMI: "+puertoRMI);
            ListaArchivos interfaz = (ListaArchivos) registry.lookup("ListasArchivos");
            //ListaArchivos listaArch = (ListaArchivos) registry.lookup("ListaArchivos");

            ArrayList <Archivo> archbusca = interfaz.busquedaArchivos(archivo,direccion);

            Lista.addAll(archbusca);

        }catch(Exception ex){
            ex.printStackTrace();
        }
    }
}
```

2.1.11. Clase ClienteMulticast

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.InetAddress;
import java.net.MulticastSocket;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ClienteMulticast extends Thread{
    private final MulticastSocket ms;
    private final String host = "228.1.1.1";
    private final int PuertoRMI;
    private final int PuertoFlujo;
    private final String ID;
    private ArrayList <Datos> servidores;
```

```
    private final String direccion;



    public ClienteMulticast(MulticastSocket ms, int PuertoRMI, int PuertoFlujo, String ID, ArrayList
<Datos> servidores, String direccion)
    {
        this.ms = ms;
        this.PuertoRMI = PuertoRMI;
        this.PuertoFlujo = PuertoFlujo;
        this.ID = ID;
        this.servidores = servidores;
        this.direccion = direccion;
    }

    @Override
    public void run()
    {
        try
        {
            InetAddress gp = InetAddress.getByName(host);
            ms.joinGroup(gp);
            System.out.println("Cliente unido a la direccion de grupo 228.1.1.1");

            //Creamos un datagrama de lectura
            DatagramPacket dpRecive = new DatagramPacket(new byte[6535], 6535);

            while(true)
            {
                //Recibimos los datos provenietes del servidor multicast
                ms.receive(dpRecive);
                String mensaje = new String(dpRecive.getData(), 0 ,dpRecive.getLength());
                String datos[] = mensaje.split(" ");
                String PuertoRMI = datos[0];
                String PuertoEnvio = datos[1];
                String ID = datos[2];
                String direccion = datos[3];

                //Datos(String PuertoRMI, String PuertoEnvio, String ID, String Dirrecion)
                Datos dat = new Datos(PuertoRMI, PuertoEnvio, ID, direccion);
                dat.setID(ID);
                dat.setPuertoEnvio(PuertoEnvio);
                dat.setPuertoRMI(PuertoRMI);
                dat.setDirrecion(direccion);
                //Creamos una lista de servidores en linea, vemos si tiene distinto ID para no meter
el mismo
                int encontrado = 0;
                for(int i=0; i<servidores.size(); i++)
                {
                    if(servidores.get(i).getID().equals(dat.getID())){
                        encontrado++;
                    }

                }

                if(encontrado == 0 && !dat.getID().equals(this.ID))
                {
                    servidores.add(dat);
                    //ServidorMulticast(MulticastSocket ms, int PuertoRMI, int PuertoEnvio, String
ID, String direccion)
                    new ServidorMulticast(ms, this.PuertoRMI, this.PuertoRMI, this.ID,
this.direccion).start();
```

```
                }


                //System.out.println("Servidores disponibles: "+servidores.size());

            }

        } catch (IOException ex) {
            Logger.getLogger(ClienteMulticast.class.getName()).log(Level.SEVERE, null, ex);
        }


    }



}
```

2.1.12. Clase ClienteFlujo

```
import java.io.BufferedOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.FileInputStream;
import java.io.RandomAccessFile;
import java.net.Socket;


public class ClienteFlujo extends Thread {
   private int puerto;
   private long tam;
   private String path;
   private String nombre;
   String host = "localhost";

   public ClienteFlujo(String path, String nombre, int puerto, long tam)
   {
       this.path = path;
       this.puerto = puerto;
       this.tam = tam;
       this.nombre = nombre;
   }

    @Override
    public void run()
    {
        try
        {
            Socket cliente = new Socket(host, puerto);
            DataOutputStream dos = new DataOutputStream(cliente.getOutputStream());
            System.out.println(path);
            DataInputStream dis = new DataInputStream(new FileInputStream(path));

            dos.writeUTF(this.nombre);
            dos.flush();
            dos.writeLong(this.tam);
            dos.flush();

            long enviados = 0;
            int l = 0, porcentaje = 0;
            byte[] b = null;
```

```
            while(enviados < tam || (l=dis.read(b))!= -1)
            {
                b = new byte[8192];
                l = dis.read(b);
                if(l == -1)
                    break;
                dos.write(b,0,l);
                dos.flush();
                enviados = enviados + 1;
                porcentaje = (int) ((enviados * 100) / tam);
                //System.out.println("\rDescargando el " + porcentaje + " % del archivo");
            }


            dis.close();
            dos.close();

        }catch(Exception ex){
            ex.printStackTrace();
        }
    }
}
```

2.1.13. Clase InterfazBusc

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.ArrayList;

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java to edit this template
 */

/**
 *
 * @author tecni
 */
public interface InterfazBusc extends Remote{
    ArrayList <Archivo> busqueda(String archivo) throws RemoteException;
}
```

2.2. Capturas de la ejecución

La interfaz de la práctica luce de la siguiente forma.

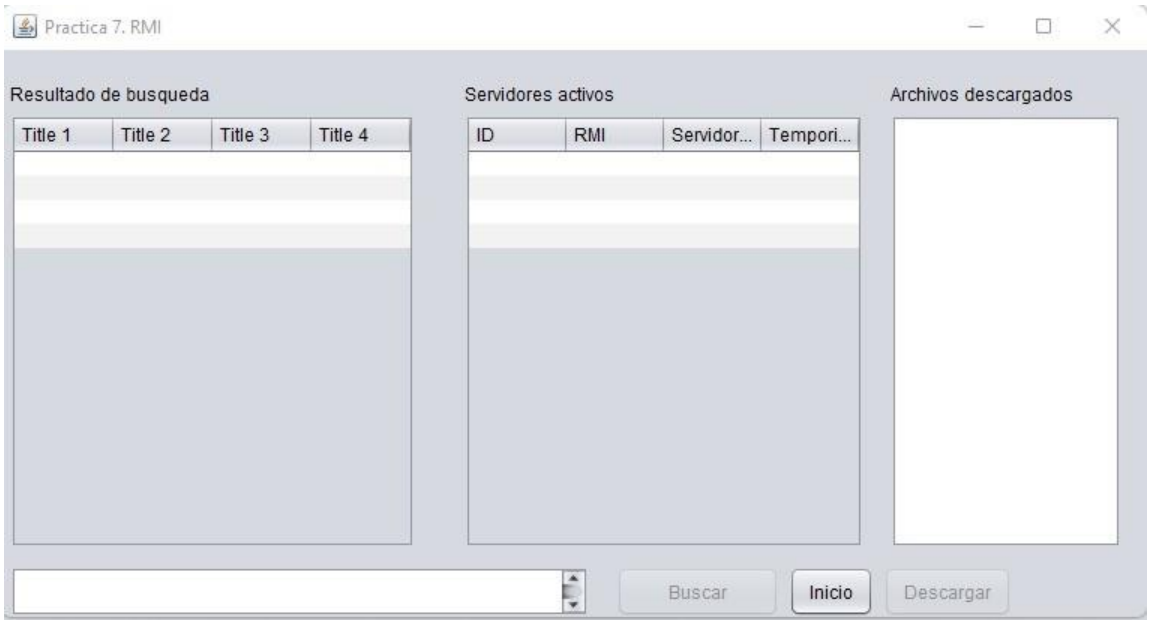

Figura 4. Interfaz de la práctica.

Cuando se da clic em el botón de inicio, se muetran los servidores activos.

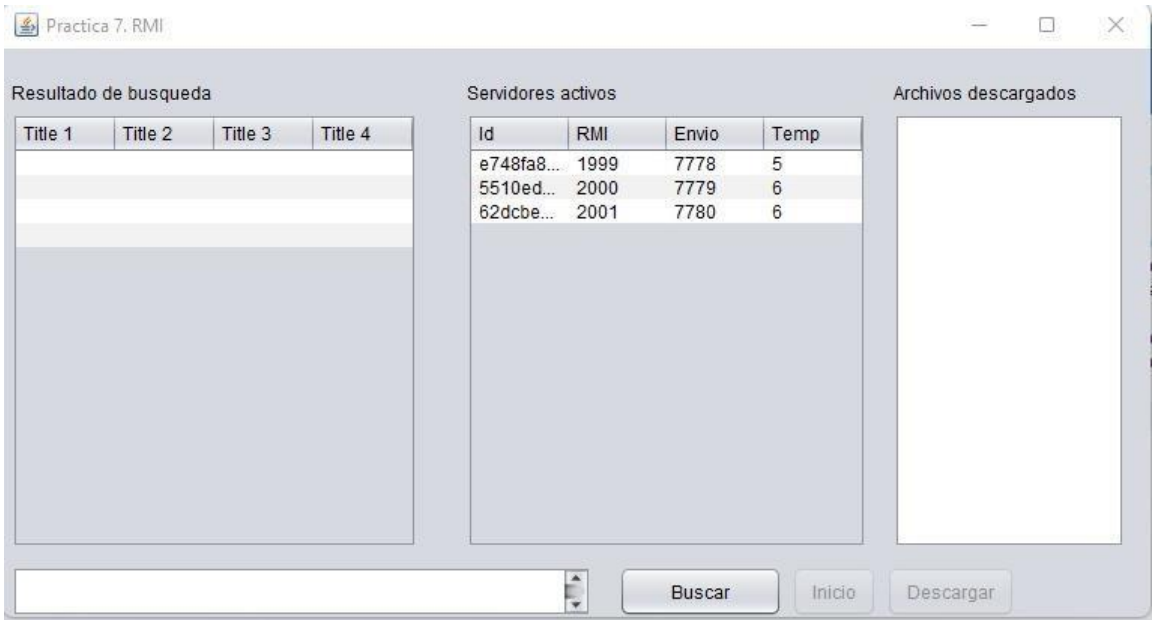

Figura 5. Servidores disponibles.

Si se da clic en el botón de buscar, aparecen los resultados obtenidos junto con su MD5 en base a el nombre del archivo ingresado en la caja de texto, asimismo, se muestran las carpetas en los servers en donde se encuentra ese archivo.

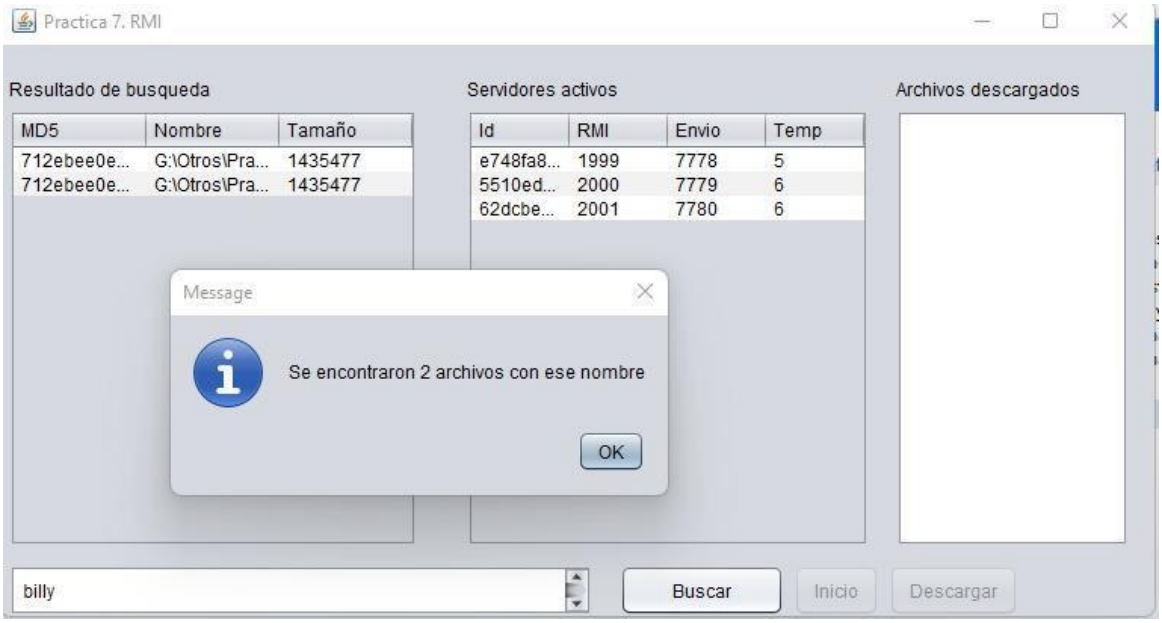

Figura 6. Búsqueda de un archivo.

Enseguida, se selecciona un archivo y se presiona el botón para descargarlo aunado con un mensaje informando en cuántos servidores se dividirá la descarga.

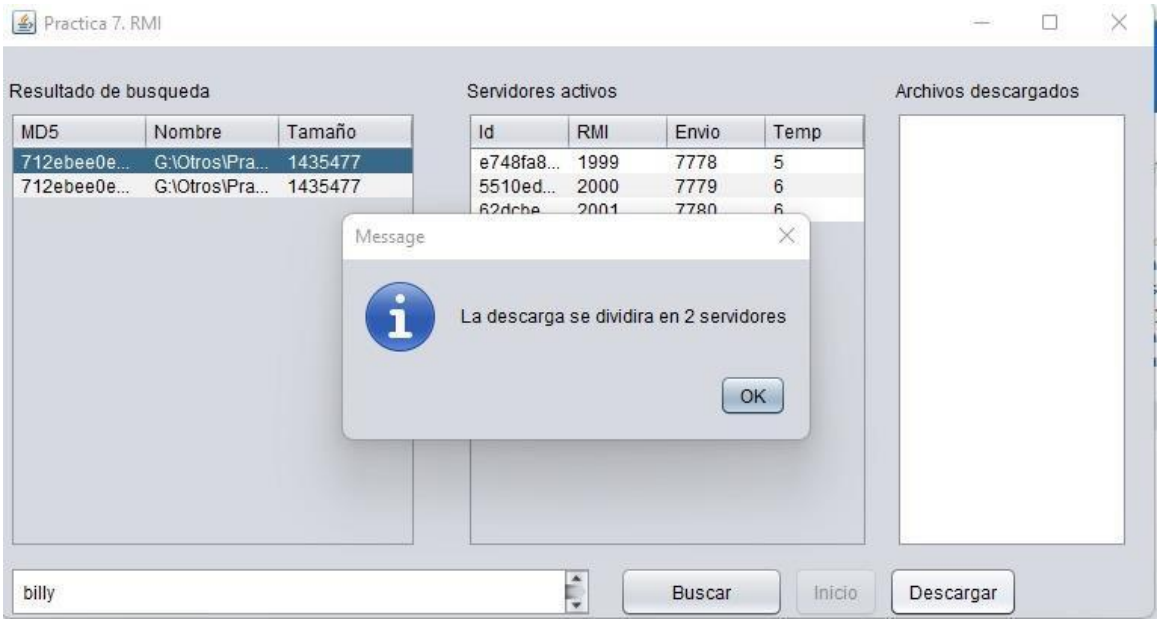

Figura 7. Descarga entre los servidores disponibles.

Cuando la descarga se completa, se muestra un mensaje satisfactorio.

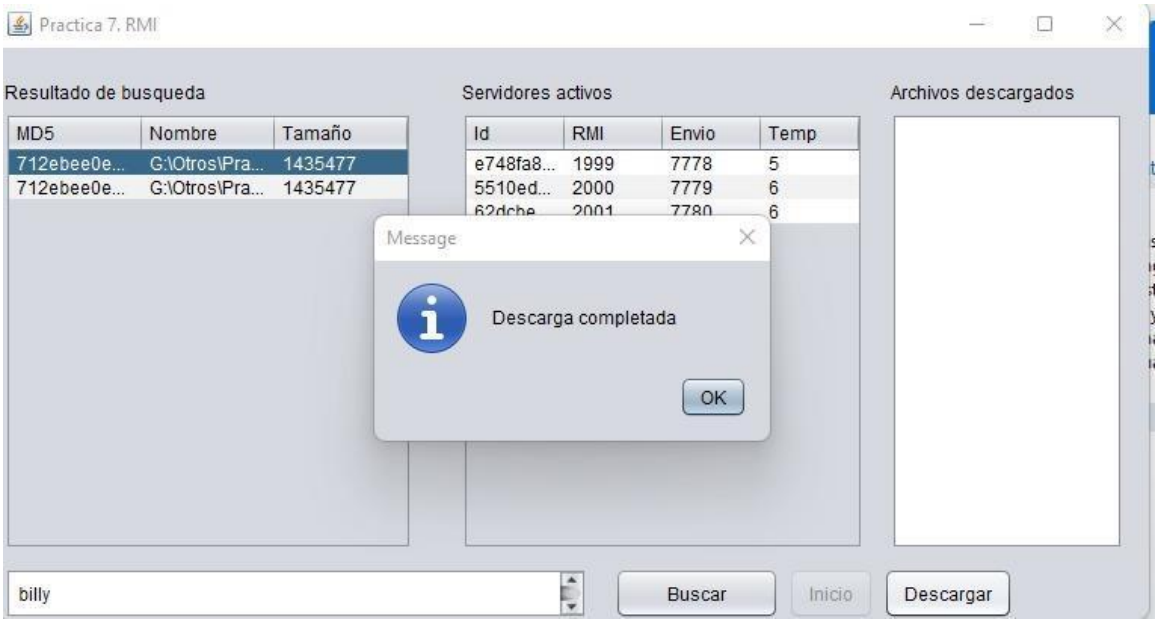

Figura 8. Mensaje que informa sobre la descarga se hizo correctamente.

El archivo descargado se muestra en el panel de la derecha.

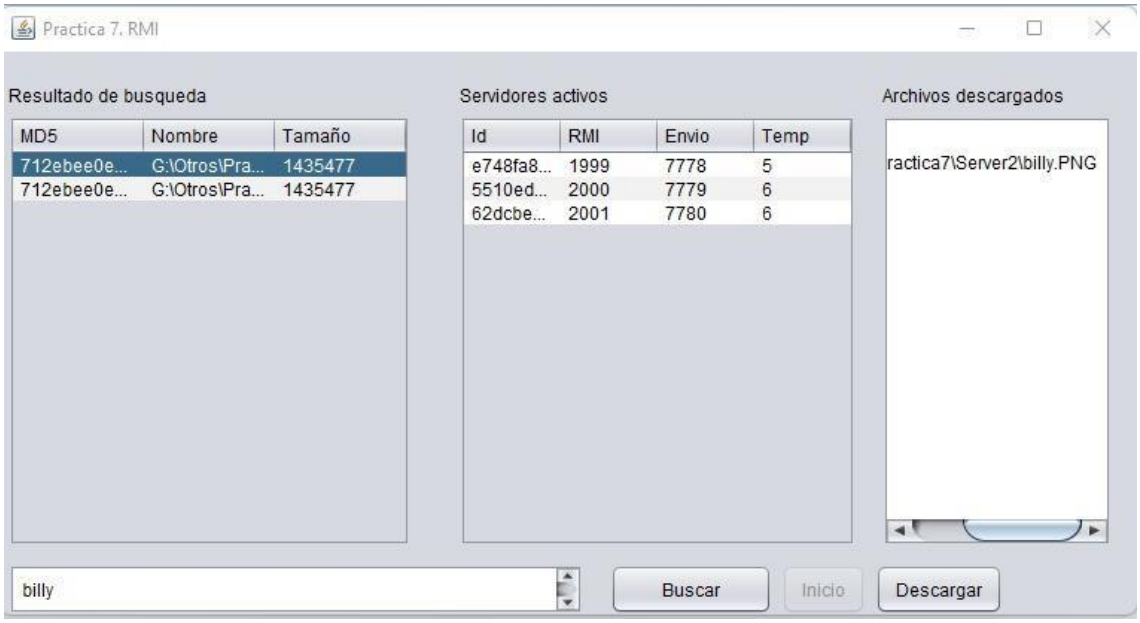

Figura 9. Descarga visible en el panel a la derecha.

En cuanto a los folders, se tiene uno para cada uno de los servidores y otro para las descargas que se realicen.

Figura 10. Folders con los datos de los servidores y las descargas.

Dos de los servidores tienen el archivo que se descargó en este ejemplo.

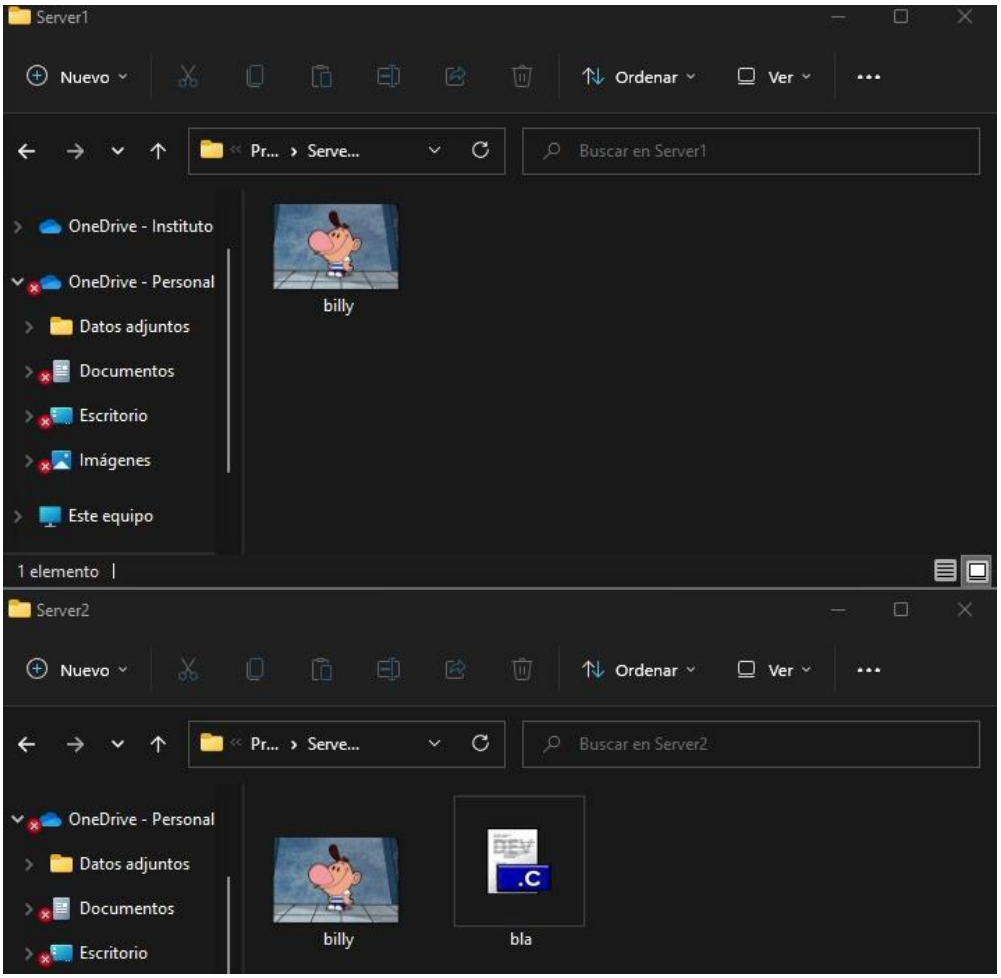

Figura 11. Archivo descargado disponible en dos servidores.

Finalmente, se muestra que el archivo descargado se encuentra en la carpeta para las descargas.

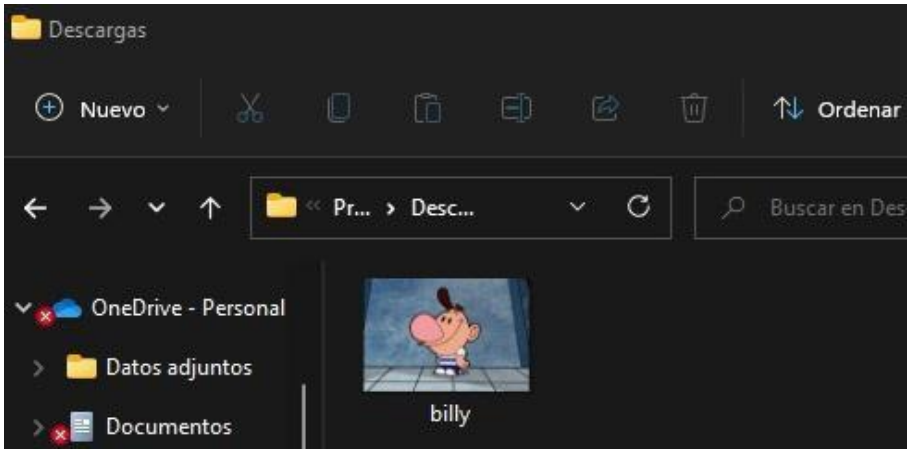

Figura 12. Archivo descargado en la carpeta de descargas.

3. Conclusiones

En esta práctica se desarrolló una aplicación P2P que permite la búsqueda y descarga de archivos. Los archivos que se buscan se encuentran dispersos entre varios servidores mientras que la descarga de los archivos se hace en una misma carpeta.

Para tener la capacidad de hacer una descarga se utilizo la transmisión de archivos Multicast y servidores de tipo RMI para crear la aplicación P2P. Con el propósito de verificar si las descargas se hicieron correctamente, se utiliza el algoritmo de Hash MD5.

El desarrollo de esta práctica conllevo problemas con la búsqueda de los archivos porque se debía configurar la ruta correctamente para encontrar todos los archivos. A la hora de la descarga, se tuvieron problemas con la unificación de la descarga cuando se hacía con varios servidores.

En general, este fue el primer acercamiento con una aplicación de tipo P2P que se tuvo en toda la carrera, con ello se aprendió que no se necesita un servidor central, sino que todos los clientes que se conectan también fungen el rol de servidor y permiten enviar los archivos de los cuales pueden tener acceso.

Bibliografía

[1] PANDA Security. *¿Qué es peer-to-peer (P2P)?*. [En línea]. Disponible en: http://resources.pandasecurity.com/enterprise/solutions/8.%20WP%20PCIP%20que%20es%20p2p.pdf

[2] S. D. Luz. (2021, septiembre 21). *Tráfico Multicast. Qué es y por qué es tan importante.* [En línea]. Disponible en: https://www.redeszone.net/tutoriales/internet/que-es-trafico-multicast/

[3] A. González. *RMI: Remote Method Invocation (Invocación Remota de Métodos)*. [En línea]. Disponible en: http://profesores.elo.utfsm.cl/~agv/elo330/2s09/lectures/RMI/RMI.html

[4] EcuRed. (2012, noviembre 13). *MD5*. [En línea]. Disponible en: https://www.ecured.cu/MD5