CSC-300L Operating Systems

Project Proposal

Section: Software Engineering

Smart City

Submitted by:

Abdullah Feroz	2023-SE-03
Umer Awais	2023-SE-17
Fiza Niazi	2023-SE-23

Submitted to:

Ms. Maryam Manzoor

Dated: 8th December, 2025

Department of Computer Science

University of Engineering and Technology Lahore, New Campus

Smart City - Traffic Control System

Objective

To design an automated Traffic Control System that simulates concurrent resource management. The goal is to maximize traffic flow (Throughput) and guarantee safety (Deadlock Prevention) using Operating System principles.

Introduction

This project simulates a high-stakes 4-way intersection managed by an automated "Traffic Controller" process. The system coordinates incoming traffic (cars) from North, South, East, and West to prevent accidents. It features a special "Emergency Mode" where ambulances are given immediate priority, and a "Smart Mode" where signal timings adapt to traffic density.

Features

- **Traffic Generation**
  The system automatically generates vehicles arriving at random intervals to simulate unpredictable traffic loads.

- **Priority Handling**
  Vehicles are flagged as "Regular" or "Emergency." Emergency vehicles (Ambulances) preempt regular traffic immediately.

- **Process Aging (Starvation Prevention)**
  To ensure fairness, "Regular" cars that wait in the queue beyond a specific time threshold have their priority boosted, preventing them from waiting indefinitely.

- **Deadlock Management**
  The system implements a strict safety algorithm to monitor resource allocation, ensuring that a circular wait (gridlock) between North, South, East, and West never occurs.

- **Dynamic Signal Timing**
  A monitoring system detects lane density; if a lane is heavily congested, the Green Light duration extends automatically to clear the backlog.

- **Performance Tracking:**
  A real-time dashboard displays system metrics, including Average Wait Time and Throughput, to visualize efficiency.

Concepts Used in Project (OS Concepts)

- **Process Creation:**
  Each vehicle is a unique process created using `fork()`. The Traffic Controller acts as the parent process managing these children.

- **Synchronization & Deadlock Management**
  The intersection is a Critical Section. We use **Mutexes** to enforce Mutual Exclusion. We also implement a **Deadlock Prevention** strategy by strictly ordering resource requests (traffic flows) to prevent circular wait conditions.

- **CPU Scheduling (Priority + Aging)**
  The Controller acts as the scheduler. It uses **Priority Scheduling** to handle Ambulances and implements **Aging** to gradually increase the priority of waiting processes, preventing starvation.

- **IPC**
  Vehicles communicate their arrival time, direction, and type to the Controller using **Message Queues**.

- **Threads**
  A separate **Pthread** acts as a "Sensor," running concurrently to monitor queue sizes and update global timer variables without blocking the main process.

- **Memory Management**
  We use **Dynamic Memory Allocation** (Linked Lists) to create and manage the "Ready Queues" for each lane, simulating the OS ready queue structure.

Development Timeline

- **Phase 1 (Setup)**
  Define struct for vehicles, implement Linked List logic, and set up Message Queues (IPC).

- **Phase 2 (Core Logic)**
  Implement the Traffic Generator and the Priority Scheduler (Ambulance logic).

- **Phase 3 (Sync & Safe)**
  Add Mutex locks for the intersection and implement Starvation Prevention (Aging).