La clase actualizarMiddle es un hilo que forma parte de nuestra clase middleware en un sistema de directorio distribuido. Su función principal es mantener actualizados los archivos en función de su tiempo de vida (TTL)

* Inicialización: El hilo actualizarMiddle se inicia con un objeto UDP como parámetro, lo que le permite comunicarse con nuestro servidor.
* Cálculo del TTL Máximo: En cada ciclo del bucle, se calcula el TTL máximo entre todos los archivos en la lista. Esto es importante porque nos aseguramos de que el tiempo no supere el TTL máximo para evitar errores causados por números demasiado altos.
* Incremento de Tiempo: El tiempo se incrementa en 1 milisegundo en cada iteración. Esto se hace para mantener un seguimiento preciso del tiempo y garantizar que se cumpla el TTL de los archivos correctamente.
* Verificación del Tiempo y TTL: Para cada archivo en la lista, se verifica si su TTL es diferente de cero. Si el TTL es mayor que cero, se verifica si el tiempo actual es un múltiplo exacto del TTL del archivo. Esto se hace verificando si el cociente entre el tiempo y el TTL es un número entero. Si es así, se envía un mensaje al servidor correspondiente para actualizar el archivo.
* Espera de 1 Milisegundo: Después de cada ciclo del bucle, se introduce una pausa de 1 milisegundo para controlar el incremento de tiempo y evitar un consumo excesivo de recursos.

package DNS;

import java.util.ArrayList;

class ActualizarMiddle extends Thread{

    private ArrayList<ArchivoGlobales> archivoGlobales = new ArrayList<>(); // un array donde guargamos cada archivo

    private UDP servidor;

    private int tiempo = 0; // variable para llevar el tiempo

    private int ttlMaximo = 0; // ttl maximo

    public ActualizarMiddle(UDP serv, ArrayList<ArchivoGlobales> archivos){

        servidor = serv;

        archivoGlobales = archivos;

    }

    @Override

public void run() {

    while (true) {

        ttlMaximo = encontrarTTLMaximo();

        // Incrementa el tiempo en 1 milisegundo en cada iteración

        tiempo = tiempo + 1;

        // Si el tiempo supera el TTL máximo, reinicia el tiempo para evitar errores por numeros altos

        if (tiempo > ttlMaximo + 1) {

            tiempo = 0;

        }

        // Quiero que cuente el tiempo

        for (ArchivoGlobales archivoGlobales2 : archivoGlobales) {

            if (archivoGlobales2.TTL != 0) {

                // Verifica si el cociente entre tiempo y TTL es un número entero ejmplo 1000/1000 = 1 1500/1000 = 1.5

                if (tiempo % archivoGlobales2.TTL == 0) {

                    servidor.enviarMensaje("mensaje", archivoGlobales2.IP, 5000);

                }

            }

        }

        try {

            Thread.sleep(1); // Espera 1 milisegundo antes de la siguiente iteración

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

    }

}

    public void actualizarArchivos(ArrayList<ArchivoGlobales> original){

        archivoGlobales = original;

    }

    private int encontrarTTLMaximo() {

        int maximo = 0;

        for (ArchivoGlobales archivo : archivoGlobales) {

            if (archivo.TTL > maximo) {

                maximo = archivo.TTL;

            }

        }

        return maximo;

    }

}

Vamos a establecer una función en la clase UDP que permitirá la conexión con nuestro Middleware para que este pueda manejar las solicitudes.

private DatagramSocket socket;

private Middleware middleware;

-----------------------------------------------

public void agregarMiddleware(Middleware mi){

        middleware = mi;

    }

Luego, en nuestro método principal, crearemos estos objetos:

UDP servidor = new UDP(); // creamos el objeto UDP

Middleware middleware;

----------------------------------------------------------------------------

servidor.start();

middleware = new Middleware(servidor);

servidor.agregarMiddleware(middleware);

Con esto, hemos configurado la conexión entre el servidor UDP y el Middleware para que puedan trabajar juntos de manera efectiva.