Benemérita Universidad Autónoma De Puebla

Facultad de Ciencias de la Computación

Materia:
Programación Concurrente y Paralela (25581)

Título:
Sockets

Profesora:
Carmen Cerón Garnica

Alumno:
Jared Rai Serrano Navarro (202160998)

Periodo:
Otoño 2025

REPORTE DE PRÁCTICA: SOCKETS

1. Introducción

Como bien sabemos, una de las mayores ventajas de Java es la portabilidad gracias a la JVM, algo crucial en Internet donde conviven todo tipo de dispositivos. Precisamente para aprovechar esta ventaja en el mundo conectado de hoy, la API de Java nos ofrece bibliotecas específicas para programar aplicaciones en red.

2. Desarrollo de la práctica

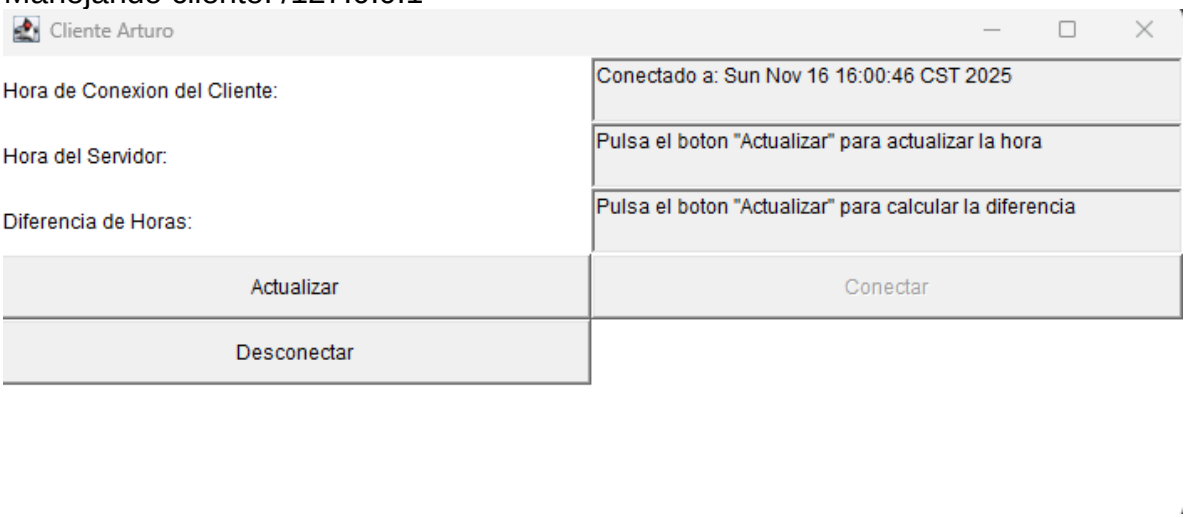
Actividades para realizar:

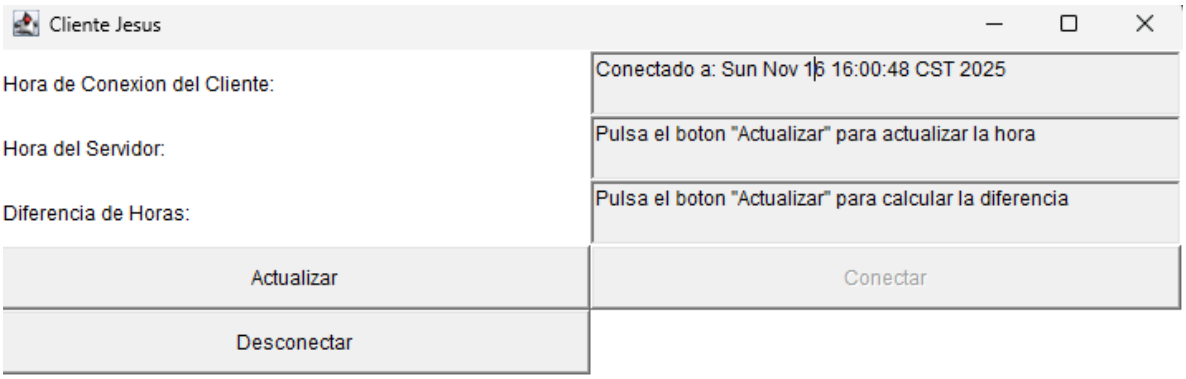
1. Para esta práctica estarás utilizando la carpeta sockets donde se encuentra 5 programas clienteservidor UDP y TCP, prueba cada uno.

Ejecución:

Realizamos la conectividad entre cliente y servidor, y nos muestra la siguiente salida en la UI.

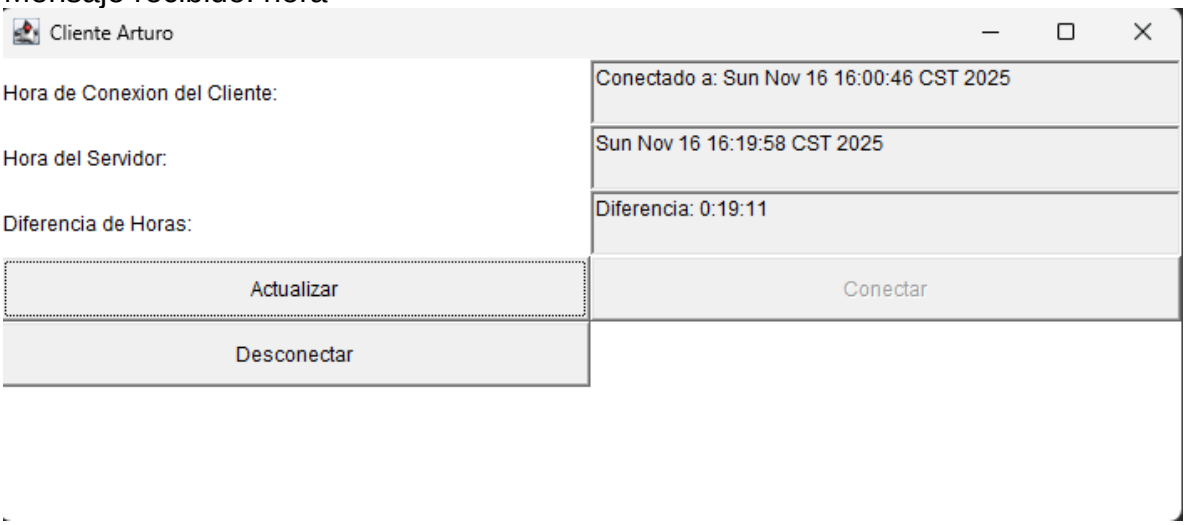
Salida de la consola:
Servidor TCP iniciado en puerto 5000...
Cliente TCP conectado: /127.0.0.1
Manejando cliente: /127.0.0.1
Cliente TCP conectado: /127.0.0.1
Manejando cliente: /127.0.0.1





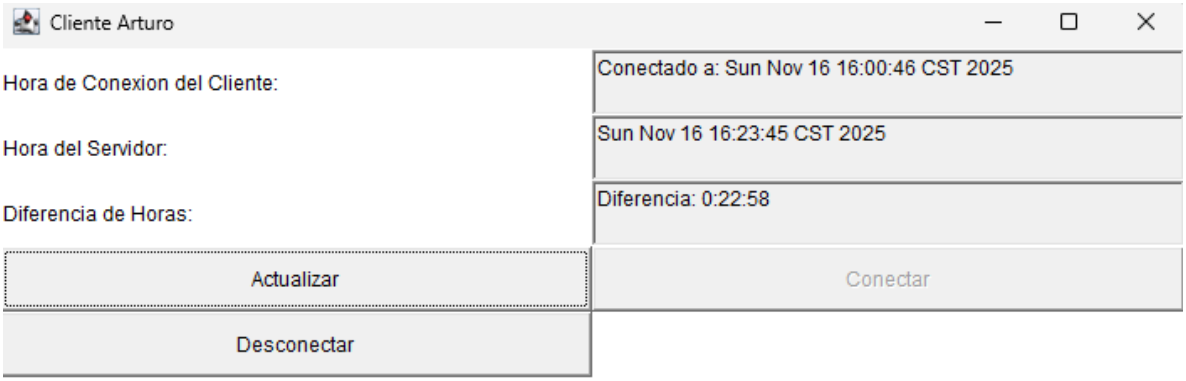
Al actualizar se muestra la hora del servidor y se actualiza la diferencia de horas que pasaron entre la hora de conexión del cliente y la hora de conexión del servidor.

Salida consola:
Mensaje recibido: hora
Hora enviada: Sun Nov 16 16:19:58 CST 2025
Mensaje recibido: hora
Hora enviada: Sun Nov 16 16:20:13 CST 2025
Mensaje recibido: hora



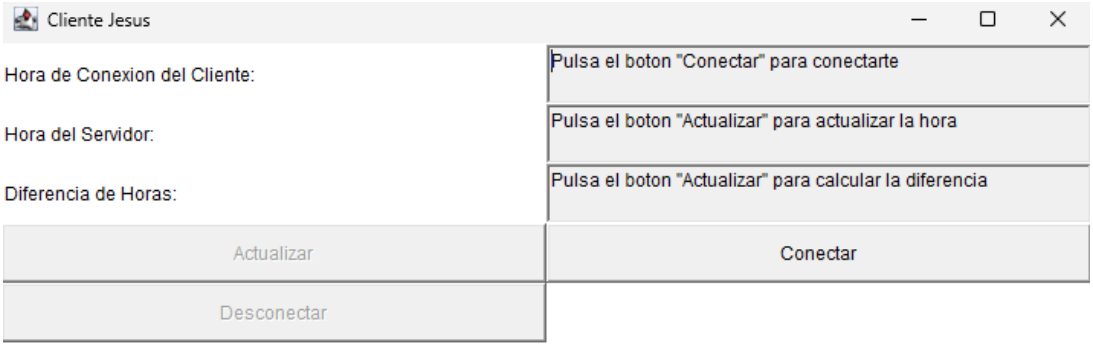
Al dar de nuevo en actualizar solo se actualiza la hora del servidor y la diferencia de horas.

Salida consola:
Hora enviada: Sun Nov 16 16:22:59 CST 2025
Mensaje recibido: hora
Hora enviada: Sun Nov 16 16:23:00 CST 2025
Mensaje recibido: hora
Hora enviada: Sun Nov 16 16:23:45 CST 2025



Al presionar el botón de desconectar nos desconecta de la conexión cliente-servidor.

Salida consola:
Mensaje recibido: salir
Cliente desconectado: /127.0.0.1



CODIGO: SERVIDOR

```
// Servidor.java
import java.io.*;
import java.net.*;
import java.util.*;
```

```
class Servidor extends Thread  // Agregué public
{
   public static Vector usuarios = new Vector();

   public static void main (String args[])
   {
      ServerSocket sfd = null;
      try
      {
         sfd = new ServerSocket(7000);
         System.out.println("Servidor de Chat iniciado en puerto 7000...");
      }
      catch (IOException ioe)
      {
         System.out.println("Comunicación rechazada."+ioe);
         System.exit(1);
      }

      while (true)
      {
         try
         {
            Socket nsfd = sfd.accept();
            System.out.println("Conexion aceptada de: "+nsfd.getInetAddress());
            Flujo flujo = new Flujo(nsfd);
            Thread t = new Thread(flujo);
            t.start();
         }
         catch(IOException ioe)
         {
            System.out.println("Error: "+ioe);
         }
      }
   }
}
```

CODIGO: CLIENTE

```
// Cliente.java
import java.io.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;


public class Cliente extends Frame implements ActionListener
{
   static Socket sfd = null;
   static DataInputStream EntradaSocket;
   static DataOutputStream SalidaSocket;
   static TextField salida;
   static TextArea entrada;
   String texto;

   public Cliente()
```

```
    {
        setTitle("Chat");
        setSize(350,200);
        salida = new TextField(30);
        salida.addActionListener(this);

        entrada = new TextArea();
        entrada.setEditable(false);

        add("South",salida);
        add("Center", entrada);
        setVisible(true);
    }

    public static void main(String[] args)
    {
        Cliente cliente = new Cliente();
        try
        {
            sfd = new Socket("localhost",7000);  // o "127.0.0.1"
            EntradaSocket = new DataInputStream(new BufferedInputStream(sfd.getInputStream()));
            SalidaSocket = new DataOutputStream(new
BufferedOutputStream(sfd.getOutputStream()));
        }
        catch (UnknownHostException uhe)
        {
            System.out.println("No se puede acceder al servidor.");
            System.exit(1);
        }
        catch (IOException ioe)
        {
            System.out.println("Comunicación rechazada.");
            System.exit(1);
        }
        while (true)
        {
            try
            {
                String linea = EntradaSocket.readUTF();
                entrada.append(linea+"\n");
            }
            catch(IOException ioe)
            {
                System.exit(1);
            }
        }
    }

    public void actionPerformed (ActionEvent e)
    {
        texto = salida.getText();
        salida.setText("");
        try
        {
            SalidaSocket.writeUTF(texto);
            SalidaSocket.flush();
```

```
        }
        catch (IOException ioe)
        {
            System.out.println("Error: "+ioe);
        }
    }

    public boolean handleEvent(Event e)
    {
        if ((e.target == this) && (e.id == Event.WINDOW_DESTROY))
        {
            if (sfd != null)
            {
                try
                {
                    sfd.close();
                }
                catch (IOException ioe)
                {
                    System.out.println("Error: "+ioe);
                }
                this.dispose();
            }
        }
        return true;
    }
}
```

CODIGO: CLIENTETCP

```
//ClienteTCP.java
import java.awt.*;
import java.net.*;
import java.io.*;
import java.awt.event.*;
import java.util.*;

class ClienteTCP extends Frame implements ActionListener {
    Panel panel;
    Socket conexion;
    TextField textent, textent2, textent3;
    Button actualizar, conectar, desconectar;
    DataOutputStream salida;
    DataInputStream entrada;
    long horaConexion;

    ClienteTCP(String nombre) {
        super(nombre);
        setSize(350, 250); // Aumenté el tamaño
        panel = new Panel();
        panel.setLayout(new GridLayout(7, 2));

        textent = new TextField(30);
        textent.setText("Pulsa el boton \"Conectar\" para conectarte");
        textent.setEditable(false);

        textent2 = new TextField(30);
```

```
    textent2.setText("Pulsa el boton \"Actualizar\" para actualizar la hora");
    textent2.setEditable(false);

    textent3 = new TextField(30);
    textent3.setText("Pulsa el boton \"Actualizar\" para calcular la diferencia");
    textent3.setEditable(false);

    actualizar = new Button("Actualizar");
    actualizar.setEnabled(false);
    conectar = new Button("Conectar");
    desconectar = new Button("Desconectar");
    desconectar.setEnabled(false);

    panel.add(new Label("Hora de Conexion del Cliente:"));
    panel.add(textent);
    panel.add(new Label("Hora del Servidor:"));
    panel.add(textent2);
    panel.add(new Label("Diferencia de Horas:"));
    panel.add(textent3);
    panel.add(actualizar);
    panel.add(conectar);
    panel.add(desconectar);

    actualizar.addActionListener(this);
    conectar.addActionListener(this);
    desconectar.addActionListener(this);
    addWindowListener(new Cerrar());
    add(panel);
    setVisible(true);
}

public void actionPerformed(ActionEvent e) {
    String com = e.getActionCommand();

    if (com.equals("Actualizar")) {
        try {
            // Enviar solicitud y recibir hora del servidor
            salida.writeUTF("hora");
            salida.flush();
            String horaServidor = entrada.readUTF();
            textent2.setText(horaServidor);

            // Calcular diferencia
            long horaActual = System.currentTimeMillis();
            long diferencia = horaActual - horaConexion;
            long segundos = diferencia / 1000;
            long minutos = segundos / 60;
            long horas = minutos / 60;

            textent3.setText(String.format("Diferencia: %d:%02d:%02d",
                    horas, minutos % 60, segundos % 60));

        } catch(IOException excepcion) {
            textent2.setText("Error al comunicar con servidor");
        }
    } else if (com.equals("Conectar")) {
```

```
            try {
                conexion = new Socket("localhost", 5000); // Cambié a localhost
                salida = new DataOutputStream(conexion.getOutputStream());
                entrada = new DataInputStream(conexion.getInputStream());

                conectar.setEnabled(false);
                desconectar.setEnabled(true);
                actualizar.setEnabled(true);

                horaConexion = System.currentTimeMillis();
                Date fechaConexion = new Date(horaConexion);
                textent.setText("Conectado a: " + fechaConexion.toString());

            } catch(IOException excepcion) {
                textent.setText("Error al conectar con servidor");
            }
        } else if (com.equals("Desconectar")) {
            try {
                if (salida != null) {
                    salida.writeUTF("salir");
                    salida.flush();
                }
                if (conexion != null) {
                    conexion.close();
                }
                conectar.setEnabled(true);
                desconectar.setEnabled(false);
                actualizar.setEnabled(false);
                textent.setText("Pulsa el boton \"Conectar\" para conectarte");
                textent2.setText("Pulsa el boton \"Actualizar\" para actualizar la hora");
                textent3.setText("Pulsa el boton \"Actualizar\" para calcular la diferencia");
            } catch(IOException excepcion) {
                System.out.println("Error al desconectar: " + excepcion);
            }
        }
    }

    class Cerrar extends WindowAdapter {
        public void windowClosing(WindowEvent e) {
            try {
                if (conexion != null) {
                    conexion.close();
                }
            } catch (IOException ex) {
                System.out.println("Error al cerrar: " + ex);
            }
            dispose();
            System.exit(0);
        }
    }

    public static void main(String args[]) {
        new ClienteTCP("Cliente Arturo");
        new ClienteTCP("Cliente Jesus");
    }
}
```

CODIGO: FLUJO

```
//Flujo.java
import java.net.*;
import java.io.*;
import java.util.*;

class Flujo extends Thread {
   Socket nsfd;
   DataInputStream FlujoLectura;
   DataOutputStream FlujoEscritura;

   public Flujo(Socket sfd) {
      nsfd = sfd;
      try {
         FlujoLectura = new DataInputStream(new BufferedInputStream(sfd.getInputStream()));
         FlujoEscritura = new DataOutputStream(new
BufferedOutputStream(sfd.getOutputStream()));
      } catch(IOException ioe) {
         System.out.println("IOException(Flujo): "+ioe);
      }
   }

   public void run() {
      broadcast(nsfd.getInetAddress()+"> se ha conectado");
      Servidor.usuarios.add(this);

      while(true) {
         try {
            String linea = FlujoLectura.readUTF();
            if (linea != null && !linea.trim().equals("")) {
               String mensaje = nsfd.getInetAddress() + "> " + linea;
               broadcast(mensaje);
            }
         } catch(IOException ioe) {
            Servidor.usuarios.remove(this);
            broadcast(nsfd.getInetAddress()+"> se ha desconectado");
            break;
         }
      }
   }

   public void broadcast(String mensaje) {
      synchronized (Servidor.usuarios) {
         for (int i = 0; i < Servidor.usuarios.size(); i++) {
            Flujo f = (Flujo) Servidor.usuarios.get(i);
            try {
               synchronized(f.FlujoEscritura) {
                  f.FlujoEscritura.writeUTF(mensaje);
                  f.FlujoEscritura.flush();
               }
            } catch(IOException ioe) {
               System.out.println("Error en broadcast: "+ioe);
            }
         }
      }
```

```
    }
}
```

CODIGO: SERVIDORTCP

```
import java.net.*;
import java.io.*;
import java.util.*;

public class ServidorTCP {
    public static void main(String[] args) {
        try {
            ServerSocket server = new ServerSocket(5000);
            System.out.println("Servidor TCP iniciado en puerto 5000...");

            while (true) {
                Socket cliente = server.accept();
                System.out.println("Cliente TCP conectado: " + cliente.getInetAddress());
                new ManejadorCliente(cliente).start();
            }
        } catch (IOException ioe) {
            System.out.println("Error en servidor TCP: " + ioe);
        }
    }
}

class ManejadorCliente extends Thread {
    private Socket cliente;
    private DataInputStream entrada;
    private DataOutputStream salida;

    public ManejadorCliente(Socket socket) {
        this.cliente = socket;
    }

    public void run() {
        try {
            entrada = new DataInputStream(cliente.getInputStream());
            salida = new DataOutputStream(cliente.getOutputStream());

            System.out.println("Manejando cliente: " + cliente.getInetAddress());

            while (true) {
                String mensaje = entrada.readUTF();
                System.out.println("Mensaje recibido: " + mensaje);

                if (mensaje.equals("salir")) {
                    System.out.println("Cliente desconectado: " + cliente.getInetAddress());
                    break;
                } else if (mensaje.equals("hora")) {
                    Date ahora = new Date();
                    String horaServidor = ahora.toString();
                    salida.writeUTF(horaServidor);
                    salida.flush();
                    System.out.println("Hora enviada: " + horaServidor);
                }
            }
```

```
            cliente.close();
        } catch (IOException ioe) {
            System.out.println("Error con cliente: " + ioe);
        }
    }
}
```