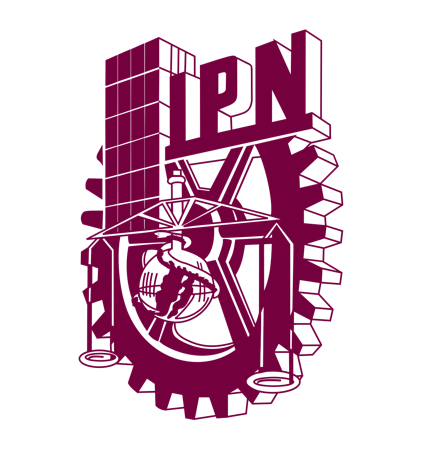
****

**INSTITUTO POLITECNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO.**

DÍAZ MEDINA JESÚS KAIMORTS

ESQUIVEL VALDEZ ALBERTO

VARGAR ROMERO ERICK EFRÁIN

**Aplicaciones para comunicaciones de red**

Axel Ernesto Moreno Cervantes

**Práctica 7. RMI.**

**Introducción.**

Si deseamos desarrollar aplicaciones sin tenernos que meter en problemas con sockets de datagrama, de flujo, etc. , existe una herramienta que nos ayuda a ello, RMI.

**RMI. (Remote Method Invocation)**

Es un mecanismo ofrecido por [Java](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java) para invocar un [método](https://es.wikipedia.org/wiki/M%C3%A9todo_(inform%C3%A1tica)) de manera remota. Proporciona un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas en Java.

RMI se caracteriza por la facilidad de su uso en la programación por estar específicamente diseñado para Java; proporciona paso de objetos por referencia, [recolección de basura](https://es.wikipedia.org/wiki/Recolecci%C3%B3n_de_basura) distribuida y paso de tipos arbitrarios.

A través de RMI, un programa Java puede exportar un [objeto](https://es.wikipedia.org/wiki/Objeto_(programaci%C3%B3n)), con lo que dicho objeto estará accesible a través de la red y el programa permanece a la espera de peticiones en un [puerto TCP](https://es.wikipedia.org/wiki/Transmission_Control_Protocol). A partir de ese momento, un cliente puede conectarse e invocar los métodos proporcionados por el objeto.

La invocación se compone de los siguientes pasos:

* Encapsulado.
* Invocación del método.
* Serialización del valor de retorno.
* El código cliente recibe la respuesta y continúa como si la invocación hubiera sido local.

**Desarrollo.**

Para esta práctica se realizará una aplicación que nos permita hacer la descarga de archivos encontrados en diferentes computadoras. Cada una de ellas funcionara como cliente y servidor.

Se realizarán tres acciones en tres hilos, el primero consiste en que, mediante sockets multicast, se conectarán varios servidores a un mismo grupo, esto servirá para que, al buscar un archivo a descargar, este archivo se busque en todos servidores que se encuentren conectados al grupo, que es en donde entra el segundo hilo que con RMI se mandará la búsqueda correspondiente a cada uno de los servidores conectados que nos va a regresar una lista con todas las coincidencias encontradas, y finalmente el tercer hilo que se encargará de descargar los o el archivo seleccionado.

**Código.**

package Frame\_Cliente;

import java.io.File;

import java.util.ArrayList;

public class Puertos {

public int puerto;

public ArrayList<File> archivo = new ArrayList();

public Puertos(int pto){

this.puerto = pto;

}

public int getPuerto() {

return puerto;

}

public ArrayList<File> getArchivo() {

return archivo;

}

public void setArchivo(ArrayList<File> archivo) {

this.archivo = archivo;

}

}

package Frame\_Servidor;

public class iniciarServidores {

public static void main(String[] args) {

Frame F1 = new Frame(5000);

F1.setVisible(true);

Frame F2 = new Frame(6000);

F2.setVisible(true);

Frame F3 = new Frame(7000);

F3.setVisible(true);

Frame F4 = new Frame(8000);

F4.setVisible(true);

Frame F5 = new Frame(9000);

F5.setVisible(true);

}

}

package md5;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.InputStream;

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

public class MD5\_Check{

public String getCodigoHash(String path) throws NoSuchAlgorithmException, FileNotFoundException, IOException{

MessageDigest digest = MessageDigest.getInstance("MD5");

File f = new File(path);

InputStream is = new FileInputStream(f);

byte[] buffer = new byte[(int) f.length()];

int read = 0;

while ((read = is.read(buffer)) > 0){

digest.update(buffer, 0, read);

}

byte[] md5sum = digest.digest();

BigInteger bigInt = new BigInteger(1, md5sum);

String output = bigInt.toString(16);

is.close();

return output;

}

/\*public static boolean Comparar(String file, String hashCode) throws NoSuchAlgorithmException, FileNotFoundException, IOException{

return hashCode.equals(getCodigoHash(file));

}\*/

public boolean comparar(String file1, String file2) throws NoSuchAlgorithmException, FileNotFoundException, IOException {

//System.out.println(getCodigoHash(file1));

//System.out.println(getCodigoHash(file2));

return getCodigoHash(file1).equals(getCodigoHash(file2));

}

public boolean compararWithMD5(String file1, String md5) throws NoSuchAlgorithmException, FileNotFoundException, IOException {

//System.out.println(getCodigoHash(file1));

//System.out.println(getCodigoHash(file2));

return getCodigoHash(file1).equals(md5);

}

/\*public void main(String[] args) throws NoSuchAlgorithmException, IOException {

System.out.println(comparar("prueba.txt","prueba2.txt"));

}\*/

}

package cliente\_paquete;

import Frame\_Cliente.Frame;

import Frame\_Cliente.Puertos;

import java.io.BufferedInputStream;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.Socket;

import java.util.ArrayList;

import javax.swing.DefaultListModel;

import javax.swing.JCheckBox;

import javax.swing.JOptionPane;

public class Cliente{

public Socket cl;

public ArrayList<File> archivos;

public ArrayList<Puertos> Servidores = new ArrayList();

ArrayList<File> archivosSelected = new ArrayList();

public int archivoRecibido = 0;

public String archBusq;

ArrayList<File> archivosRee = new ArrayList();

public Cliente(ArrayList<Puertos> Servidores){

this.Servidores.add(new Puertos(5000));

this.Servidores.add(new Puertos(6000));

this.Servidores.add(new Puertos(7000));

this.Servidores.add(new Puertos(8000));

this.Servidores.add(new Puertos(9000));

}

public boolean knowIfExist(ArrayList<File> archServidores,String archivoBuscar){

for(int i=0;i<archServidores.size();i++){

if(archServidores.get(i).getAbsolutePath().toLowerCase().contains(archivoBuscar)){

this.archivosSelected.add(archServidores.get(i));

//System.out.println(archServidores.get(i).getAbsolutePath().toLowerCase());

return true;

}

}

return false;

}

public ArrayList<File> refrescar(String host , String archivoBuscar, JCheckBox ch1,JCheckBox ch2,JCheckBox ch3,JCheckBox ch4,JCheckBox ch5) throws IOException, ClassNotFoundException{

this.archivosSelected.clear();

this.archBusq = archivoBuscar;

this.archivoRecibido = 0;

this.archivosRee.clear();

for(int i=0;i<this.Servidores.size();i++){

this.Servidores.get(i).archivo.clear();

}

for(int i=0;i<Servidores.size();i++){

this.cl = new Socket(host,Servidores.get(i).getPuerto());

DataInputStream dis = new DataInputStream(this.cl.getInputStream());

int totArchivos = dis.readInt();

System.out.println("total de archivos: "+totArchivos);

if(totArchivos != 0){

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

System.out.println("contiene archivos");

System.out.println("entroCliente");

this.cl = new Socket(host,Servidores.get(i).getPuerto());

DataOutputStream dos = new DataOutputStream(this.cl.getOutputStream());

dos.writeUTF(this.archBusq);

dos.flush();

this.cl = new Socket(host,Servidores.get(i).getPuerto());

dos.writeInt(this.archivoRecibido);

dos.close();

this.cl = new Socket(host,Servidores.get(i).getPuerto());

dis = new DataInputStream(this.cl.getInputStream());

this.archBusq = dis.readUTF();

this.cl = new Socket(host,Servidores.get(i).getPuerto());

dis = new DataInputStream(this.cl.getInputStream());

int knowSI\_ARCH = dis.readInt();

if(knowSI\_ARCH != 0){

this.cl = new Socket(host,Servidores.get(i).getPuerto());

ObjectInputStream ois = new ObjectInputStream(this.cl.getInputStream());

Servidores.get(i).archivo.add((File)ois.readObject());

System.out.println("entroIF");

this.archivoRecibido = 1;

switch(i){

case 0:ch1.setSelected(true);break;

case 1:ch2.setSelected(true);break;

case 2:ch3.setSelected(true);break;

case 3:ch4.setSelected(true);break;

case 4:ch5.setSelected(true);break;

}

}

}else{

System.out.println("NO contiene archivos");

}

/\*if(knowIfExist(Servidores.get(i).archivo,archivoBuscar)){

switch(i){

case 0:ch1.setSelected(true);break;

case 1:ch2.setSelected(true);break;

case 2:ch3.setSelected(true);break;

case 3:ch4.setSelected(true);break;

case 4:ch5.setSelected(true);break;

}

}\*/

System.out.println("salioCliente");

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

//this.archivosRee = new ArrayList();

for(int i=0;i<Servidores.size();i++){

if(!Servidores.get(i).archivo.isEmpty()){

for(int j=0;j<Servidores.get(i).archivo.size();j++){

this.archivosRee.add(this.Servidores.get(i).archivo.get(j));

}

}

}

return this.archivosRee;

/\*this.cl = new Socket(host,pto);

ObjectInputStream bis = new ObjectInputStream(this.cl.getInputStream());

this.archivos = (ArrayList<File>) bis.readObject();

this.cl = new Socket(host,pto+2);

DataInputStream dis = new DataInputStream(this.cl.getInputStream());

int ServidorPto = dis.readInt();

//if(!Servidores.contains(ServidorPto)){

box.addItem(ServidorPto);

//}\*/

}

public void download(String host, int pto, File destino, File seleccionado, int totalCheck) throws IOException, ClassNotFoundException{

int total=0;

/\*for(int i=0;i<this.Servidores.size();i++){

for(int j=0;j<this.Servidores.get(i).archivo.size();j++){

total++;

}

}\*/

for(int i=0;i<this.Servidores.size();i++){

if(!this.Servidores.get(i).archivo.isEmpty()){

if(total<=totalCheck-1){

JOptionPane.showMessageDialog(null,"Descarga de Servidor");

total++;

}

for(int j=0;j<this.Servidores.get(i).archivo.size();j++){

this.cl = new Socket(host,this.Servidores.get(i).puerto+1);

ObjectOutputStream oos = new ObjectOutputStream(this.cl.getOutputStream());

oos.writeObject(seleccionado);

InputStream is = cl.getInputStream();

FileOutputStream fos = new FileOutputStream(destino);

int b,cont=0;

int totalLeidos = is.available()/totalCheck;

System.out.println(totalLeidos+"|"+is.available());

while((b = is.read()) != -1){

fos.write(b);

cont++;

}

fos.close();

}

}

}

JOptionPane.showMessageDialog(null,"Se descargo el archivo correctamente");

//this.cl.close();

/\*FileOutputStream fos = new FileOutputStream(destino);

int total=0;

for(int i=0;i<this.Servidores.size();i++){

for(int j=0;j<this.Servidores.get(i).archivo.size();j++){

total++;

}

}

System.out.println(total);

int empezar=0, terminar;

for(int i=0;i<this.Servidores.size();i++){

if(!this.Servidores.get(i).archivo.isEmpty()){

this.cl = new Socket(host,this.Servidores.get(i).puerto+1);

ObjectOutputStream oos = new ObjectOutputStream(this.cl.getOutputStream());

for(int j=0;j<this.Servidores.get(i).archivo.size();j++){

oos.writeObject(this.Servidores.get(i).archivo.get(j).getAbsolutePath());

InputStream is = cl.getInputStream();

byte b[] = new byte[1024];

int cont=0;

int tot = is.available()/total;

BufferedInputStream bin = new BufferedInputStream(is);

byte[] contents = new byte[1024];

int bytesRead = 0;

while ((bytesRead = bin.read(contents)) != -1) {

cont += contents.length;

if(cont >= tot){

j = this.Servidores.get(i).archivo.size();

break;

}else{

fos.write(contents, empezar, bytesRead);

}

//strFileContents = new String(contents, 0, bytesRead);

//System.out.print(strFileContents);

}

bin.close();

/\*while(is.read(b, empezar, tot) != -1){

fos.write(bytes);

if(cont == tot){

j = this.Servidores.get(i).archivo.size();

break;

}

cont++;

}

}

}

}

fos.close();\*/

}

}

package servidor\_paquete;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.InputStream;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.OutputStream;

import java.math.BigInteger;

import java.net.\*;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.ArrayList;

import java.util.logging.Level;

import java.util.logging.Logger;

import md5.MD5\_Check;

public class Servidor implements Runnable{

public ServerSocket ss,serverFile,serv;

public ArrayList<File> archivos;

public Servidor(int pto, ArrayList<File> archivos) throws IOException{

this.ss = new ServerSocket(pto);

this.serverFile = new ServerSocket(pto+1);

this.serv = new ServerSocket(pto+2);

this.archivos = archivos;

}

@Override

public void run() {

System.out.println("Servidor Iniciado...");

new Thread(new sendList(this.ss,this.archivos)).start();

new Thread(new sendFile(this.serverFile)).start();

}

}

class sendList implements Runnable{

public ServerSocket ss;

public ArrayList<File> archivos;

sendList(ServerSocket ss, ArrayList<File> archivos){

this.ss = ss;

this.archivos = archivos;

}

@Override

public void run() {

while(true){

try {

Socket s = this.ss.accept();

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

System.out.println("sendList");

DataOutputStream dos = new DataOutputStream(s.getOutputStream());

dos.writeInt(this.archivos.size());

System.out.println("servidor...");

if(!this.archivos.isEmpty()){

s = this.ss.accept();

System.out.printf("Cliente %s conectado desde puerto %s\r\n",s.getInetAddress(),s.getPort());

DataInputStream dis = new DataInputStream(s.getInputStream());

String archivoBuscar = dis.readUTF();

//s.close();

System.out.println("=>"+archivoBuscar);

s = this.ss.accept();

int seTiene = dis.readInt();

//s.close();

System.out.println("=>"+seTiene);

//s = this.ss.accept();

if(this.archivos.isEmpty()){

s = this.ss.accept();

s = this.ss.accept();

}

for(int i=0;i<this.archivos.size();i++){

if(seTiene == 0){ //No se ha encontrado el archivo

if(this.archivos.get(i).getAbsolutePath().toLowerCase().contains(archivoBuscar)){

s = this.ss.accept();

//MD5\_Check md5 = new MD5\_Check();

String md5\_S = getCodigoHash(this.archivos.get(i).getAbsolutePath());

System.out.println(archivoBuscar+"|"+md5\_S);

DataOutputStream sod = new DataOutputStream(s.getOutputStream());

sod.writeUTF(md5\_S);

sod.flush();

sod.close();

s = this.ss.accept();

sod = new DataOutputStream(s.getOutputStream());

sod.writeInt(1);

sod.flush();

sod.close();

s = this.ss.accept();

ObjectOutputStream oos = new ObjectOutputStream(s.getOutputStream());

oos.writeObject(this.archivos.get(i));

oos.close();

}else{

s = this.ss.accept();

DataOutputStream sod = new DataOutputStream(s.getOutputStream());

sod.writeUTF(archivoBuscar);

sod.flush();

sod.close();

s = this.ss.accept();

sod = new DataOutputStream(s.getOutputStream());

sod.writeInt(0);

sod.flush();

sod.close();

/\*s = this.ss.accept();

ObjectOutputStream oos = new ObjectOutputStream(s.getOutputStream());

oos.writeObject("");

oos.close();\*/

}

}else{ //Ya se encontro el archivo

if(compararWithMD5(this.archivos.get(i).getAbsolutePath(),archivoBuscar)){

s = this.ss.accept();

DataOutputStream sod = new DataOutputStream(s.getOutputStream());

sod.writeUTF(archivoBuscar);

sod.flush();

sod.close();

s = this.ss.accept();

sod = new DataOutputStream(s.getOutputStream());

sod.writeInt(1);

sod.flush();

sod.close();

s = this.ss.accept();

//String md5Recibido = dis.readUTF();

//if(compararWithMD5(this.archivos.get(i).getAbsolutePath(), archivoBuscar)){

ObjectOutputStream oos = new ObjectOutputStream(s.getOutputStream());

oos.writeObject(this.archivos.get(i));

oos.close();

//}

}else{

s = this.ss.accept();

DataOutputStream sod = new DataOutputStream(s.getOutputStream());

sod.writeUTF(archivoBuscar);

sod.flush();

sod.close();

s = this.ss.accept();

sod = new DataOutputStream(s.getOutputStream());

sod.writeInt(0);

sod.flush();

sod.close();

}

}

}

System.out.println("termino servidor");

s.close();

}

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

} catch (IOException | NoSuchAlgorithmException ex) {}

}

}

public String getCodigoHash(String path) throws NoSuchAlgorithmException, FileNotFoundException, IOException{

MessageDigest digest = MessageDigest.getInstance("MD5");

File f = new File(path);

InputStream is = new FileInputStream(f);

byte[] buffer = new byte[(int) f.length()];

int read = 0;

while ((read = is.read(buffer)) > 0){

digest.update(buffer, 0, read);

}

byte[] md5sum = digest.digest();

BigInteger bigInt = new BigInteger(1, md5sum);

String output = bigInt.toString(16);

is.close();

return output;

}

/\*public static boolean Comparar(String file, String hashCode) throws NoSuchAlgorithmException, FileNotFoundException, IOException{

return hashCode.equals(getCodigoHash(file));

}\*/

public boolean comparar(String file1, String file2) throws NoSuchAlgorithmException, FileNotFoundException, IOException {

//System.out.println(getCodigoHash(file1));

//System.out.println(getCodigoHash(file2));

return getCodigoHash(file1).equals(getCodigoHash(file2));

}

public boolean compararWithMD5(String file1, String md5) throws NoSuchAlgorithmException, FileNotFoundException, IOException {

//System.out.println(getCodigoHash(file1));

//System.out.println(getCodigoHash(file2));

return getCodigoHash(file1).equals(md5);

}

}

class sendFile implements Runnable{

public ServerSocket serverFile;

sendFile(ServerSocket serverFile){

this.serverFile = serverFile;

}

@Override

public void run() {

while(true){

try {

System.out.println("sendFile");

Socket socket = this.serverFile.accept();

System.out.printf("Cliente %s conectado desde puerto %s\r\n",socket.getInetAddress(),socket.getPort());

ObjectInputStream ois = new ObjectInputStream(socket.getInputStream());

OutputStream oos = socket.getOutputStream();

File f = (File) ois.readObject();

FileInputStream fis = new FileInputStream(f);

int b;

while((b = fis.read()) != -1){

System.out.println(b);

oos.write(b);

}

socket.close();

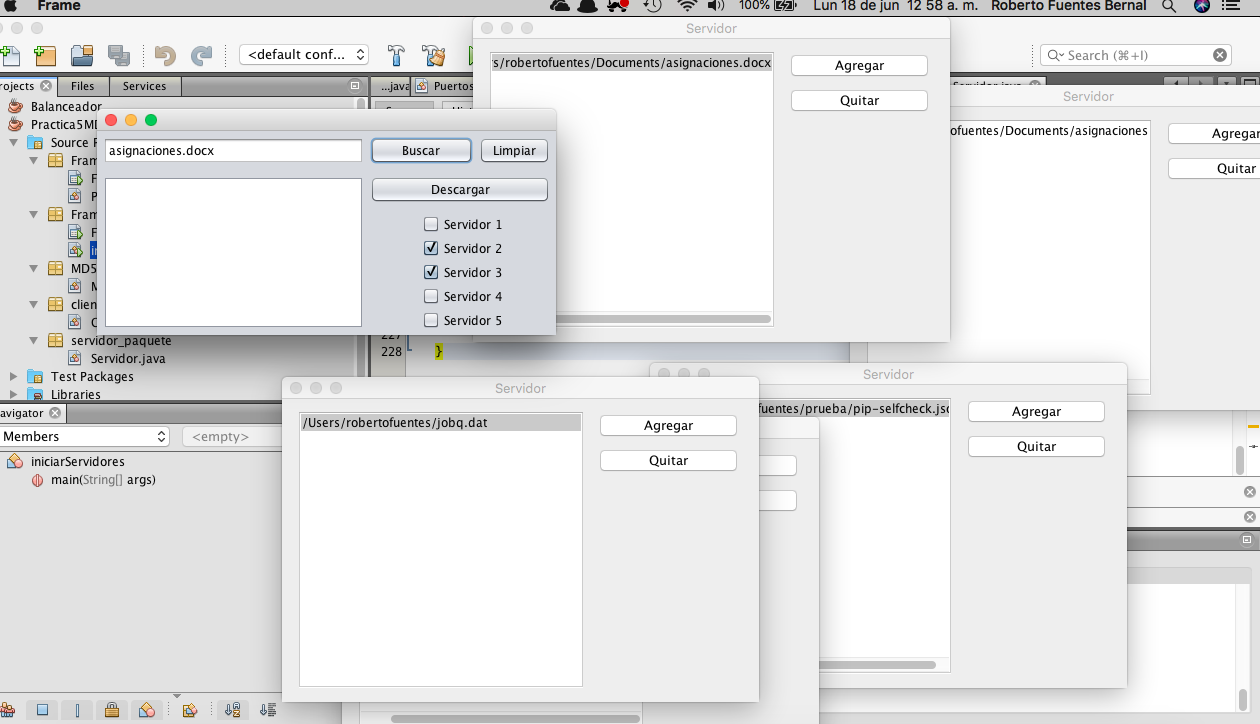
} catch (IOException | ClassNotFoundException e) {}

}

}

}

**Capturas de Pantalla.**

****

**Conclusión.**

En esta práctica reflejamos la mayoría de los conocimientos adquiridos a lo largo del semestre, desde lo primero y más básico, sockets de flujo, hasta RMI.

Siendo esta aplicación peer to peer, se nos hizo de suma relevancia el aprender cómo se desarrollan este tipo de aplicaciones puesto que son de las que más utilizamos actualmente, ya sea para la descarga de música, películas, libros, programas, etc.

Por último, la comodidad que nos brindan herramientas como RMI es agradable, pero siempre es importante saber cómo funcionan todas estas herramientas desde la raíz.

El curso nos permitió poder entender y usar todas esas herramientas, analizar el problema para poder ver que tecnología es la que más se adapta a nuestras necesidades, y eso es básicamente lo que nos llevamos de este curso.