Instituto Politécnico Nacional

Escuela Superior de Cómputo

Desarrollo de Sistemas Distribuidos

Prof. **Benjamín Cruz Torres**

Práctica No. 2
Reloj Distribuido

Grupo: 4CV3

Equipo: 6

Integrantes:

1. Acosta Rosales Jair Sebastián
2. De Jesús López David
3. Galicia Vargas Gerardo
4. Martínez Marcos Luis Eduardo
5. Octaviano Lima Elvia Jaqueline

Fecha: 13 de septiembre de 2018

Práctica 2: Reloj Distribuido

Objetivo de la Práctica Que el alumno comprenda la importancia de la programación distribuida y el trabajo en equipo.

Tecnologías a aplicar: Sockets, RMI, SOAP, Multi-cast, Hilos (threads), POO, Protocolos de comunicación.

Actividades

A partir de la práctica 1, desarrollará una aplicación para controlar un reloj digital a distancia. De acuerdo a los siguientes requerimientos:

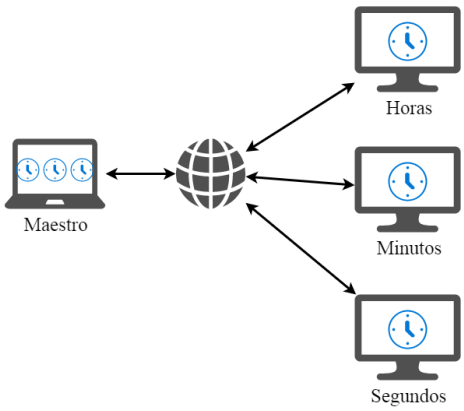


Figura 1. Reloj distribuido

Requerimientos funcionales

- Los cuatro relojes están en diferentes computadoras (Máquinas virtuales) secundarias.
- En el *maestro* hay un reloj digital.
- Cada elemento del reloj (horas, minutos, segundos) en *el maestro* tiene asociada una computadora secundaria diferente.
- En *el maestro* hay tres botones *Modificar* y tres *Enviar*, cada uno por cada elemento del reloj.
- El botón *Modificar*, permitirá modificar el elemento correspondiente en el reloj.
- El botón *Enviar*, enviará la modificación a la computadora del elemento correspondiente.
- No se pueden modificar los elementos en las computadoras secundarias.
- Inicialmente el segundero cambia cada segundo.
- El formato de hora es de 24 hrs.

Requerimientos no funcionales

- Al pulsar el botón modificar, permitirá cambiar el elemento asignado a dicho botón.
- Al pulsar el botón de enviar. Se modificará el elemento del reloj en la computadora correspondiente.
- Al inicio el reloj maestro tendrá una hora elegida al azar. La cual enviará directamente a las computadoras secundarias.
- Se podrá cambiar la velocidad de actualización del segundero.

Investigación

Comunicación cliente/servidor con Socket TCP

El interfaz Java que da soporte a sockets TCP está constituida por las clases *ServerSocket* y *Socket*.

1. *ServerSocket*: es utilizada por un servidor para crear un socket en el puerto en el que escucha las peticiones de conexión de los clientes. Su método *accept* toma una petición de conexión de la cola, o si la cola está vacía, se bloquea hasta que llega una petición. El resultado de ejecutar *accept* es una instancia de *Socket*, a través del cual el servidor tiene acceso a los datos enviados por el cliente.
2. *Socket*: es utilizada tanto por el cliente como por el servidor. El cliente crea un socket especificando el nombre DNS del host y el puerto del servidor, así se crea el socket local y además se conecta con el servicio. Esta clase proporciona los métodos *getInputStream* y *getOutputStream* para acceder a los dos *streams* asociados a un socket (recordemos que son bidireccionales), y devuelve tipos de datos *InputStream* y *OutputStream*, respectivamente, a partir de los cuales podemos construir *BufferedReader* y *PrintWriter*, respectivamente, para poder procesar los datos de forma más sencilla.

Si nos centramos en la parte de comunicaciones, la forma general de implementar un cliente será:

1. Crear un objeto de la clase *Socket*, indicando host y puerto donde corre el servicio.
2. Obtener las referencias al *stream* de entrada y al de salida al *socket*.
3. Leer desde y escribir en el *stream* de acuerdo al protocolo del servicio. Para ello emplear alguna de las facilidades del paquete java.io.
4. Cerrar los *streams*.
5. Cerrar el *socket*.

La forma de implementar un servidor será:

1. Crear un objeto de la clase *ServerSocket* para escuchar peticiones en el puerto asignado al servicio.
2. Esperar solicitudes de clientes
3. Cuando se produce una solicitud:
    - Aceptar la conexión obteniendo un objeto de la clase *Socket*
    - Obtener las referencias al *stream* de entrada y al de salida al *socket* anterior.
    - Leer datos del *socket*, procesarlos y enviar respuestas al cliente, escribiendo en el *stream* del *socket*. Para ello emplear alguna de las facilidades del paquete java.io.
4. Cerrar los *streams*.
5. Cerrar los *sockets*.

Introducción

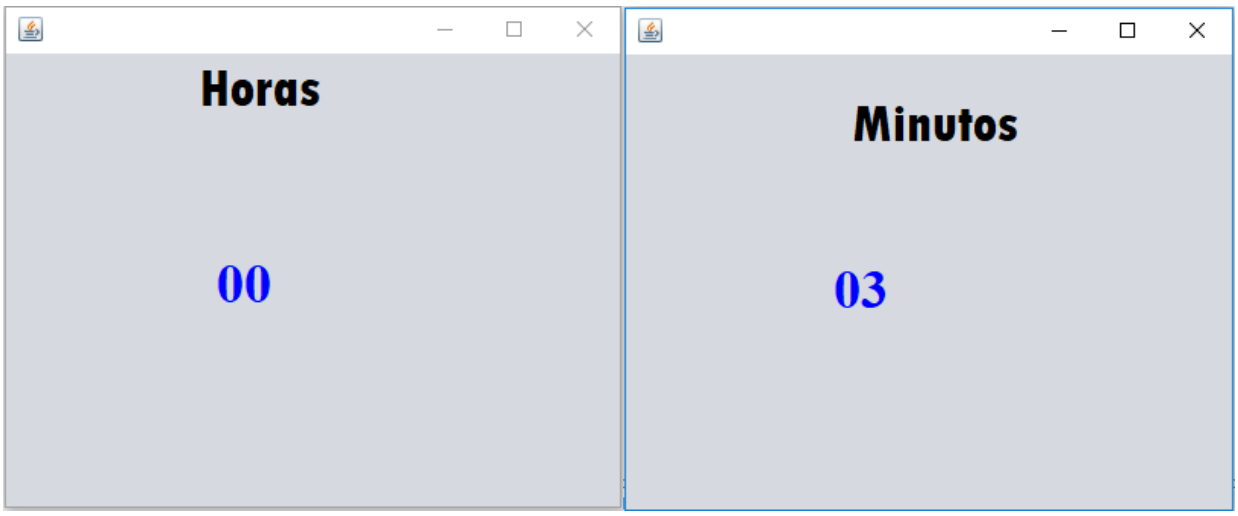
La presente, es una implementación de sockets e hilos para una comunicación cliente servidor, donde un cliente con un reloj que se instancia tomando la hora del sistema envía datos a tres servidores, cada uno de los cuales toma ese dato para actualizar su propio reloj local, mostrando solo alguno de tres valores: horas, minutos, o segundos.

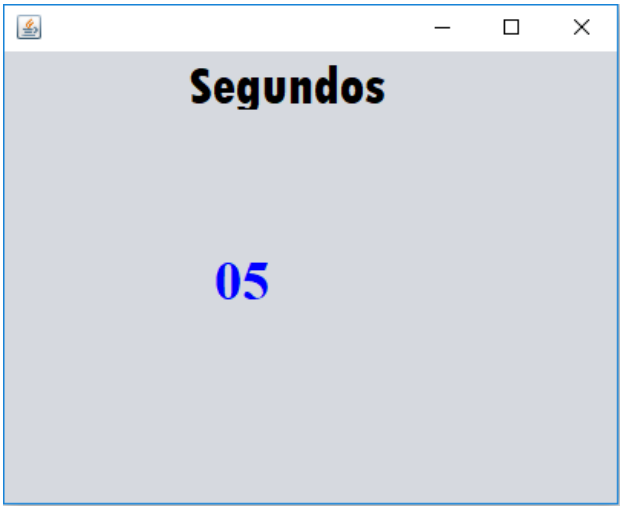
Desarrollo de la práctica

Para la presente práctica se retomaron los relojes de la primera, incluyendo la funcionalidad de tales relojes.

Para su resolución se consideró instanciar tres servidores que se mantuvieran a la escucha de los valores enviados por el único cliente, el cual funge como maestro pues éste toma la hora del sistema, muestra los valores de horas, minutos y segundos, y es el único sobre el que se pueden modificar dichos valores y enviarlos a los servidores para que éstos repliquen los valores.

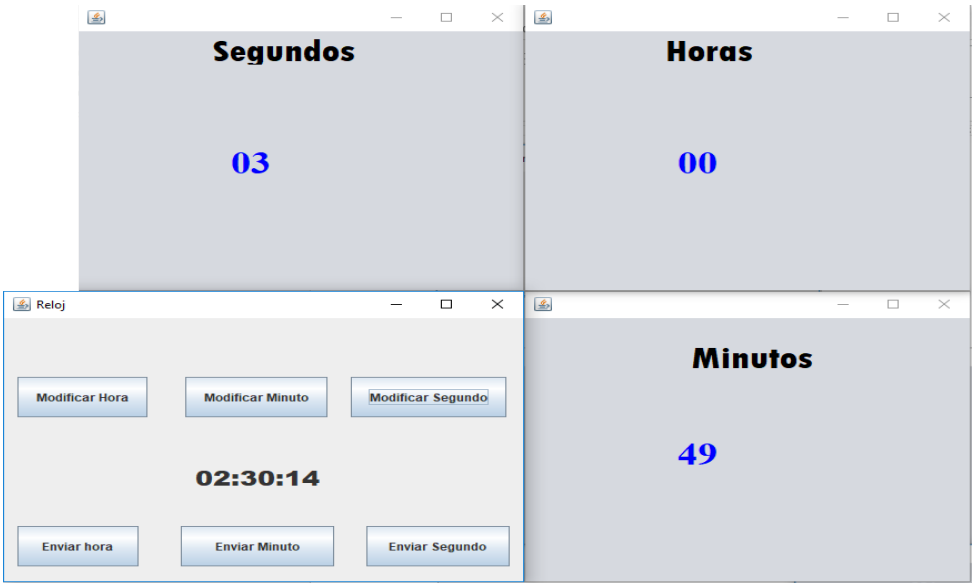
Por su parte, los servidores también se instancian con la hora del sistema (para mostrar una hora inicial cualquiera).





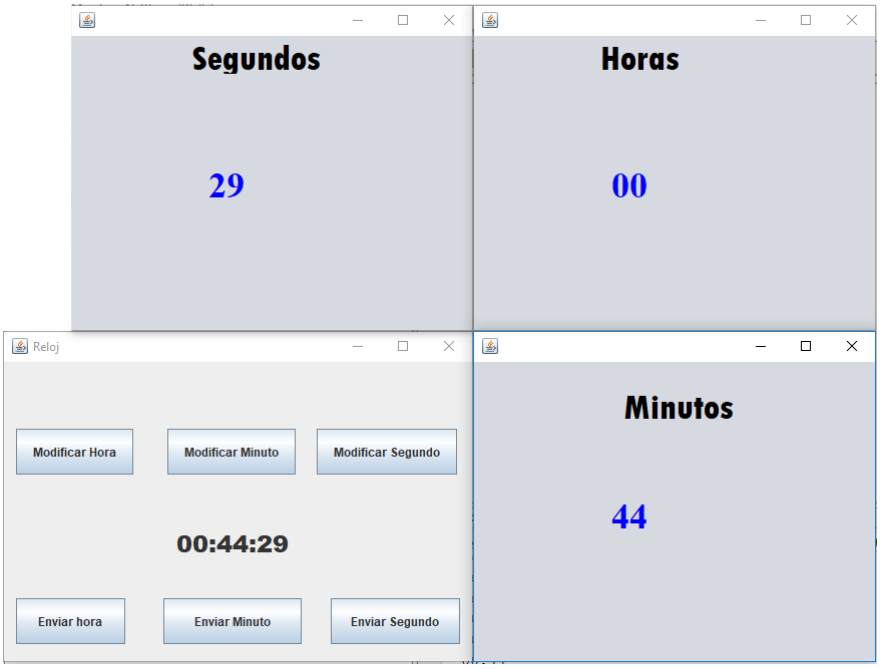
De acuerdo con los requerimientos de la práctica, se proporciona una interfaz que permite modificar los valores de horas, minutos y segundos, así como enviar cada uno de éstos por separado.

Así, se proporciona una interfaz auxiliar por cada valor que se puede modificar.

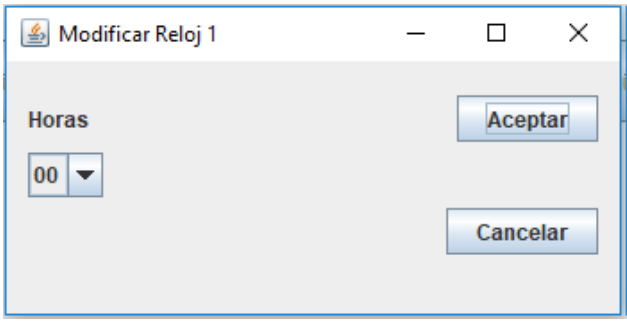


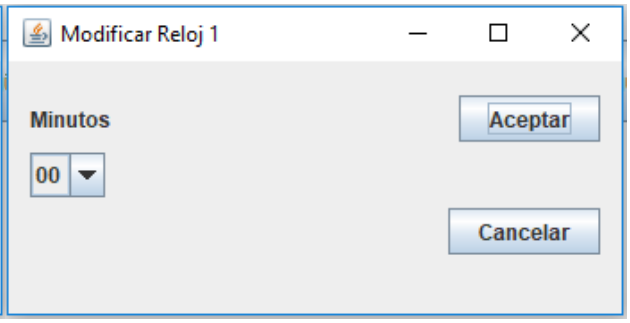
Pruebas

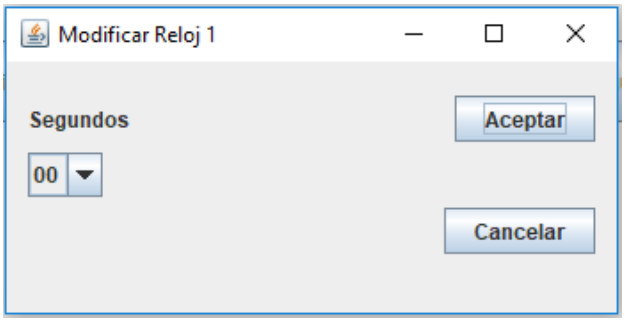
Como pruebas, inicialmente se ejecutan las clases principales de los tres servidores y el cliente; así, cada interfaz toma la hora del sistema y muestra los valores correspondientes.



En la interfaz del cliente se ingresan valores para modificar las horas, minutos y segundos del reloj maestro, con lo que la hora del cliente obtiene valores independientes de la hora del sistema, aunque los valores se calculan y actualizan cada segundo.

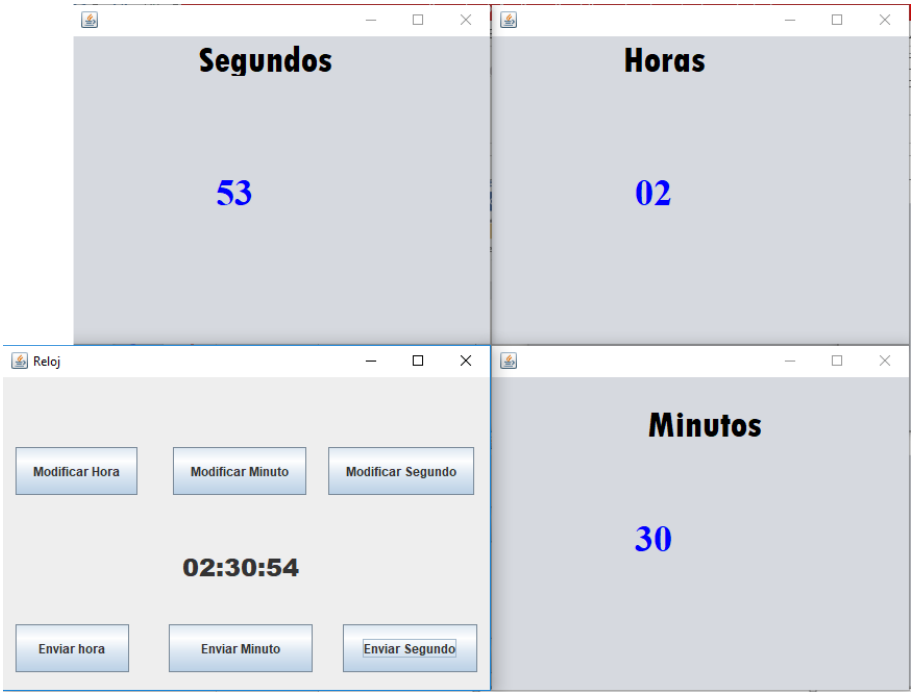






Cada vez que se ingresa un valor se observa cómo la hora mostrada se modifica y se actualiza cada segundo.

Finalmente, cada vez que se envía un valor, se sincroniza con el del correspondiente servidor. A su vez, la hora de cada servidor, aunque con sus valores modificados, continúa actualizándose cada segundo.



Conclusiones

La utilización de sockets e hilos permite implementaciones de sistemas distribuidos, aunque la presente haya sido un ejemplo bastante sencillo, pues únicamente se envía un dato cada vez que se solicita, desde un cliente a un servidor en particular, pudiendo enviar datos a tres de éstos.

Aunque inicialmente se consideró el uso de otras tecnologías, finalmente se optó por la solución de implementación que resultaba más simple y con menor tiempo de desarrollo.

Bibliografía

Trabajando con sockets TCP. (2003). Universidad Carlos III de Madrid. Departamento de Ingeniería Telemática. Recuperado de: http://www.it.uc3m.es/celeste/docencia/cr/2003/PracticaSocketsTCP/