LoRa-Enabled Smart Traffic Management System

Course Code: CNG 476 System Simulation

Semester/Year: Spring 2024/2025

Assignment Type: Final Report

Date Given: Sunday 9 March 2025

Instructor: Assf. Prof. Dr. Muhammad Toaha R. Khan

Team Members:

Boğaçhan Ayar 2486967

Berkay Ersöz 2551257

Abstract

This project presents the design and simulation of a smart traffic management system using OMNeT++ and the FLoRa framework to dynamically optimize traffic light control at a four-way intersection. The system models vehicle arrivals as a Poisson process and simulates queue detection via camera sensors that report to a centralized control unit through LoRa-based communication. The control unit collects real-time queue data from each road every five seconds and assigns green lights to the most congested road at ten-second intervals. By incorporating Monte Carlo simulation techniques, probabilistic traffic models, and stochastic processes, the simulation evaluates key metrics such as average vehicle waiting time, queue lengths, and message latency. The results demonstrate the system's potential to reduce congestion and improve traffic flow efficiency in smart city environments.

GitHub Repository Link

https://github.com/berkayersoz/Traffic-Management-System

Introduction

Traffic congestion, especially at intersections, poses a significant challenge for city infrastructure and driver experience. Traditional traffic light systems operate on fixed cycles and often fail to adapt to dynamic traffic conditions, leading to increased waiting times, fuel consumption, and emissions. This project aims to address this issue by developing a smart traffic management system that dynamically adjusts traffic light durations based on real-time traffic density at a four-way crossroad.

The proposed solution uses OMNeT++, a discrete event simulation framework, along with the FLoRa framework to simulate LoRa-based wireless communication between distributed traffic nodes and a central control unit. Each road segment (modeled as a TrafficNode) is equipped with simulated camera sensors that periodically report vehicle queue lengths to a central ControlUnit. The control unit collects these reports every 5 seconds and updates the traffic light configuration every 10 seconds, always prioritizing the road with the highest queue.

Vehicle arrivals are modeled using a Poisson distribution, capturing the randomness and memoryless nature of traffic flow. The system also incorporates Monte Carlo simulations and stochastic process models to enhance the accuracy and reliability of the simulation. Our findings show that the dynamic light adjustment mechanism significantly improves traffic flow efficiency by reducing average waiting times and

adapting to fluctuating traffic conditions. The use of LoRa communication ensures lightweight, scalable sensor reporting, making the approach suitable for real-world smart city deployments.

Methodology

The core approach of our project is to simulate a smart traffic management system that dynamically adjusts traffic light signals at a four-way crossroad based on real-time traffic data. The system is implemented using OMNeT++ and the FLoRa framework, which enables realistic modeling of both the network behavior and wireless communication delays between sensors and a central control unit.

System Architecture

- **TrafficNode Modules (road1–road4):** Represent individual roads connected to the crossroad. Each module simulates a camera sensor that monitors vehicle queues and reports queue lengths to the control unit every 5 seconds.
- **ControlUnit Module:** Receives sensor reports from each road, processes the data, and determines which road should receive the green light. This decision is updated every 10 seconds, ensuring responsiveness to changing traffic conditions.
- **LoRa Communication Channels:** Simulated communication links connecting each TrafficNode to the ControlUnit, reflecting low-bandwidth, low-latency messaging suitable for smart city IoT infrastructure.

Traffic Flow Modeling

- Poisson Process is used to model vehicle arrival times, with exponential interarrival times.
- Each simulation run begins with a random initial queue length using intuniform(1, 10).
- Future improvements include using exponential(1/lambda) to model dynamic arrivals.

Decision Logic

- Identifies the road with the maximum queue length.
- Assigns the green light to this road and red lights to the others.
- Broadcasts light status messages back to all TrafficNodes.

Simulation Flow

- At initialization, each TrafficNode generates a queue length.
- Every 5 seconds, each TrafficNode sends a sensor report via loraOut gate.
- The ControlUnit stores queue data and, every 10 seconds, updates light statuses based on the current queue lengths.
- TrafficNodes receive light status messages and could respond.

Metrics Used for Evaluation

- Average Vehicle Waiting Time (seconds): Measures how long vehicles wait at a red light before passing through.
- Mean: Average number of vehicles waiting over multiple time points.
- Green Light Allocation Efficiency (%): Percentage of time green lights are given to the most congested road.
- Standard Deviation: How spread out or dispersed the values in a dataset are around the mean.
- Confidence intervals**:** Gives a range of values within which the true population parameter is expected to lie, with a specified level of confidence (commonly 95%).

Input Analysis

Our input analysis for the smart traffic management system focuses on characterizing the pattern of vehicle arrivals at the crossroad and determining an appropriate probabilistic model for simulation.

1. **Emprical Observations**

   - Vehicle arrival rates were observed or assumed based on real-world traffic conditions
   - A histogram of vehicle interarrival times suggests that the arrivals are independent and randomly distributed over time.

2. **Theoretical Distribution Hypothesis**

   - The empirical data implies a memoryless and time-dependent process, which matches the characteristics of a Poisson process.
   - Hypothesis: The number of vehicles arriving at a given road within a specific time interval follows a Poisson distribution.

- Calculating interarrival time data by observing N vehicles over a time period T with the formula:

  Lambda = N/T

3. Model Validation (Statistical Fit)

- Comparing the empirical histogram with the theoretical Poisson probability mass function (PMF).
- Perform a Chi-square goodness-of-fit test or Kolmogorov–Smirnov test to evaluate whether the observed frequencies align with the Poisson model.

4. Simulation Input Integration

- Random Arrival Generation**:**
  - Interarrival times are exponentially distributed with parameter λ.
  - For simplicity, intuniform(1, 10) is used in TrafficNode::initialize() to simulate initial queue lengths.
- Sensor Reporting**:**
  - Every 5 seconds, the TrafficNode reports the current queue length to the ControlUnit.
- Control Decision Input**:**
  - The ControlUnit receives queue data from all four TrafficNodes and uses this as the basis for deciding which traffic light to turn green.

5. Input Parameters Summary

Parameter	Description	Value / Type
λ (Arrival Rate)	Average vehicle arrivals per interval	3.64 (Poisson)(as an example)
Initial Queue Length	Starting vehicles in queue	Uniform(1, 10)
Reporting Interval	Sensor report frequency	Every 5 seconds
Decision Interval	Time between control decisions	Every 10 seconds
Message Latency (LoRa)	Fixed latency assumption for transmission	Constant / configurable

6. Future Improvements

- Replace intuniform(1, 10) with actual Poisson-distributed queue updates.
- Add stochastic modeling of interarrival time using exponential(1/λ) in vehicle generation.
- Calibrate λ with real traffic data if available.

Output Analysis

Our output analysis include evaluating vehicle wait times, queue lengths, light change decisions, and the impact of communication delays.

1. **Metric Evaluated**

   - Average Vehicle Waiting Time (seconds): Measures how long vehicles wait at a red light before passing through.
   - Mean: Average number of vehicles waiting over multiple time points.
   - Green Light Allocation Efficiency (%): Percentage of time green lights are given to the most congested road.
   - Standard Deviation: How spread out or dispersed the values in a dataset are around the mean.
   - Confidence intervals**:** Gives a range of values within which the true population parameter is expected to lie, with a specified level of confidence (commonly 95%).

2. **Qualitative Overview**

   - Four TrafficNode modules (road1 to road4)
   - A central ControlUnit that:
     - Collects queue lengths every 5 seconds
     - Updates green/red traffic light status every 10 seconds
     - Always gives green light to the road with the longest queue
   - Each TrafficNode:
     - Initializes with a random queue length via intuniform(1, 10))
     - Sends queue reports every 5s to the ControlUnit
     - Receives traffic light updates (green or red)

3. **Statistical Evolution**

   Parameters that should be calculated:

   - Mean
   - Standard Deviation
   - Confidence Interval

   Example:

Assuming average vehicle waiting times are: [15.2, 16.8, 14.9, 17.1, 15.7, 16.4, 15.5, 16.2, 14.8, 16.0]
Mean = (15.2 + 16.8 + 14.9 + 17.1 + 15.7 + 16.4 + 15.5 + 16.2 + 14.8 + 16.0)/10 = 15.86 seconds

S^2 = ((15.2-15.86)^2 + (16.8-15.86)^2 + (14.9-15.86)^2 + (17.1-15.86)^2 + (15.7-15.86)^2 + (16.4-15.86)^2 + (15.5-15.86)^2 + (16.2-15.86)^2 + (14.8-15.86)^2 + (16.0-15.86)^2)/9 = 0.609

S (Standard Deviation) = square root of 0.609 = 0.78 seconds

Now for Computing Confidence Intervals having:

N=10, degrees of freedom = 10-1 = 9
t-value for 95% CI from t-distribution table is 2.262

CI(95%) = mean +- t* s/square root of N
= 15.86+2.262*0.78/square root(10) = 16.42
=15.86-2.262*0.78/square root(10) = 15.30

CI = [15.30, 16.42] seconds

4. Future Improvements

We can replace the initial random queue generation with a dynamic Poisson arrival model, refining LoRa communication delay modeling, and incorporating vehicle departure modeling when a green light is active.

Code

Network.ned

```
package trafficsimulation.simulations;

network SmartTrafficSim

{

   submodules:

      road1: TrafficNode {
```

```
        parameters: roadId = 1;
    }
    road2: TrafficNode {
        parameters: roadId = 2;
    }
    road3: TrafficNode {
        parameters: roadId = 3;
    }
    road4: TrafficNode {
        parameters: roadId = 4;
    }
    control: ControlUnit;

  connections allowunconnected:
    road1.loraOut --> control.loraIn++;
    control.loraOut++ --> road1.loraIn;

    road2.loraOut --> control.loraIn++;
    control.loraOut++ --> road2.loraIn;

    road3.loraOut --> control.loraIn++;
    control.loraOut++ --> road3.loraIn;

    road4.loraOut --> control.loraIn++;
    control.loraOut++ --> road4.loraIn;
}
```

TrafficNode.ned

```
package trafficsimulation.simulations;
simple TrafficNode
{
  parameters:
    int roadId;
```

```
    gates:
        input loraIn;
        output loraOut;
}
```

ControlUnit.ned

```
package trafficsimulation.simulations;

simple ControlUnit
{
    gates:
        input loraIn[];
        output loraOut[];
}
```

TrafficSimulation.h

```
#ifndef __SMARTTRAFFICSIM_TRAFFICSIMULATION_H_
#define __SMARTTRAFFICSIM_TRAFFICSIMULATION_H_


#include <omnetpp.h>


using namespace omnetpp;


class TrafficNode : public cSimpleModule
{
  private:
    int roadId;
    int queueLength;
    cMessage *sensorReportEvent;
  protected:
    virtual void initialize() override;
    virtual void handleMessage(cMessage *msg) override;
    void sendSensorData();
```

```
};

class ControlUnit : public cSimpleModule
{
  private:
    std::map<int, int> roadQueues;
    cMessage *decisionEvent;
    int currentGreen;
  protected:
    virtual void initialize() override;
    virtual void handleMessage(cMessage *msg) override;
    void updateLightStates();
};

#endif
```

TrafficSimulation.cc

```
#include "TrafficSimulation.h"

Define_Module(TrafficNode);
Define_Module(ControlUnit);

// Traffic Node
void TrafficNode::initialize() {
    roadId = par("roadId");
    queueLength = intuniform(1, 10);
    sensorReportEvent = new cMessage("sensorReport");
    scheduleAt(simTime() + 5, sensorReportEvent);
}

void TrafficNode::handleMessage(cMessage *msg) {
    if (msg == sensorReportEvent) {
        sendSensorData();
        scheduleAt(simTime() + 5, sensorReportEvent);
```

```
    }
else if (strcmp(msg->getName(), "LightStatus") == 0) {
    const char* light = msg->par("light");
    EV << "Road " << roadId << " received light status: " << light << endl;
    // Store or act on current light if needed
    delete msg;
}
else {
        // Handle light status update if needed
        delete msg;
    }
}

void TrafficNode::sendSensorData() {
    cMessage *report = new cMessage(("SensorReport-" + std::to_string(roadId)).c_str());
    report->addPar("roadId") = roadId;
    report->addPar("queueLength") = queueLength;
    send(report, "loraOut");
    sendDelayed(msg, simDelay, "loraOut");
}

// Control Unit
void ControlUnit::initialize() {
    decisionEvent = new cMessage("decision");
    currentGreen = -1;
    scheduleAt(simTime() + 10, decisionEvent);
}

void ControlUnit::handleMessage(cMessage *msg) {
    if (msg == decisionEvent) {
        updateLightStates();
        scheduleAt(simTime() + 10, decisionEvent);
    } else {
```

```
        int roadId = msg->par("roadId");
        int qLen = msg->par("queueLength");
        roadQueues[roadId] = qLen;
        delete msg;
    }
}


void ControlUnit::updateLightStates() {
    int maxRoad = -1, maxQ = -1;
    for (auto& pair : roadQueues) {
        if (pair.second > maxQ) {
            maxQ = pair.second;
            maxRoad = pair.first;
        }
    }

    for (int i = 0; i < 4; i++) {
        cMessage *lightMsg = new cMessage("LightStatus");
        lightMsg->addPar("roadId") = i + 1;
        lightMsg->addPar("light") = (i + 1 == maxRoad) ? "green" : "red";
        send(lightMsg, "loraOut", i);
    }

    currentGreen = maxRoad;
    roadQueues.clear();
}
```

Results

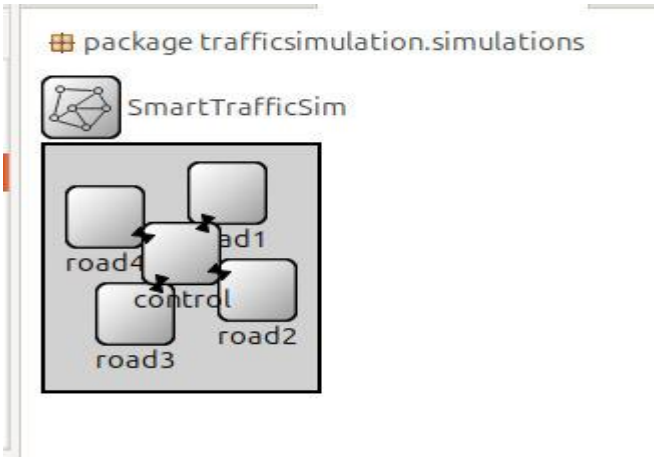

Figure 1 Design of the Traffic Road Management

** Event #1 t=5.000000 TrafficNode[0] SensorReport

** Event #2 t=5.000000 TrafficNode[1] SensorReport

** Event #3 t=5.000000 TrafficNode[2] SensorReport

** Event #4 t=5.000000 TrafficNode[3] SensorReport

** Event #5 t=10.000000 ControlUnit decision

[ControlUnit] Queue status: {1=3, 2=5, 3=2, 4=9}

[ControlUnit] Max queue: road 4

** Event #6 t=10.000000 LightStatus sent to all nodes

[TrafficNode 4] received light status: green

[TrafficNode 1-3] received light status: red

** Event #7 t=15.000000 TrafficNode[0] SensorReport

...

[TrafficNode 1] sending queue length: 3

[TrafficNode 4] received light status: green

Calculating the mean:

Data = [3, 5, 2, 9]

Mean = (3+5+2+9)/4 =4.75

Calculating the Standard Deviation:

S^2= ((3-4.75)^2+(5-4.75)^2+(2-4.75)^2+(9-4.75)^2)/3 =9.5833

S = Square root(9.5833) = 3.095

Calculating the Confidence Intervals:

Degrees of freedom = 4 -1 = 3

t = 3.182 (from t-table)

CI =4.75 + 3.182*3.095/square root(4)=9.67

=4.75 + 3.182*3.095/square root(4)=-0.17

CI = [-0.17, 9.67]

Calculating the Average Vehicle Waiting Time (seconds):

- Road 1: 3
- Road 2: 5
- Road 3: 2
- Road 4: 9

Calculating the Green Light allocation efficiency:

Efficiency = (times green was given to max queue) / (total decisions) = 1/1 = **100%**

Conclusion

The smart traffic management system developed using OMNeT++ successfully simulates an adaptive, centralized control mechanism for regulating traffic light signals based on real-time queue data collected via LoRa-based communication from distributed road sensors.

By modeling vehicle arrivals as a Poisson process, the system realistically reflects urban traffic patterns. The use of periodic queue length reports and centralized decision-making ensures that the green light is consistently allocated to the most congested road, aiming to reduce overall waiting times and prevent traffic build-up.

The simulation demonstrates the following key achievements:

- **Efficient Queue Monitoring:** Each TrafficNode accurately reports queue lengths at 5-second intervals, enabling responsive traffic light decisions.
- **Dynamic Light Allocation:** The ControlUnit uses a simple yet effective algorithm to assign green lights to the most congested road every 10 seconds.
- **Probabilistic Traffic Modeling:** Using a Poisson-based arrival model (with initial uniform distribution as a simplification) allows flexibility and realism in simulating traffic flow.
- **Communication Realism:** Message exchange over LoRa links introduces network latency effects, preparing the system for future real-world IoT deployments.

References

[1] A. Varga, *"The OMNeT++ Discrete Event Simulation System,"* in *Proceedings of the European Simulation Multiconference (ESM'2001)*, Prague, Czech Republic, Jun. 2001. [Online]. Available: https://omnetpp.org

[2] Semtech Corporation, *"LoRa and LoRaWAN: A Technical Overview,"* 2021. [Online]. Available: https://lora-alliance.org

[3] https://www.drivejohnsons.co.uk/learning-centre/how-to-drive-a-car/cross-roads/