Proyecto 2
Evaluador de servidores con diferentes tipos procesos

1st Acevedo Rodríguez Kevin
Instituto Tecnológico de Costa Rica
Principios de Sistemas Operativos
Cartago, Costa Rica
kevinar51@estudiantec.cr

2nd Camacho Hernández José Julián
Instituto Tecnológico de Costa Rica
Principios de Sistemas Operativos
Cartago, Costa Rica
jcamacho341@estudiantec.cr

3rd Venegas Vega Jose Agustín
Instituto Tecnológico de Costa Rica
Principios de Sistemas Operativos
Cartago, Costa Rica
joseagustinveneveg@estudiantec.cr

4th Solís Arguello Juan Antonio
Instituto Tecnológico de Costa Rica
Principios de Sistemas Operativos
Cartago, Costa Rica
JuanSa@estudiantec.cr

***Resumen*—This document contains the whole documentation of a project that involves a client-server architecture with servers of different types of processes, such as a short introduction to explain important information to improve the comprehension of the different theoric topics, also contains the design details, usage instructions and develop activities per student, among others. This project has a lot of intersting common cases, corner cases and topics that will be explained later, syncronizers, process generation, server-client communication, graphics, between them.**

***Index Terms*—Comunicación, Memoria compartida, Memoria circular, Semáforos, Sincronizador.**

I. Introducción

En el presente documento explicaremos la creación de un sistema cliente-servidor, el cual le permite a un mismo cliente conectarse con cuatro servidores distintos, los cuales ejecutan las ordenes del cliente según cuatro procesos distintos, los cuales son: ejecución secuencial, ejecución mediante procesos pesados, ejecución mediante hilos y ejecución mediante procesos pesados predefinidos, respectivamente. Estos últimos serán explicados brevemente a continuación.

- **Ejecución Secuencial:** Las instrucciones se ejecutan una tras otra, en orden, de manera lineal y sin interrupciones.
- **Ejecución mediante procesos pesados:** Cada proceso tiene su propio espacio de memoria y se ejecuta de forma independiente.
- **Ejecución mediante hilos:** Varios hilos comparten el mismo espacio de memoria y se ejecutan de forma concurrente, pero cada hilo tiene su propia pila de llamadas.
- **Ejecución mediante procesos pesados predefinidos:** Implica crear varios procesos hijo antes de que se necesiten para manejar las solicitudes entrantes. Cuando llega una solicitud, se asigna a uno de los procesos hijos y este procesa la solicitud.

II. Ambiente de desarrollo

A continuación se detalla la configuración básica necesaria para la ejecución del proyecto. El mismo fue desarrollado en un ambiente Linux, por tanto fue posible la utilización del conjunto de herramientas base que ofrece dicha plataforma como GNU, *gcc*, entre otras bibliotecas básicas.

Adicionalmente, entre las herramientas, *frameworks*, bibliotecas y demás aplicaciones de desarrollo que fueron utilizadas para la implementación del proyecto se encuentran las siguientes:

- **PThreads:** Biblioteca POSIX Threads que permite el trabajo concurrente por medio del manejo de hilos en un proceso. Fue útil para realizar varias solicitudes desde el cliente y para recibirlas desde el servidor de hilos.
- **Semaphore:** Biblioteca que fue fundamental para la administración de los recursos compartidos por parte de los diferentes procesos pesados creados por los servidores de *heavy process* y *pre heavy process*.
- **Socket:** Biblioteca que proporciona funciones, estructuras de datos y constantes para crear y manipular *sockets*, que funcionan como puntos finales para la comunicación de red.
- **Unistd:** Biblioteca que proporciona una amplia gama de funciones y constantes que son fundamentales para la programación del sistema y la gestión de procesos, como el `fork()`, `getpid()`, entre otras.
- **Inet:** Biblioteca que define estructuras de datos y constantes para la familia de protocolos de Internet, incluidas las direcciones IP y los números de puerto. Se usa comúnmente para la programación de redes, especialmente con TCP/IP.
- **Types:** esta biblioteca define varios tipos de datos utilizados en las llamadas al sistema y otras interfaces del

sistema. Incluye tipos como `pid_t` (ID de proceso), `uid_t` (ID de usuario) y `gid_t` (ID de grupo).

- **Stat:** Biblioteca incluye funciones y estructuras de datos para trabajar con el estado de los archivos y los atributos del sistema de archivos, como verificar los permisos de los archivos, obtener el tamaño de los archivos y modificar los metadatos de los archivos.
- **GitHub:** Plataforma que contiene el repositorio de la tarea. Fue de gran utilidad para el manejo de versiones, la sincronización y el acceso del código actualizado para los miembros del equipo.
- **Visual Studio Code:** Editor de texto que fue útil en el manejo y programación de los diferentes archivos de código en C, *makefile*, entre otros.

III. Atributos

III-A. Aprendizaje continuo

III-A1. Necesidades actuales de aprendizaje para enfrentar: Las necesidades actuales de aprendizaje de este proyecto son principalmente relacionadas con la implementación de arquitecturas cliente-servidor en C. Se debe aprender a realizarlas utilizando diferentes tipos de procesos como hilos y procesos pesados.

También se debe pasar una curva de aprendizaje respecto a la obtención de estadísticas como tiempos de ejecución, uso de memoria y de CPU de los diferentes procesos.

III-A2. Tecnologías que se pueden utilizar para el desarrollo: Para el desarrollo se pueden utilizar las diferentes tecnologías para el manejo de los diferentes tipos de procesos como por ejemplo las bibliotecas pthreads, socket, unistd. También se pueden utilizar herramientas de inteligencia artificial como *ChatGPT* que puede ser de gran utilidad en la resolución de dudas y problemas presentes durante el desarrollo del proyecto.

III-A3. Acciones implementadas para el desarrollo del proyecto: El grupo tuvo una reunión inicial para la lectura de la especificación del proyecto y división de tareas de investigación y búsqueda de información relacionada que podrían facilitar el desarrollo del proyecto.

Luego de esto, nos reunimos para realizar el desarrollo del cliente con hilos, y división del trabajo de los diferentes tipos de servidores. Paulatinamente, se fue desarrollando el proyecto en grupo e investigando de manera grupal, tríos, parejas, o de forma individual los problemas que se fueron presentando.

III-A4. Evaluación de forma crítica de la eficiencia de las acciones implementadas en el contexto tecnológico: Desde un punto crítico, se cumplen con todos los objetivos diseñados para la elaboración del proyecto. Desde nuestro punto de vista, lo hace de una manera eficiente y confiable, se utilizan múltiples herramientas que nos permitieron desarrollar código de manera eficiente y segura también trabajamos de manera ordena y equipo por lo tanto el desarrollo del proyecto fue un éxito.

III-B. Herramientas de ingeniería

III-B1. Herramientas, bibliotecas o recursos que se utilizaron en el proyecto: Se utilizan diferentes tecnologías, herramientas, y bibliotecas para el desarrollo del proyecto entre las más destacadas son la siguientes: PThreads, Semaphore, Socket, Unistd, Inet, Types, Stat, Github, GitKraken, Visual Studio Code entre otras herramientas que han sido mencionadas en el presente paper que nos permitieron desarrollar el proyecto de una manera más eficiente.

III-B2. Manera en que se aplicaron los recursos o bibliotecas seleccionados: Tanto la biblioteca Semaphore se utilizó para el control de los diferentes semáforos utilizados en el desarrollo del proyecto , la biblioteca time se utilizó para la obtención de la hora en la cual fue introducción o leído un carácter, tanto github como gti kraken nos permitieron el alojamiento y la gestión del código y visual studio code como editor de texto con todas sus extensiones que nos permiten un desarrollo más cómodo del código.

III-B3. Manera en que se adaptaron los recursos para desarrollar el proyecto: Las bibliotecas utilizadas en el proyecto se aprovecharon y adaptaron para lograr desarrollar la funcionalidad detallada en la especificación del proyecto. Por ejemplo, la biblioteca pthreads se utiliza para controlar diferentes hilos dentro del cliente y servidor de hilos. GitHub fue útil para colaborar en equipo y mantener un historial de cambios en el código fuente y con todas estas adaptaciones y el lenguaje de desarrollo C, se logró desarrollar de manera efectiva el proyecto. Otros usos y adaptaciones de detallaron en la sección II.

IV. Detalles del diseño

A continuación, se presenta una serie de diagramas ayudan a describir el funcionamiento general del sistema.

IV-A. Diagrama de Funcionalidad

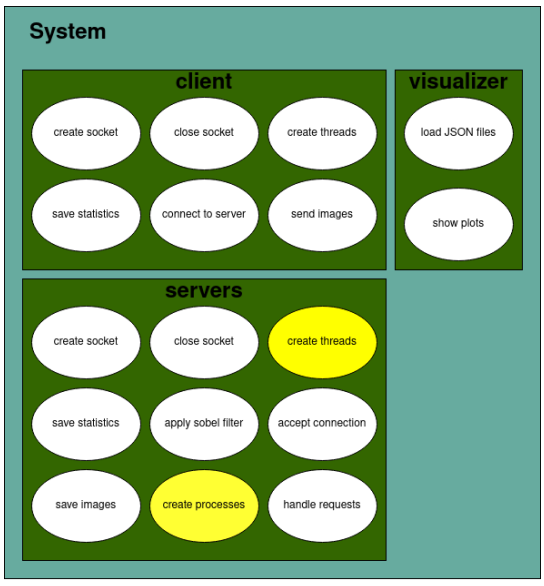

Figura 1. Diagrama de funcionalidad

En el fig 1 podemos observar el diagrama de funcionalidades presentes en el desarrollo de este proyecto, en este se puede observar las funcionalidades más importantes de las tres partes del sistema (cliente, servidores, y el visualizador). Es importante recalcar que en las funcionalidades de servidores se muestran unas formas en blanco y otras en amarillo, las formas que aparecen en blanco con funcionalidades comunes para los 4 tipos de servidores y las amarillas son específicas de un tipo de servidor.

IV-B. Diagrama de Arquitectura

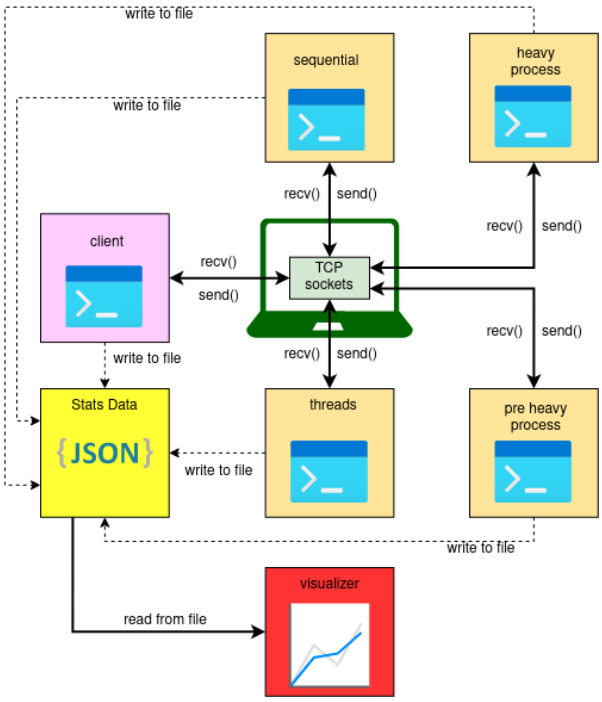

Figura 2. Diagrama de arquitectura de la solución planteada.

A como se puede ver en la figura 2, el diagrama de arquitectura muestra de qué manera se comuncican los servidores y el cliente (TCP socket), los tipos de comunicación (send y recv), los tipos de datos que generan de salida (Stats Data), y finalmente, el papel que juega el módulo de visualizar (Este lee los datos a partir de los archivos generados por la interacción cliente-servidor).

IV-C. Diagrama de Componentes

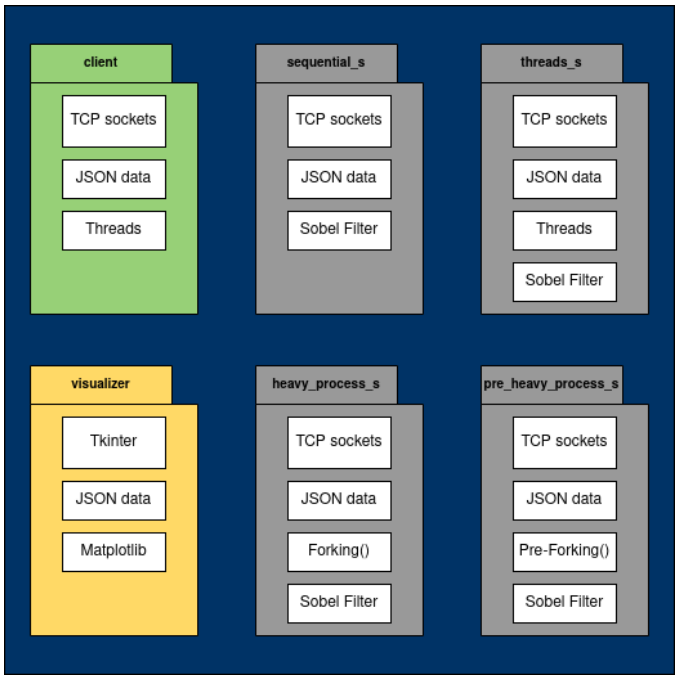

Figura 3. Diagrama de componentes de la solución planteada.

Tal y como se observa en la figura 3, los componentes principales del sistema se dividen en componentes de cliente, servidores, y visualizador.

El módulo de cliente cuenta con un componente de TCP socket que se usa para establecer una conexión con cualquier servidor, un manejador de datos JSON, y además, un componente de Threads para distribuir las solicitudes.

Los módulos de servidores implementan varios componentes en común (Para los 4 tipos de servidores), estos son: un componente de TCP socket, un manejador de datos JSON, un componente que se encarga de modificar las imágenes con Sobel Filter. Además, algunos de ellos contienen componentes específicos del tipo de servidor como lo son Forking, Pre-Forking, y Threads.

Finalmente, se tiene un módulo llamado visualizer (Esta es la parte gráfica del sistema), el cual cuenta con un componente de creación de ventanas (Tkinter), un manejado de datos JSON (para leer los datos de entrada), y también, un componente de graficación de datos (Matplotlib).

IV-D. Diagramas de secuencia

Se presentan dos diagramas de secuencia para poder describir de manera general la interacción principal entre cliente-servidor.

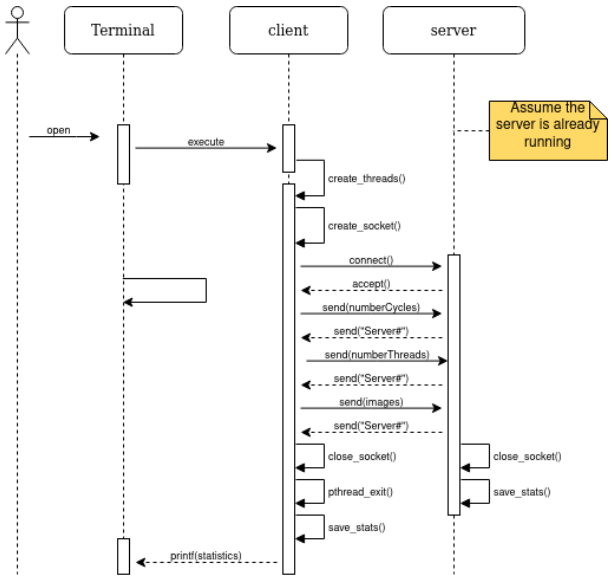

Figura 4. Diagrama de secuencia para describir al emisor.

En la figura 4 se describe de manera general la ejecución del cliente por medio de los siguientes mensajes entre objetos:

1. El usuario abre una terminal para ejecutar el programa del cliente.
2. El cliente crea N Threads.
3. El cliente crea los sockets para comunicarse con los servidores (Se crea 1 socket por cada Thread creado).
4. El cliente intenta conectarse a un servidor (Se asume que en este punto el servidor ya está escuchando en un puerto definido).
5. El servidor responde a las conexiones entrantes del cliente (por medio de un accept).
6. El cliente envía la cantidad de Ciclos.
7. El servidor confirma que ha recibido el mensaje.
8. El cliente envía la cantidad de Threads creados.
9. El servidor confirma que ha recibido el mensaje.
10. El cliente envía las imágenes respectivas con cada uno de los Threads creados.
11. El servidor confirma que ha recibido las imágenes.
12. El cliente y el servidor cierran las conexiones de socket.
13. El servidor confirma que ha recibido el mensaje.
14. El servidor guarda los datos medidos para las estadísticas (Las del lado del servidor).
15. El cliente cierra los Threads creados.
16. El cliente guarda los datos medidos para las estadísticas (Las del lado del cliente).
17. El cliente envía mensajes en la terminal del usuario para mostrar las estadísticas medidas.

V. Tabla de actividades por cada estudiante

A continuación se presenta la distribución de las tareas identificadas en la tarea por estudiante, así como datos como su fecha límite establecida, horas dedicadas y estado de completitud.

Descripción	**Responsable(s)**	**Horas de trabajo**	**Estado**
Generar un sistema servidor sencillo que procese imágenes y se comunique.	Julián y Juan	**3**	Completado
Generar un sistema cliente sencillo que envíe imágenes mediante comunicación	Julián y Kevin	**3**	Completado
Generar primera versión del servidor secuencial	Julián y Juan	**2**	Completado
Generar primera versión del servidor de procesos pesados	Kevin	**3**	Completado
Generar primera versión del servidor de hilos	Agustín	**2**	Completado
Generar primera versión del servidor de procesos pesados predefinidos	Kevin y Agustín	**3**	Completado
Desarrollo de interfaz con opciones de visualización	Juan	**4**	Completado
Desarrollo de algoritmo de filtro para imágenes	Julián	**3**	Completado
Reunión explicativa para medir avance y evaluación de requerimientos faltantes	Todos	**3**	Completado
Generación de archivos que almacenen datos estadísticos en una estructura de datos	Agustín	**4**	Completado
Cobertura de casos esquina	Kevin y Agustín	**3**	Completado
Recolección de datos estadísticos	Kevin	**2**	Completado
Adaptación de interfaz a archivos generados	Juan	**2**	Completado
Integración filtro con servidores	Julián	**2**	Completado
Revisión detallada de generación de datos y gráficas	Juan y Agustín	**1**	Completado
Integración completa mediante estructuras de comandos en makefile y corrección de errores	Todos	**3**	Completado

Cuadro I
Tabla de actividades por estudiante

VI. Conclusiones

En el proceso de realización del trabajo fue posible comprobar la importancia de las diversas herramientas de trabajo y tecnologías que están a disposición para el implementación de este tipo de proyectos, ya que apoyan y facilitan las labores de los desarrolladores.

Adicionalmente, se reforzaron conceptos relacionados con el análisis de requerimientos, diseño de sistemas, y desarrollo de programas en lenguaje C.

De igual manera, se reforzó el tema visto en el curso sobre procesos pesados e hilos. Se logró corroborar que los hilos comparten espacios de memoria del procesos padre, mientras que los procesos pesados necesitan algún tipo de memoria compartida. Al analizar las diferentes estadísticas que se obtuvieron a partir de la ejecución de los tipos de servidores, fue posible comprobar que realizar trabajos concurrentes, ya sea mediante hilos o procesos pesados pre-creados, mejora los

tiempos de procesamiento de datos en una arquitectura como la de cliente-servidor.

Finalmente, durante el proyecto fue posible observar el comportamiento de los diferentes procesadores según las estadísticas plantedas, observando como cada uno de los tipos de servidores tiene sus propias ventajas y desventajas según la naturaleza del servidor. Por ejemplo, la diferencia de velocidad entre un procesado secuencial y un procesador bajo el uso de hilos, o la cantidad de memoria utilizada por un procesador con la naturaleza de proceso pesado contra uno secuencial.

VII. Sugerencias y recomendaciones

El equipo de trabajo reflexionó sobre todo lo realizado, y se considera que en base a nuestras experiencias, podríamos aportar cierto conjunto de sugerencias y recomendaciones, las cuales serían muy útiles a futuro, para quien interese y desee entender y/o desarrollar un proyecto como el realizado:

- Si se trabajará en equipo, se recomienda mantener actividades, reglas y medidas que contribuyan a la compenetración del equipo, al desarrollo constante y al interés por el trabajo individual de cada uno, esto contribuirá al entendimiento general de la aplicación y su desarrollo.
- Para este tipo de proyectos donde se deben manejar clientes con hilos y servidores de varios tipos de procesos, se recomienda en gran medida realizar un diseño previo del trabajo. Esto con el objetivo de ubicarse con lo que se realizará en lugar de codificar sin una estructura definida.
- Respecto a sugerencias funcionales, para la implementación de arquitecturas cliente-servidor en C se recomienda utilizar bibliotecas como `socket`. Es importante tener claro el protocolo y realizar un buen manejo de errores en las conexiones para la depuración.
- Con el fin de evitar el *busy waiting*, se sugiere colocar un tiempo de *timeout* para que los servidores no esperen por conexiones indeterminadamente.
- En la utilización de los hilos, se sugiere el uso de bibliotecas como `pthread`. En una arquitectura cliente-servidor, se recomienda implementar una función que maneje las conexiones con la otra parte, así como esperar a que todos los hilos terminen.
- Finalmente, respecto a los procesos pesados con `fork()` se recomienda utilizar memoria compartida, y por ende semáforos, para obtener estadísticas totales de cada uno de los procesos creados, ya que son procesos con su propio espacio de memoria.

VIII. Referencias:

[1] Tanenbaum. Operating systems: design and implementation. Prentice Hall. 1988

[2] Peterson. Operating Systems Concepts. Addison Wesley, Second Edition. 1985

[3] Lubomir y Shaw, The logical design of operating systems. Prentice Hall, Second Edition. 1988.

[4] Kamburugamuve, S., Wickramasinghe, P., Govindarajan, K., Uyar, A., Gunduz, G., Abeykoon, V., Fox, G. (2018, July). Twister: Netcommunication library for big data processing in hpc and cloud environments. In 2018 IEEE 11th International Conference on Cloud Computing (CLOUD) (pp. 383-391). IEEE.

[5] Liguori, A. N., Wilson, M. S., Nowland, I. P. (2018). U.S. Patent No. 9,886,297. Washington, DC: U.S. Patent and Trademark Office.