PL3	7	Armindo Rivero Vázquez Prieto	Saul Juan José
Nº PLo	Equipo	Apellidos	Nombre

44.436.879-M 71.737.084-S	uo278499@uniovi.es uo282978@uniovi.es
DNI	e-mail

1	Desarrollo de un inyector de carga	
Nº Práctica	Título	Calificación

Comentarios sobre la corrección

Asignatura de

CONFIGURACIÓN Y EVALUACIÓN DE SISTEMAS

Curso 2022-2023

Área de Arquitectura y Tecnología de Computadores
Departamento de Informática de la Universidad de Oviedo

1.Tabla de resultados

Prueba	Concepto	Valor esperado	Valor obtenido
1	Media del tiempo de reflexión	1	1.27630947
	Número de peticiones por hilo	10	10
2	Media del tiempo de reflexión	1	1.04442643
	Número de peticiones por hilo	100	100
3	Media del tiempo de reflexión	1	6.3381562
	Número de peticiones por hilo	10	10
4	Media del tiempo de reflexión	1	5.1228521
	Número de peticiones por hilo	100	100

2.Preguntas del anexo.

¿Todas las filas del archivo del registro contador de rendimiento son significativas? ¿Por qué?

No, ya que las primeras filas y las últimas, están midiendo el rendimiento cuando el inyector no se ha activado todavía y cuando se desactiva.

¿Cuál es la utilización promedio del procesador y explica cómo la has calculado?

Aproximadamente entre un 36.996% y un 37%, esto lo hemos calculado mediante el promedio de todos los valores de cada prueba.

¿Cuál es la utilización promedio de la memoria y explica cómo la has calculado?

Aproximadamente un 23.39991% contando que el PC en el que se ha realizado cuenta con 32GB de RAM. Calculado mediante la formula de: ((Memoria Total – Memoria Cache – Memoria Libre) / Memoria Total) * 100

Como el experimento (inyector y servidor) se ejecutan en la misma máquina no debe existir tráfico de red. ¿Cuál es el ancho de banda actual? ¿En qué unidades viene expresado? Suponiendo que el valor medio de contador Total de bytes / s fuera de 850000. ¿cuál sería la utilización de la red? Explica cómo la has calculado

Sería su totalidad medida en bits/segundo así que si calculamos la formula de:

$$\%\ Utilizacion = \frac{Total\ de\ bytes/s.}{Ancho\ de\ banda\ actual} x100$$

Nos da un resultado de 0.085% = (850000/1000000000)*100

3.Código Fuente

```
#include <windows.h>
#include <iostream>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <fstream>

using namespace std;

// DAR VALORES A ESTAS CONSTANTES
#define MAXPETICIONES 10
#define MAXUSUARIOS 50
#define PUERTO 57000
#define TAM_PET 1250
#define TAM_RES 1250

// INTRODUCIR VALORES DE FUNCIONAMIENTO
// BIEN AQUÍ O LEYENDOLOS COMO VALORES POR TECLADO

int numUsuarios;
int numPeticiones;
float tReflex;

// Estructura de almacenamiento

struct datos {
	int contPet;
	float reflex[MAXPETICIONES];
};

datos datoHilo[MAXUSUARIOS];


//-------------------------------------------------------------------------
float NumeroAleatorio(float limiteInferior, float limiteSuperior) {
	float num = (float)rand();
	num = num * (limiteSuperior - limiteInferior) / RAND_MAX;
	num += limiteInferior;
	return num;
}

//-------------------------------------------------------------------------
float DistribucionExponencial(float media) {
	float numAleatorio = NumeroAleatorio(0, 1);
	while (numAleatorio == 0 || numAleatorio == 1)
		numAleatorio = NumeroAleatorio(0, 1);
	return (-media) * logf(numAleatorio);
}

//-------------------------------------------------------------------------
// Funci�n para cargar la librer�a de sockets
int Ini_sockets(void) {
	WORD wVersionDeseada;
	WSAData wsaData;

	int error;

	wVersionDeseada = MAKEWORD(2, 0);
	if (error = WSAStartup(wVersionDeseada, &wsaData) != 0) {
		return error;
```

```
	}

	// Comprobar si la DLL soporta la versión 2.0

	if (LOBYTE(wsaData.wVersion) != 2 || HIBYTE(wsaData.wVersion) != 0) {
		error = 27;
		cerr << "La librería no soporta la versión 2.0" << endl;
		WSACleanup();
	}
	return error;
}

//--------------------------------------------------------------------
// Funci□n para descargar la librer□a de sockets
void Fin_sockets(void) {
	WSACleanup();
}

//-----------------------------------------------------

// Funcion preparada para ser un thread

DWORD WINAPI Usuario(LPVOID parametro) {

	DWORD dwResult = 0;
	int numHilo = *((int*)parametro);
	int i;
	float tiempo;

	//Variables para los sockets

	SOCKET elSocket;
	sockaddr_in dirServidor;
	char peticion[TAM_PET];
	char respuesta[TAM_RES];
	int valorRetorno;  // Para control de errores


	srand(113 + numHilo * 7);

	datoHilo[numHilo].contPet = 0;

	// ... Resto de cosas comunes para cada usuario

	for (i = 0; i < numPeticiones; i++) {
		// PRINTF solo para depuración NUNCA en medición BORRARLO
		//printf("SOLO DEPURACION - Peticion %d para el usuario %d\n", i, numHilo);

		// Hacer petición cuando se implementen los sockets
		// Implica:
		// 1 .- Creación del socket
		//
		elSocket = socket(AF_INET, SOCK_STREAM, 0);
		if (elSocket == INVALID_SOCKET) {
			//Control de posibles errores
			printf("Error %d INVALID_SOCKET", WSAGetLastError());
			exit(EXIT_FAILURE);
		}

		// 2 .- Conexión con el servidor
		//

		dirServidor.sin_family = AF_INET;
```

```
                dirServidor.sin_addr.s_addr = inet_addr("127.0.0.1");
                dirServidor.sin_port = htons(PUERTO + numHilo);
                valorRetorno = connect(elSocket, (struct sockaddr*) & dirServidor,
sizeof(dirServidor));
                if (valorRetorno == SOCKET_ERROR) {
                        printf("Error %d SOCKET_ERROR", WSAGetLastError());
                        exit(EXIT_FAILURE);
                }

                // 3 .- Enviar una cadena
                //
                valorRetorno = send(elSocket, peticion, sizeof(peticion), 0);
                if (valorRetorno == SOCKET_ERROR) {
                        //Control de posibles errores
                        printf("Error %d SOCKET_ERROR", WSAGetLastError());
                        exit(EXIT_FAILURE);
                }

                // 4 .- Recibir la respuesta
                //
                valorRetorno = recv(elSocket, respuesta, sizeof(respuesta), 0);
                if (valorRetorno != TAM_RES) {
                        //Control de posibles errores
                        printf("Error %d TAM_RES", WSAGetLastError());
                        exit(EXIT_FAILURE);
                }

                // 5 .- Cerrar la conexió
                //

                closesocket(elSocket);

                // Fin de la petición

                // Calcular el tiempo de reflexión antes de la siguiente petició
                tiempo = DistribucionExponencial((float)tReflex);

                // Guarda los valores de la petició
                datoHilo[numHilo].reflex[i] = tiempo;
                datoHilo[numHilo].contPet++;

                // Espera los milisegundos calculados previamente
                Sleep((unsigned int)tiempo / 1000);
        }
        return dwResult;
}


int main(int argc, char* argv[])
{
        int i, j;
        HANDLE handleThread[MAXUSUARIOS];
        int parametro[MAXUSUARIOS];

        // Leer por teclado los valores para realizar la prueba o asignarlos
        //POR HACER
        numUsuarios = atoi(argv[0]);
        tReflex = atoi(argv[1]);
        numPeticiones = atoi(argv[2]);

        // Inicializar los sockets UNA sola vez en el programa

        Ini_sockets();
```

```
	// Lanza los hilos

	for (i = 0; i < numUsuarios; i++) {
		parametro[i] = i;
		handleThread[i] = CreateThread(NULL, 0, Usuario, &parametro[i], 0, NULL);
		if (handleThread[i] == NULL) {
			cerr << "Error al lanzar el hilo" << endl;
			exit(EXIT_FAILURE);
		}
	}

	// Hacer que el Thread principal espere por sus hijos

	for (i = 0; i < numUsuarios; i++)
		WaitForSingleObject(handleThread[i], INFINITE);

	// Descargar la librería de sockets, aquí o donde se acabe
	// el programa

	Fin_sockets();

	// Recopilar resultados y mostrarlos a pantalla o
	// guardarlos en disco
	//POR HACER
	FILE* testfile;
	fopen_s(&testfile, "test.txt", "w");
	fprintf(testfile, "NumeroUsuarios NumeroPeticiones TiempoReflexion\n");
	for (i = 0; i < numUsuarios; i++) {
		for (j = 0; j < numPeticiones; j++) {
			fprintf(testfile, "%d ; %d ; %f\n", i, j, datoHilo[i].reflex[j]);
		}
	}
	fclose(testfile);

	return 0;
}
```