

**Práctica: 4***Múltiples clientes-múltiples servidores*

Alumno: Reyes Reyes Jayim Javier

Profesor: Chadwick Carreto Arellano

Materia: Sistemas Distribuidos

Grupo:7CM1

Fecha de entrega: 19/03/25

# Antecedente

En los sistemas distribuidos es necesario tener control y sincronización debido a que la complejidad aumenta cada vez que se soluciona un problema, entre ellos es la sincronización del tiempo, entre los componentes de la red.

Entre los algoritmos de sincronización tenemos el de reloj que parte de la base en que cada nodo de una red puede tener su propio reloj interno y estos relojes suelen derivarse con el tiempo por las diferencias de frecuencia en la oscilación de sus cristales de cuarzo. Para poder sincronizar el reloj es posible utilizar algoritmos de Berkeley, Cristian, Lamport o algoritmo de bullying.

Berkeley funciona con un nodo maestro que es el servidor el cual recoge los tiempos de los nodos clientes, en resumen, calcula la diferencia promedio y envía los ajustes a los clientes, de esta manera los clientes modifican sus relojes según las correcciones recibidas.

Cristian se resume a que un cliente solicita la hora a un servidor de tiempo, el servidor responde con su hora actual. El cliente ajusta su reloj, considerando la latencia estimada de la red.

Lamport, Usa marcas de tiempo lógicas en lugar de sincronizar relojes físicos. Cuando un proceso envía un mensaje, incrementa su contador lógico. El receptor actualiza su contador al valor más alto entre su reloj y el del mensaje recibido +1.

Bullying. Cuando un nodo detecta que el líder (nodo con el ID más alto) ha fallado, inicia una elección. Envía mensajes a nodos con mayor ID. Si no recibe respuesta, se autoproclama líder. Si recibe una respuesta, cede el liderazgo al nodo de mayor ID.

# Planteamiento del problema

Desarrollar en Java un ejemplo de utilizar varios servidores con un algoritmo de sincronización.

# Propuesta de solución

Varias conexiones de servidores para implementar el algoritmo de Berkeley y calcule los tiempos de ajuste, el server debe mandar a los servidores la sincronización y a los clientes el tiempo que demora y su ajuste.

# Materiales y métodos empleados

Lenguaje : Java

Librerías: import java.io.\*;

import java.net.\*;

import java.util.Random;

# Desarrollo de la solución

**Link del programa para descargarlo directamente en zip:**

**Explicación del código:**

**MasterServer.Java**

**Texto

El contenido generado por IA puede ser incorrecto.**

Definimos el puerto del servidor maestro, después creamos listas para almacenar las conexiones de clientes, servidores y tiempos.

**Texto

El contenido generado por IA puede ser incorrecto.**

Creamos la conexión con serversocket, y aceptamos 2 servidores y 3 clientes. Cada vez que un servidor o cliente se conecta, lo agregamos a su respectiva lista (servers o clients).

**Texto

El contenido generado por IA puede ser incorrecto.**

Leemos el tiempo enviado por cada servidor/cliente y lo almacenamos en times.

**Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.**

Se genera un tiempo aleatorio para el servidor maestro. Calculamos el tiempo promedio, incluyendo el tiempo del maestro. Calculamos la diferencia entre el tiempo promedio y el tiempo de cada nodo. Enviamos el ajuste a cada cliente y servidor para que sincronicen su reloj.

**SEVER2.JAVA**

**Texto

El contenido generado por IA puede ser incorrecto.**

Conectamos los servidores secundarios se conectan con el servidor maestro. Se genera un tiempo aleatorio y se envía al Servidor Maestro. Recibe el ajuste del maestro y actualiza su tiempo.

**CLIENTE.JAVA**

**Texto

El contenido generado por IA puede ser incorrecto.**

Genera un tiempo aleatorio, lo envía al servidor maestro, recibe el ajuste y actualiza su reloj.

# Resultados

En el servidor maestro recibe el tiempo de los servers y clientes y muestra los ajustes enviados.

Texto

El contenido generado por IA puede ser incorrecto.

Otro server.

Texto

El contenido generado por IA puede ser incorrecto.

Otro server.

Texto

El contenido generado por IA puede ser incorrecto.

Cliente 1

Texto

El contenido generado por IA puede ser incorrecto.

Cliente 2

Texto

El contenido generado por IA puede ser incorrecto.

Cliente 3

Texto

El contenido generado por IA puede ser incorrecto.

# Conclusiones

Los algoritmos de sincronización permite administrar y controlar mejor los tiempos de conexión, sincronización precisa, adaptabilidad, escalabilidad, sin embargo existen muchos contras que no se han solucionado, entre ello es la latencia que afecta la precisión y estabilidad de las conexiones y por otro lado la falla en los nodos pues un cliente se puede desconectar y esto afecta la sincronización. Definitivamente debe ser importante administrar las conexiones y utilizar estos algoritmos es un buen inicio para redes pequeñas.