Instituto Politécnico Nacional
Escuela Superior de Cómputo

Tarea 1

"Cálculo de PI"

Integrantes:

Jaime Villanueva Héctor Israel

Juárez Espinoza Ulises

Machorro Vences Ricardo Alberto

Equipo 7

Desarrollo de sistemas distribuidos

4CV11

19 de febrero de 2022

Introducción

En una red cliente-servidor, el dispositivo que solicita información o requiere algún servicio se denomina “cliente”, y el dispositivo que escucha esta solicitud y da una respuesta se denomina “servidor”. Este tipo de red mantiene una comunicación bidireccional permitiendo el flujo de datos en ambas direcciones. Para crear esta conexión se puede hacer uso de sockets que se pueden ver como puentes para poder transportar o enviar nuestros datos entre dispositivos.

En esta práctica, se va a realizar un programa donde pongamos lo anterior a prueba. Vamos a hacer un cliente que pueda obtener el valor de PI y que, para lograrlo, va a sumar el resultado que lleguen de los cuatro servidores.

Objetivo

Aprender el funcionamiento y el desarrollo de una red sencilla cliente-servidor haciendo uso de sockets y multihilos.

Desarrollo

Se quiere calcular una aproximación de PI utilizando la serie de Gregory-Leibniz. Para lograrlo, se va a desarrollar un programa distribuido en el que el cliente va a conectarse a cuatro servidores y estos deben dar un resultado, el cliente sumará estos y ese valor se mostrará como resultado final.

Clase “Mensajero”

Para inicializarla se requiere pasarle el puerto que se va a ocupar y, para este caso, el servidor será la computadora local por lo que no fue necesario pedirlo.

Esta clase tiene la función de conectar con el servidor indicado y obtener un valor de tipo double de este, el cual vamos a sumarlo al valor de PiValue (que es la variable que va a contener nuestro resultado final).

Algo en tener en cuenta es que se utilizó la instrucción synchronized, el cual utiliza un objeto “obj” para sincronizar los hilos. Al utilizar esta instrucción, nuestro programa se vuelve un poco mas lento debido a que se obliga que ciertas partes del programa se ejecuten en serie y no en paralelo.

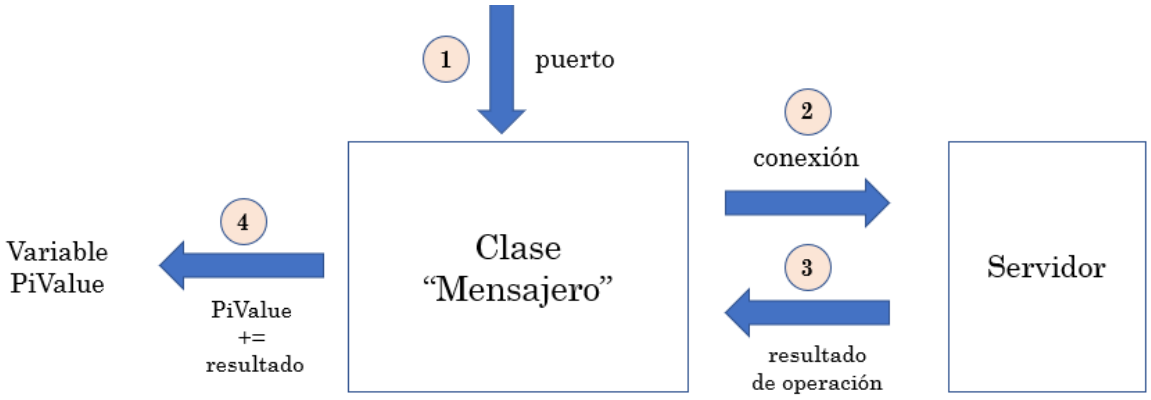

Ilustración 1 Funcionamiento de clase Mensajero

Clase Servidor

Para inicializar es necesario pasarle el número de nodo (para esta práctica solo se permiten cuatro números: 1, 2, 3 y 4) y el número de puerto.

Esta clase va a crear un socket servidor con el puerto indicado y va a ocupar el método accept() para que el hilo principal del programa quedé en estado de espera hasta recibir una conexión del cliente, y esto se va a asignar a una variable de tipo socket.

Cuando se reciba la señal de un cliente, se crea una conexión (DataOutputStream) con el que podremos enviarle el resultado de la operación al cliente.

Por último, se cierra la conexión.

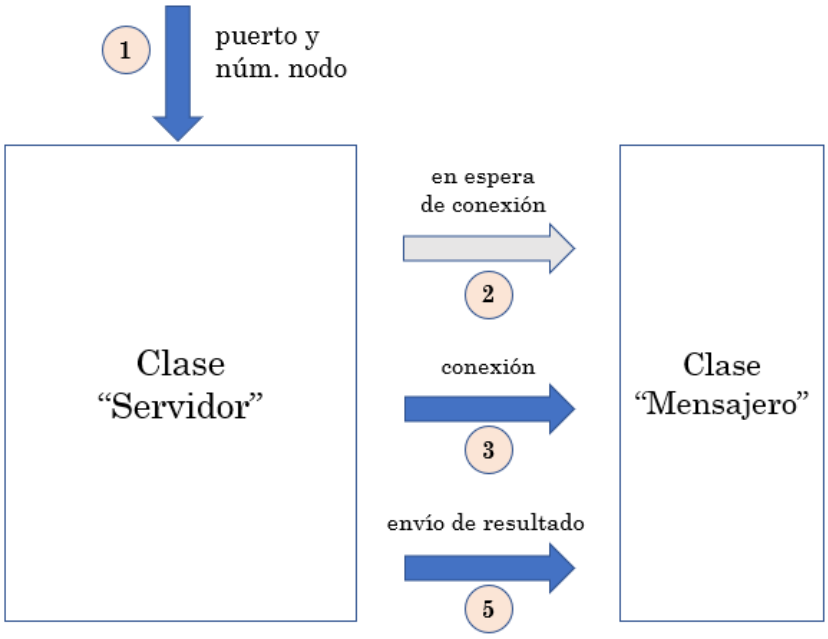

Ilustración 2Funcionamiento de clase Servidor

Clase CalcularPI

Esta es la clase que inicia todo el programa. Para ejecutarlo, hay que pasarle como argumento un número del 0 al 4, si se escoge el 0 entonces se estaría ejecutando el cliente y si es 1, 2, 3 o 4, un servidor.

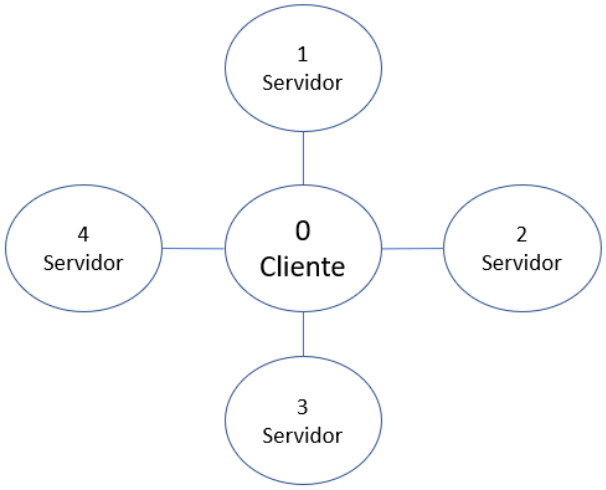

Ilustración 3 Conexión cliente con servidores

Para la parte del cliente, se declararon cuatro variables de tipo Mensajero a los cuales se les pasará el número de puerto del servidor. Después se invoca el método start() en cada una de las variables declaradas anteriormente, que tendrá la función de crear un nuevo hilo y correr el método run() de la clase Mensajero. Se invoca el método join() para que el hilo principal (el hilo que invocó el método start())espere a que termine de ejecutar el hilo de los servidores. El resultado se verá reflejado en nuestra variable global de tipo double llamada PiValue, el cual la mostraremos por pantalla.

Y para la parte del servidor, solo se muestra por pantalla que se está ejecutando un servidor, se declara una variable de tipo Servidor al cual le pasaremos el puerto que vamos a ocupar y por último, ejecutaremos el método empieza() para que ejecute el código contenido en dicha clase.

Compilamos el programa, comprobamos que no nos da errores, así como los archivos y el ejecutable que genera.

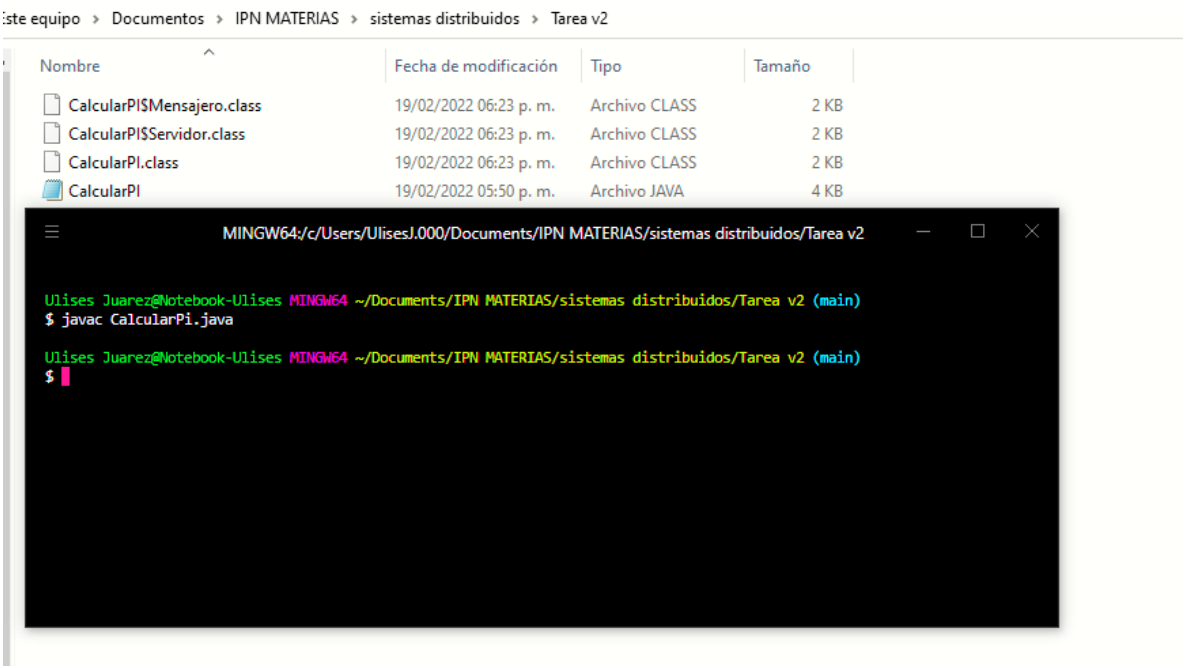

Ilustración 4 Compilación y archivos

Como se muestra a continuación no es necesario que los servidores estén activos para poder correr el cliente, este hace reintentos de reconexión, en pocas palabras espera a los servidores.

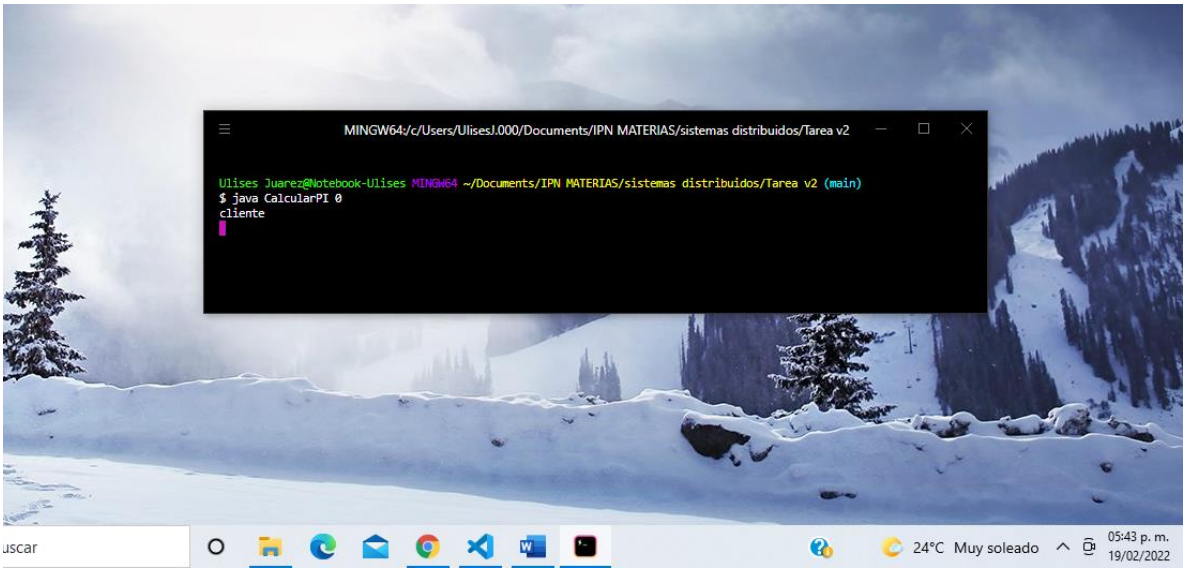

Ilustración 5 Cliente con intentos de reconexión

Ahora ejecutare algunos servidores en un orden aleatorio, ya que tampoco es de importancia.

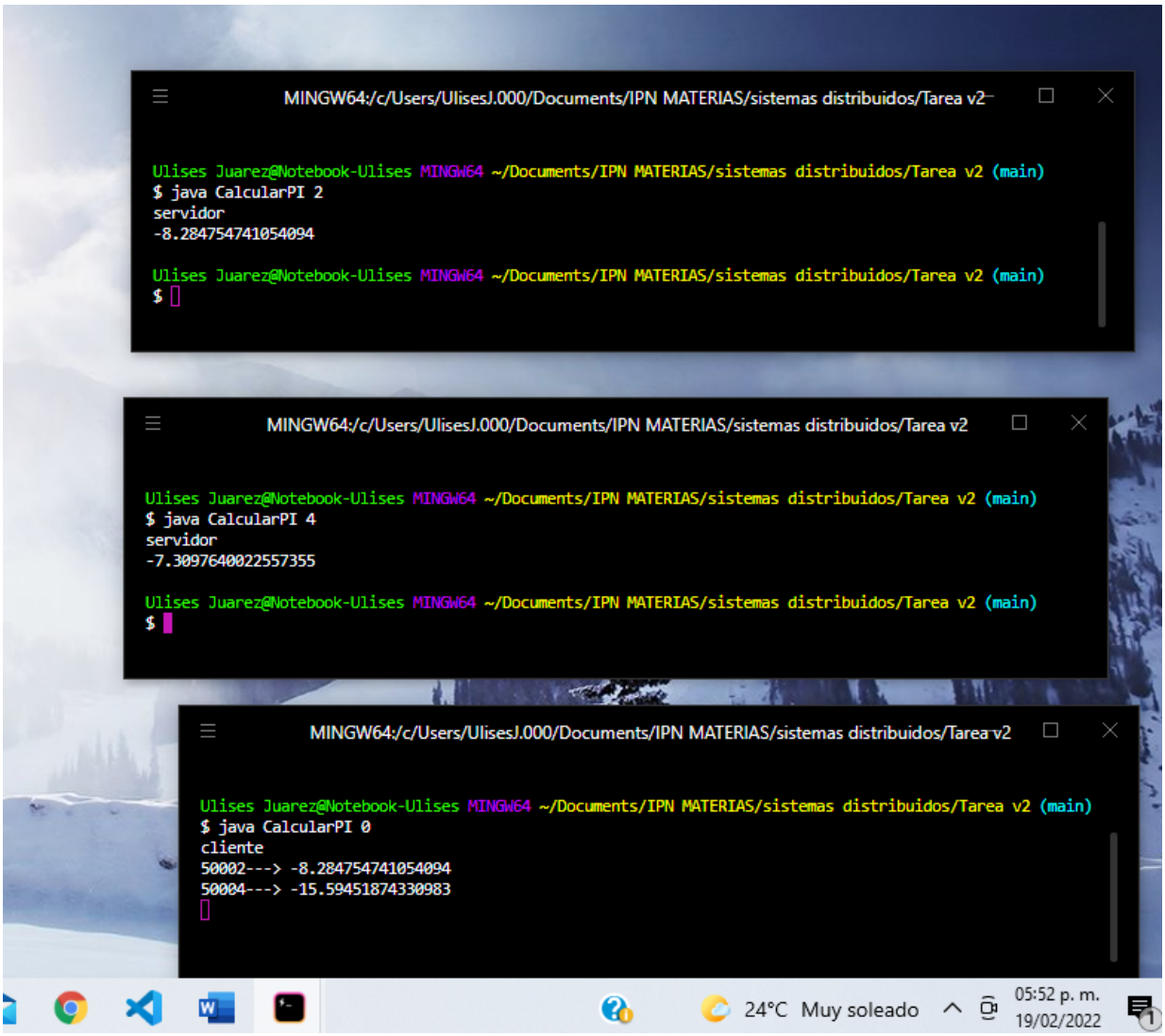

Ilustración 6 Ejecutando servidores al azar

Finalmente terminamos de ejecutar los servidores faltantes y observamos la aproximación a pi, impresa en el cliente.

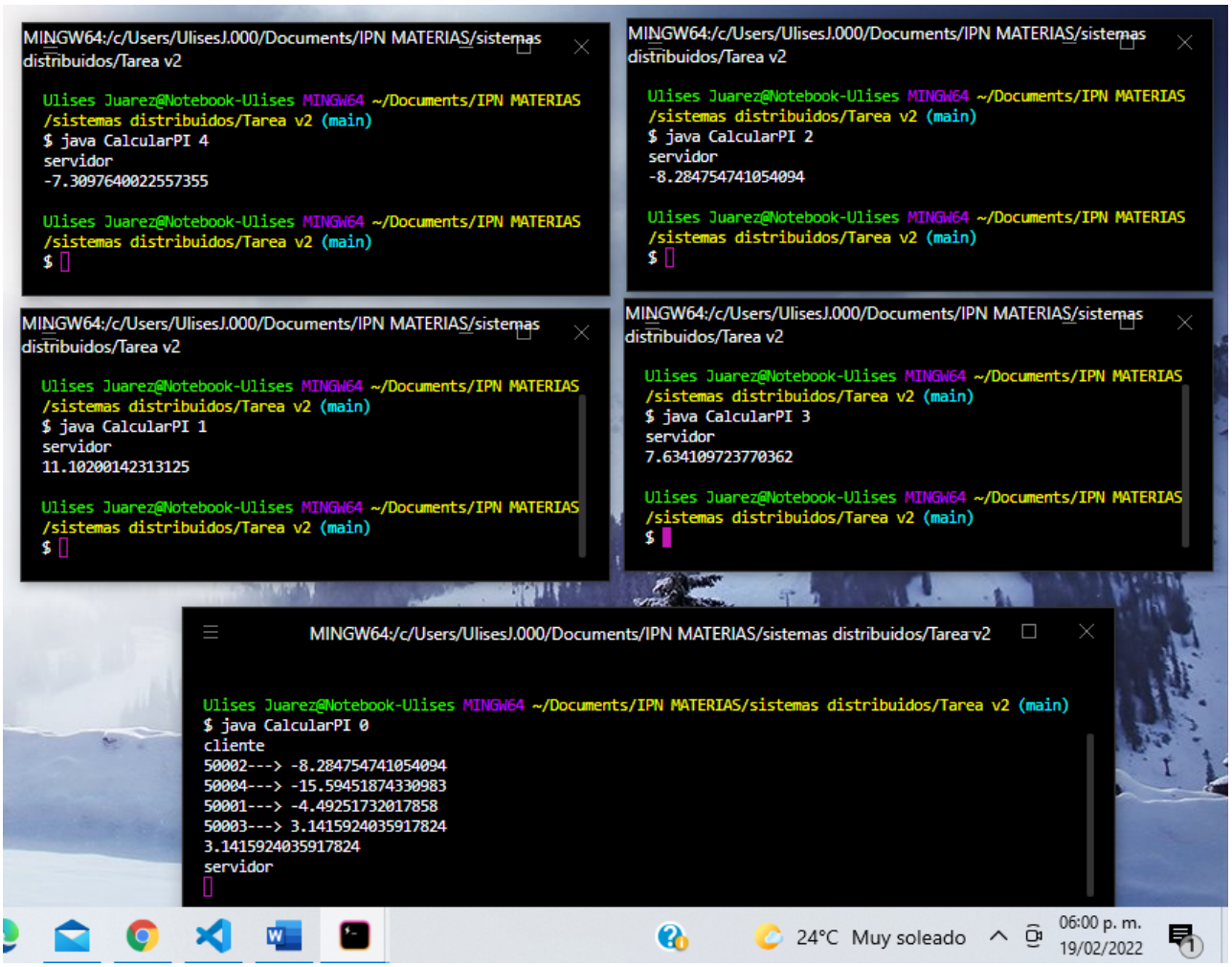

Ilustración 7 Resultado final

Conclusiones

Héctor Israel Jaime Villanueva:

Me pareció una práctica interesante al ver cómo se puede dividir una tarea en diferentes máquinas y como puede haber ciertos problemas si no se llegan a manejar bien los hilos. Hasta ahora supe realmente como es una conexión cliente-servidor y que tan sencillo o complejo puede llegar a ser un sistema implementándolo.

Ricardo Alberto Machorro Vences:

Esta tarea mostro como es que se puede usar la conexión para la comunicación entre clientes y servidores de tal forma que se pueda emular el comportamiento que se tienen las comunicaciones de la mayoría de las aplicaciones y servicios en la vida real con comportamientos como lo son el reintento de conexión por parte de los clientes a los servidores y el uso de sincronización de hilos para que para poder coordinar acciones entre elementos. Con esto ejercicio se puede ver de manera próxima como se puede dar la comunicación implementada en código en un lenguaje relativamente sencillo de

comprender, comparado con otros como C, pero con la suficiente complejidad para poder comprender hasta cierto nivel máquina saber que se está haciendo.

En resumen, esto lo considero como un ejercicio que muestra muy bien lo que se tiene que implementar para dar lo básico para el manejo de comunicaciones entre varios elementos que intercambian información y que por tal ejercicio significativo son la base de los sistemas distribuidos.

Juárez Espinoza Ulises:

Esta tarea más allá de enseñarme como se hace una aproximación al numero pi, mediante la serie Gregory-Leibniz. Me enseño la manera en la que un mismo programa puede actuar como servidor o como cliente. Previamente había entendido como funcionaba esta comunicación mediante los hilos y especificando el puerto en el socket tanto del lado del cliente, como del servidor.

Sinceramente el implementarlo en un solo programa mediante el uso de parámetros, en donde si el parámetro era un 0 al momento de correr el programa, esa máquina/consola actuaba como cliente y los parámetros del 1 al 4 actuaban como servidores.

Me pareció muy interesante el hecho de que cada uno de los nodos pudiera actuar sin problema gracias a que usaban un puerto diferente, asi mismo muy útil que el cliente tuviera intentos de reconexión ya que no es necesario que los servidores estuvieran activos al momento de que el cliente se conectará. Así mismo me ayudo a comprender como funcionan en su forma mas simple los sistemas distribuidos.