



Nombre del curso: Programación Paralela y Distribuida

Avance del Proyecto 1: Simulador de Control de Autobuses:

San José- Golfito- Paso Canoas-San José

Realizado por:

- Mariela Montero Mata
- Jose Daniel Sanchez Perez
- Andrés Sotela Gutierrez
- Samuel Vieira do Salao

Profesor: Jorge Bastos

IIC 2025

18/06/2025

Índice

Descripción del proyecto	3
Funcionalidad de la aplicación	3
Estrategia de solución	4
Representación de cada autobús	6
Mapa	10
Diseños de Estructura	10
Manual Técnico	10
Objetivo del programa	12
Alcance	12
Público Objetivo	12
Requisitos del sistema	12
Hardware	12
Software	12
Conceptos	13
Hilos	13
UDP	13
TCP	14
Referencias	15

Descripción del proyecto

Este proyecto se enfoca en el desarrollo de un simulador de Control de Autobuses que modela el desplazamiento de 10 unidades de transporte público a lo largo de una ruta predefinida con 20 paradas claves. La aplicación se construirá utilizando Java, haciendo uso intensivo de conceptos de concurrencia y paralelismo para simular de manera realista el movimiento independiente y simultáneo de cada autobús.

El sistema operará bajo una arquitectura cliente-servidor. El servidor será el cerebro central que gestiona el estado de todos los autobuses, sus posiciones y el progreso a lo largo de la ruta, mientras que una o varias interfaces cliente, permiten la visualización y el monitoreo en tiempo real de la simulación.

La aplicación busca emular la logística de una flota de autobuses en una ruta, específica, poniendo a prueba la implementación de técnicas de programación concurrente y paralela para lograr una simulación robusta, eficiente y visualmente representativa.

Funcionalidad de la aplicación

El objetivo principal de esta aplicación es simular y monitorear el movimiento de 10 autobuses a lo largo de una ruta con 20 paradas, donde cada autobús se mueva de una parada a otra con velocidades variables, reflejando las condiciones reales de viaje. utilizando un modelo cliente-servidor.

La simulación se divide en cuatro funciones principales:

1. La simulación del movimiento y progreso de los autobuses

- El tiempo entre paradas es variable.
- Cada autobús es un hilo independiente que avanza por las 20 paradas.
- Cada autobús inicia la ruta al mismo tiempo.

2. Funcionalidad del servidor

- El servidor registra la ubicación actual de cada autobús.
- También controla el tiempo y actualiza la posición de cada autobús.

3. Funcionalidad del cliente

- El cliente recibe actualizaciones sobre las paradas de los autobuses y hora de llegada.
- El cliente puede monitorear los autobuses.

4. Interfaz

- Representaciones visuales de los autobuses con iconos de diferentes colores.
- Interfaz tiene un mapa de las 20 paradas.

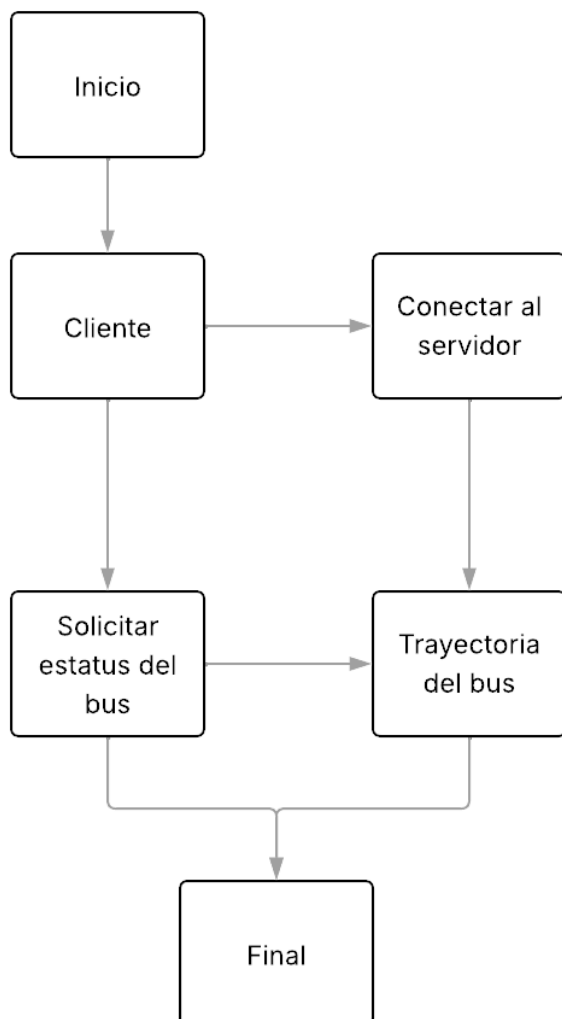
Estrategia de solución

Nuestra estrategia girará en torno a una arquitectura cliente-servidor en Java, enfocándonos en la concurrencia y el paralelismo robustos para manejar nuestros 10 autobuses.





- Tecnologías y Principios Clave:
 - Librerías estándar de Java: Utilidades de concurrencia (`java.util.concurrent`).
 - Diseño Orientado a Objetos: Autobús, Parada y Ruta.
 - Concurrencia por Diseño: Cada autobús tiene su propio hilo de ejecución.
 - Gestión de Estado Compartido: El servidor será el centro para el estado compartido.
- Pasos de Desarrollo:

- Definir Modelos Centrales
- Definición de Ruta
- Hilos básicos para el autobús
- Framework del servidor
- Implementación de Concurrencia
- Comunicación Básica
- Lógica de Velocidad Variable
- Refinamiento y Pruebas

Diagrama de flujo



Representación de cada autobús

Autobús 1	
Autobús 2	
Autobús 3	
Autobús 4	

Autobús 5



Autobús 6





Autobús 7



Autobús 8



Autobús 9	
Autobús 10	

Paradas de la ruta

1	Terminal de transporte
2	Terminal municipal ciudad Nelly
3	Sucursal Dos Pinos, Río Claro
4	Kilometro 30
5	Parada Río Esquinas

6	Olla Cero
7	Ojo de Agua
8	Tica Bus Uvita
9	Guapil
10	Portalón
11	Llamarón
12	Pocares
13	Carretera Pacífica Fernández Oreamuno
14	Pochotal
15	Carretera Pacífica Fernández Oreamuno #2
16	Soda el Higuero
17	Escobal
18	Autopista José María Castro Madriz
19	Barrio Los Ángeles
20	Terminal Tica Bus San José

Mapa

El presente mapa detalla la trayectoria estimada de los diez autobuses programados para salir desde la ciudad capital, San José, con rumbo hacia el sur del país. A lo largo del recorrido se incluyen sus respectivas paradas estratégicas, permitiendo observar con claridad los puntos de ascenso y descenso de pasajeros. El trayecto continúa desde San José hasta la ciudad de Golfito, y finalmente culmina en Paso Canoas, frontera con Panamá. Este recorrido no solo ofrece una visión logística de la ruta, sino que también facilita la planificación de viajes y la identificación de los servicios de transporte disponibles a lo largo del itinerario.



Diseños de Estructura

Manual Técnico



Nombre del curso: Programación Paralela y Distribuida

Manual Técnico del Proyecto

- Avance 2

Realizado por:

- Mariela Montero Mata
- Jose Daniel Sanchez Perez
- Andrés Sotela Gutierrez
- Samuel Vieira do Salao

Profesor: Jorge Bastos

IIC 2025

16/07/2025

Objetivo del programa

Desarrollar una aplicación en Java (Apache NetBeans) que simule un sistema de control de autobuses que recorren una ruta específica con 20 paradas. El objetivo es mostrar el estado en tiempo real de cada autobús mediante comunicación cliente-servidor, control de hilos, concurrencia y actualización dinámica de la posición de los autobuses.

Alcance

La aplicación simula 10 autobuses circulando por una ruta predeterminada, donde cada bus avanza de forma independiente. Un cliente puede seleccionar un autobús y recibir información en tiempo real de su ubicación. Se incluyen elementos gráficos para representar buses y paradas. En esta fase del avance se entrega la interfaz gráfica y la estructura parcial del sistema.

Público Objetivo

Estudiantes, docentes del curso, y desarrolladores interesados en programación concurrente y simulaciones de sistemas distribuidos usando Java.

Requisitos del sistema

Hardware

- Procesador: Intel Core i5 o equivalente
- Memoria RAM: 8 GB
- Espacio en disco: 500 MB libres
- Resolución mínima: 1366x768

Software

- Sistema Operativo: Windows 10, Linux o macOS
- Java JDK 17 o superior
- Apache NetBeans 17 o superior
- Librerías estándar de Java SE (Swing, java.net, java.util.concurrent)

Conceptos

Hilos

¿Cómo se crean y gestionan los hilos?

En Java, los hilos se pueden crear mediante la clase `Thread` o implementando la interfaz `Runnable`. Estos mecanismos permiten ejecutar múltiples tareas en paralelo. En este proyecto, cada autobús es un hilo independiente que se mueve entre paradas. (Thread (java platform SE 8), 2025)

¿Qué tareas realiza cada hilo?

Cada hilo representa un autobús que recorre las 20 paradas asignadas. La simulación incluye pausas aleatorias entre paradas para reflejar diferencias de velocidad o retrasos.

¿Cómo se sincronizan los hilos?

Se aplican técnicas de sincronización para evitar colisiones o superposición de autobuses en las paradas. Se pueden usar bloques `synchronized`. (Javathreading, s/f)

UDP

¿Para qué se utiliza UDP?

UDP (User Datagram Protocol) es un protocolo de transporte sin conexión utilizado para enviar mensajes de forma rápida y con menor sobrecarga que TCP. A diferencia de TCP, UDP no garantiza la entrega, el orden ni la integridad de los datos, pero es útil en aplicaciones donde la velocidad es más importante que la confiabilidad.

En este proyecto, UDP podría usarse como alternativa a TCP para enviar actualizaciones frecuentes sobre la posición de los autobuses, donde pequeños errores en la transmisión no interrumpen el funcionamiento general del sistema.

UDP resulta ideal para aplicaciones como esta simulación porque:

- No necesita establecer una conexión formal antes de enviar datos.
- Reduce la latencia al eliminar confirmaciones y mecanismos de control de flujo.
- Permite una comunicación más fluida y rápida para eventos que ocurren frecuentemente (como el paso del bus por cada parada).

¿Cómo se establecen las conexiones UDP?

Aunque técnicamente no se “establece” una conexión en UDP, ambos extremos (cliente y servidor) deben crear un socket (`DatagramSocket`) y conocer las direcciones IP y puertos de destino.

1. El servidor crea un `DatagramSocket` y permanece en espera de paquetes entrantes.

2. El cliente crea su propio `DatagramSocket` y utiliza un `DatagramPacket` para enviar mensajes al servidor con información como el ID del bus o solicitud de posición.

¿Cómo se envían y reciben datos por UDP?

- Para enviar un mensaje, se crea un objeto `DatagramPacket` que contiene los datos (por ejemplo, una cadena con la posición del autobús), la IP del destinatario y el puerto. Este paquete se envía a través del socket con `send()`.
- Para recibir un mensaje, el socket escucha paquetes entrantes usando `receive()`, y luego se extraen los datos del `DatagramPacket`.

(*The User Datagram Protocol (UDP)*, s/f)

TCP

¿Para qué se utiliza TCP?

TCP (Transmission Control Protocol) es un protocolo de transporte orientado a la conexión que garantiza la entrega ordenada, confiable y libre de errores de los datos entre aplicaciones. TCP divide la información en segmentos y asegura su correcta entrega mediante confirmaciones, números de secuencia y mecanismos de retransmisión.

En este proyecto, TCP se utiliza para la comunicación entre el servidor (que controla los autobuses) y el cliente (que monitorea la ubicación de un bus específico). Dado que es esencial mantener la precisión en la transmisión de la posición de los autobuses, TCP es ideal para garantizar que los datos lleguen de forma completa y en orden.

¿Cómo se establecen las conexiones TCP?

1. El **servidor** crea un `ServerSocket`, especificando un puerto, y queda a la espera de conexiones entrantes con el método `accept()`.
2. El **cliente** crea un `Socket` para conectarse al servidor utilizando la dirección IP y el puerto correspondiente.
3. Una vez establecida la conexión, ambas partes pueden intercambiar datos de manera bidireccional, como si usaran un canal virtual privado.

Este proceso forma parte de lo que TCP denomina el *three-way handshake*, una secuencia de inicio que garantiza que ambas partes estén listas para comunicarse.

¿Cómo se envían y reciben datos por TCP?

Una vez conectados, los datos se envían mediante flujos:

- **El servidor y el cliente** usan `InputStream` para leer datos entrantes y `OutputStream` para escribir datos salientes.
- Los mensajes pueden incluir información como: ID del autobús, hora de llegada a la parada y número de parada.

TCP asegura que:

- Si un segmento se pierde, se retransmite.
- Si los datos llegan en desorden, TCP los reorganiza.
- Si el destino no está disponible, el emisor es notificado

(AIX, 2024b)

Arquitectura del programa

Diagrama de clase

Descripcion de modulos

Flujo de datos

Guia

Referencia del APP

Rendimiento

Depuracion y solucion de problemas

Referencias

AIX. (2024, agosto 27). Ibm.com.

<https://www.ibm.com/docs/es/aix/7.1.0?topic=protocols-user-datagram-protocol>

The User Datagram Protocol (UDP). (s/f). Abdn.ac.uk. Recuperado el 16 de julio de 2025, de <https://erg.abdn.ac.uk/users/gorry/course/inet-pages/udp.html>

Thread (java platform SE 8). (2025, julio 15). Oracle.com.

<https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.html>

AIX. (2024b, agosto 27). Ibm.com.

[https://www.ibm.com/docs/es/aix/7.1.0?topic=protocols-transmission-control-prot
ocol](https://www.ibm.com/docs/es/aix/7.1.0?topic=protocols-transmission-control-protocol)