COLLEGE CODE: 1128

T.J.S ENGINEERING COLLEGE

COMPUTER SCIENCE ENGINEERING

PARTHIBAN R - aut112823CSE58
SANTHOSH B - aut112823104304
DEEPAK KUMAR S - aut23cse12a
VETRI GOWTHAM - aut112823104305
SUSEENDRAN - aut23cse79a

112823104058
112823104304
112823104012
112823104305
112823104079

DATE : 07/05/2025

COMPLETED THE PROJECT NAMED AS, TRAFFIC FLOW OPTIMIZATION

TECHNOLOGY- TRAFFIC FLOW OPTIMIZATION

SUBMITTED BY,

R PARTHIBAN
B SANTHOSH
S DEEPAK KUMAR
VETRI GOWTHAM
SUSEENDRAN

Mobile No. ;

R PARTHIBAN - 9025703218
B SANTHOSH - 9566108606
S DEEPAK KUMAR - 9566319120
VETRI GOWTHAM - 91596 61756
SUSEENDRAN - 97903 03271

Traffic Flow Optimization System - Phase 4 Report

Objective

Phase 4 focuses on enhancing the performance of the traffic flow optimization system by refining traffic prediction models, scaling the system to handle city-wide traffic data, and ensuring real-time responsiveness. This phase also includes optimizing sensor and IoT integration, improving system reliability, and strengthening data handling and security protocols.

1. Model Performance Enhancement

Overview:

The traffic prediction model will be refined using updated traffic datasets and feedback from prior testing. The goal is to improve the accuracy of congestion forecasts and optimize signal control in dynamic urban environments.

Performance Improvements:

- Model Retraining: Incorporating high-volume, real-time traffic datasets from urban centers to better predict traffic congestion patterns.
- Model Optimization: Techniques like parameter tuning and model pruning are applied to enhance the system's responsiveness and decision-making efficiency.

Outcome:

The model will deliver more accurate traffic predictions and optimized signal timing, resulting in reduced congestion and improved travel times.

2. User Interface & System Responsiveness

Overview:

The user interface, including mobile and web platforms for traffic monitoring, will be optimized for real-time updates and ease of use. The backend will be enhanced to handle higher data throughput without delays.

Key Enhancements:

- Low-Latency Updates: Performance tuning for faster updates of traffic maps and route suggestions.
- User Experience: UI/UX refinements for more intuitive interaction and better display of real-time data.

Outcome:

Users will experience smoother navigation, real-time rerouting suggestions, and intuitive system interaction under varying traffic conditions.

3. IoT and Sensor Integration

Overview:

Optimization of IoT and sensor networks (traffic cameras, speed detectors, GPS, etc.) for seamless data collection and processing across traffic zones.

Key Enhancements:

- Real-Time Data Handling: Ensuring minimal latency in collecting and analyzing data from roadside units and vehicles.
- API Optimization: Enhancing connections to traffic management platforms and smart infrastructure systems.

Outcome:

Improved traffic data collection and decision-making in real time, enabling dynamic traffic light control and predictive congestion management.

4. Data Security and Privacy Performance

Overview:

This phase ensures that user and traffic data is secure and handled according to best practices as the system scales to handle larger volumes of input.

Key Enhancements:

- Advanced Encryption: Implementing robust encryption to protect traffic data and system access.
- Security Testing: Conducting stress tests and vulnerability assessments to safeguard against breaches.

Outcome:

Secure handling of traffic and user data under heavy loads, ensuring compliance with data privacy regulations.

5. Performance Testing and Metrics Collection

Overview:

Comprehensive testing will ensure that the system performs reliably under real-world traffic conditions, with a focus on throughput, stability, and efficiency.

Implementation:

- Load Testing: Simulating peak traffic data flow to validate system resilience.

- Metrics Collection: Monitoring response times, data accuracy, and failure rates.

- User Feedback Loop: Gathering feedback from field trials in multiple city zones.

Outcome:

A robust system capable of scaling city-wide, maintaining high accuracy and performance across diverse traffic scenarios.

Key Challenges in Phase 4

1. System Scalability

- Challenge: Managing high traffic volumes across large networks.

- Solution: Implementing optimized algorithms and infrastructure for distributed traffic management.

2. Security Under Load

- Challenge: Ensuring system integrity during peak usage.

- Solution: Regular security audits and encryption reinforcement.

3. Sensor Compatibility

- Challenge: Integrating various traffic monitoring hardware.

- Solution: Standardizing API protocols and testing cross-device functionality.

Outcomes of Phase 4

1. Improved Traffic Predictions: More accurate congestion forecasts and dynamic routing.

2. Enhanced System Responsiveness: Real-time updates with reduced latency.

3. Optimized Data Collection: Efficient sensor data processing across the network.

4. Strengthened Security: Resilient protection for operational and user data.

Next Steps for Finalization

In the final phase, the system will be deployed in a live environment. Feedback from real-world operation will be used to fine-tune algorithms, improve user interfaces, and finalize documentation for large-scale rollout.

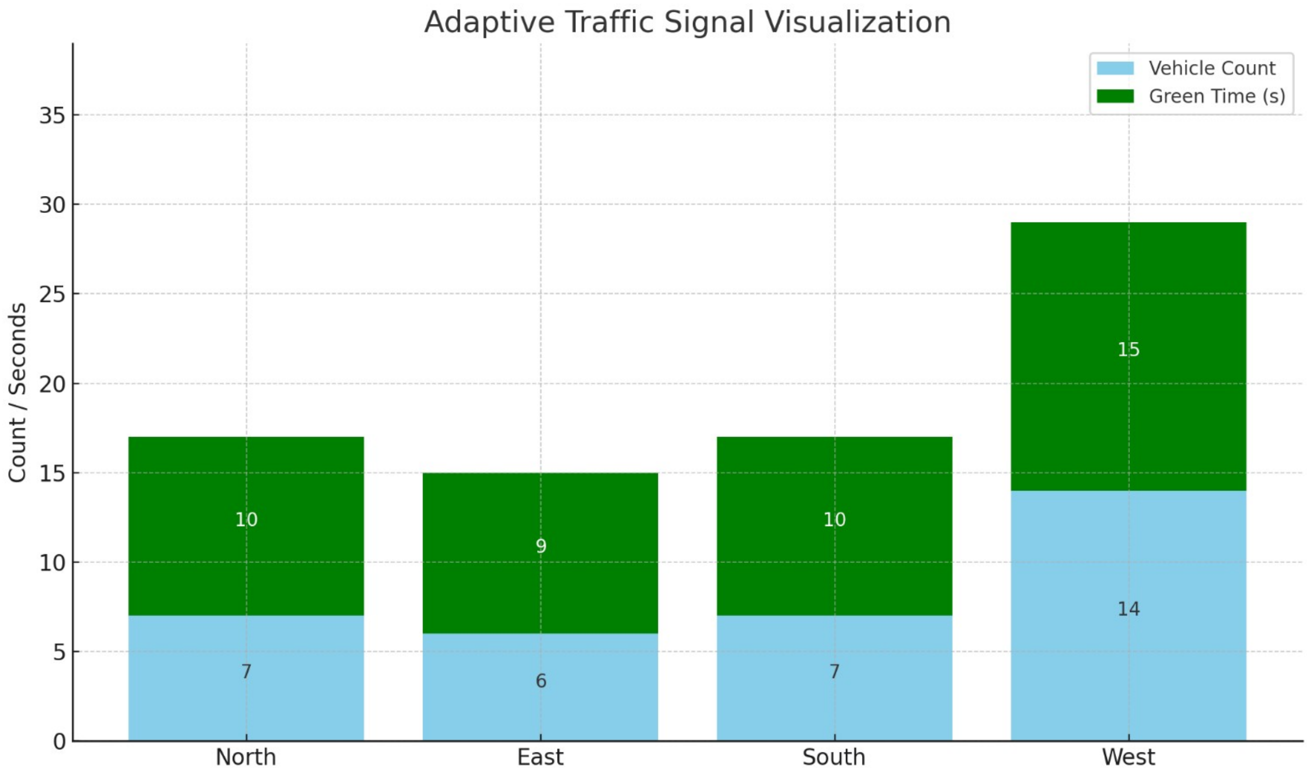
SOURCE CODE:

```
import random
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

class TrafficSignal:
    def __init__(self, direction):
        self.direction = direction
        self.vehicle_count = 0
        self.green_time = 10  # default green time in seconds

    def update_vehicle_count(self):
        self.vehicle_count = random.randint(0, 20)

    def calculate_green_time(self):
        self.green_time = 5 + (self.vehicle_count // 2)
        self.green_time = min(self.green_time, 30)

# Create signals
directions = ['North', 'East', 'South', 'West']
signals = [TrafficSignal(direction) for direction in directions]

# Create figure and axes
fig, ax = plt.subplots()

def update(frame):
    vehicle_counts = []
    green_times = []
    labels = []

    for signal in signals:
        signal.update_vehicle_count()
        signal.calculate_green_time()
        vehicle_counts.append(signal.vehicle_count)
        green_times.append(signal.green_time)
        labels.append(signal.direction)

    ax.clear()
    ax.set_title("Adaptive Traffic Signal Visualization")
    bar1 = ax.bar(labels, vehicle_counts, color='skyblue', label='Vehicle Count')
    bar2 = ax.bar(labels, green_times, bottom=vehicle_counts, color='green', label='Green Time (s)')

    for i in range(len(labels)):
        ax.text(i, vehicle_counts[i] / 2, str(vehicle_counts[i]), ha='center')
        ax.text(i, vehicle_counts[i] + green_times[i] / 2, str(green_times[i]), ha='center', color='white')

    ax.set_ylabel("Count / Seconds")
    ax.legend()

ani = FuncAnimation(fig, update, interval=3000)

plt.tight_layout()
plt.show()
```

CODE WITH SOME EXAMPLE INPUTS:

```
import random
import matplotlib
matplotlib.use('TkAgg')  # Use an interactive backend

import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

class TrafficSignal:
    def __init__(self, direction):
        self.direction = direction
        self.vehicle_count = random.randint(0, 10)
        self.green_time = 10

    def update_vehicle_count(self, frame):
        if 10 <= frame % 60 <= 20:
            change = random.randint(-1, 8)
        else:
            change = random.randint(-3, 3)
        self.vehicle_count = max(0, self.vehicle_count + change)

    def calculate_green_time(self):
        self.green_time = 5 + int(0.75 * self.vehicle_count)
        self.green_time = min(self.green_time, 30)

directions = ['North', 'East', 'South', 'West']
signals = [TrafficSignal(direction) for direction in directions]

fig, ax = plt.subplots()

def update(frame):
    vehicle_counts = []
    green_times = []
    labels = []

    for signal in signals:
        signal.update_vehicle_count(frame)
        signal.calculate_green_time()
        vehicle_counts.append(signal.vehicle_count)
        green_times.append(signal.green_time)
        labels.append(signal.direction)

    ax.clear()
    ax.set_title("Adaptive Traffic Signal Visualization")
    ax.set_ylabel("Count / Seconds")
    ax.set_ylim(0, max([vc + gt for vc, gt in zip(vehicle_counts, green_times)]) + 10)

    bar1 = ax.bar(labels, vehicle_counts, color='skyblue', label='Vehicle Count')
    bar2 = ax.bar(labels, green_times, bottom=vehicle_counts, color='green', label='Green Time (s)')

    for i in range(len(labels)):
        ax.text(i, vehicle_counts[i] / 2, str(vehicle_counts[i]), ha='center')
        ax.text(i, vehicle_counts[i] + green_times[i] / 2, str(green_times[i]), ha='center', color='white')

    ax.legend()

ani = FuncAnimation(fig, update, interval=3000)
plt.tight_layout()
plt.show(block=True)  # Ensure the window stays open
```

OUTPUT:



Adaptive Traffic Signal Visualization
Vehicle Count
Green Time (s)
Count / Seconds
0
5
10
15
20
25
30
35
North
East
South
West
10
7
9
6
10
7
15
14