Assumption :

Intersection manager (IM) structure:

The IM is divided in three areas:

1. **Storage area** - This is where the vehicles first arrive and give requests to the IM at a certain time. In my iteration, this area is divided into three lanes (For example: a lane that goes right, straight ahead and to left) for each of the four directions.

For each of the lane, it is divided into 3 zones (to keep a time marker), each certain distance from the intersection

Zone A: 10 meter from the intersection

Zone B: 20 meter from the intersection

Zone C: 30 meter from the intersection

1. **Conflict area -** Once the vehicles are 'CONFIRM'ed by the IM, they enter the intersection (irrespective of direction), such that:
   1. The path (or nodes) in front of them are cleared. The vehicle that comes first has the first priority (except in case of emergency vehicles) to go through the nodes
   2. In case of an emergency vehicle in the storage area, they go before any other vehicles in their lanes (including the ones in from of them)

1. **Exit area** - The vehicles enters this area after it leaves the last node, marking the last time the IM communicates with them

Powers:

1. It is solely up to the intersection manager to let vehicles inside the conflict area/intersection.
2. Vehicles are allowed to move at their will in the storage area. For instance, when vehicles give way for emergency vehicles in the storage area, they move on their own.
3. When vehicles move out of their way for an emergency vehicle, they are eliminated from the IM log of vehicles, such that the moved vehicles need to REQUEST again for a time slot to enter the conflict area.

V to I Communication:

Vehicle:

**REQUEST** signal : The vehicle shares their **ID, origin lane, destination lane, current location** (in terms of nodes - or position of vehicle with in a certain radius of the intersection), and **the current time**

As vehicles go from one node to another in the storage or conflict area, the time is being saved to calculate the entry\_time (to the intersection) and other time entities (based on any other new functions)

**CANCEL** signal : The vehicle can also cancel the previous REQUEST at their own will or in case of an emergency vehicles in their own lane

Intersection:

Assess the log of vehicles it received to calculate who gets the first time slot for each path, concurrently.

**CONFIRM** signal: For the vehicles that has an open path, the CONFIRM signal from the Intersection manager allows the vehicles to enter the conflict area

I to I communication:

The scope of this communication is not defined yet.

V to V communication:

The scope of this communication is not defined yet.

A cross with cars and numbers

Description automatically generated with medium confidence

Threat model:

Giving way for Emergency vehicle is an important part of a IM, given the significance of their urgency. However, if ordinary vehicles have the capacity to impose as emergency vehicles, not only would it cause unnecessary delays but also will constrain actual emergency vehicles from performing their important tasks.

So in a malicious environment:

1. There could be multiple imposters (acting as Emergency vehicles), in every direction halting traffic at an intersection completely
2. There could be just one or two imposters in the IM, such that they always get to trick the system, which becomes a major flaw in the system and unfair to other law abiding cars
3. Or in the worst case, slow down an actual emergency vehicle by tagging along with them or in other ways, which is then a threat to life and property

What sets rogue imposter vehicles apart from actual emergency vehicles?

A powerful adversary may have all the markings of a real emergency vehicle, which may make it difficult to identify with appearance or IDs.

Since mitigation has to be based on traits that cannot be mimicked by an imposter, here are a few:

1. Behavior: Rogue vehicles may exhibit unusual behavior that is inconsistent with standard emergency response procedures. For instance it may not pass through the a certain number intersection to reach a said emergency destination, in which case a I to I communication should be established. A ML model could predict the type of vehicle based on how they behave in the previous intersections.

Pros: The IM can learn the rogue nature of a vehicle with the data from other IMs, hence not giving priority anymore

Cons:

1. Radio frequencies emitted: Assuming certain vehicles communicate in certain range of frequency and that IMs can know the RFIDs of emergency vehicles

Pros: A single intersection may be able to identify the rogue vehicle

Cons: Log of all confidential RFID at every intersection may be subject to attack

Discussion:

it’s desirable to catch it as early as possible (and that should be a metric for evaluation) it’s not a catastrophe if it’s not caught immediately  (even though catching immediately would be ideal).

First we learn emergency vehicle behavior or set it.

Pros and Cons of assumptions

Adversary: Self sustained rogue vehicle

|  |  |  |  |
| --- | --- | --- | --- |
| Metric of detection | Pros | Cons | Solution |
| Unforgettable ID | The ID is always stored in a cloud and can detect imposters at the very first intersection | Data Leak | Forgettable ID |
| Forgettable ID | Not susceptible to any data leak | Cannot track vehicular motion between intersections tied to a given ID | Generate a Behavioral ID unique to every vehicle type and track that |
| Untrusted I2I |  | No way to detect |  |
| Trsuted I2I | Only way to catch a rogue vehicle is trusted I to I (as of now) | -- |  |
| Untrusted V2V |  |  |  |
| Trusted V2V | Detection method if V2I is untrusted in case of communication intercepted | If an imposter can lie to the IM couldn’t it lie to other vehicles |  |
| Trusted V2I | No extra measures required to secure the comm channel |  |  |
| IMs access to the entire EM database | Easy detection | Data Leak | Forgettable ID |

Assumption for ID:

**Constant assumption: Forgettable unique ID, no permanent database of previous vehicle IDs, trusted I2I**

**Communication includes (so far) ID, speed, acceleration, time, origin, destination. Could include GPS coordinates and others at later stage of mitigation**

Case 1: Forgettable unique ID, with intersection having no database of previous vehicles with **trusted V2I** and I2I, **untrusted/trusted V2V**

Pros:

Cons: Trusting V2I would mean there could never be rogue vehicles (vehicles cannot lie about themselves), irrespective of if V2V is trusted or not

Case 2: Forgettable unique ID, with intersection having no database of previous vehicles with **trusted V2I** and I2I, **trusted V2V**

Pros: All three comm channel are right all the time

Cons: V2V may communicate what they see at the moment, lets say visual identifiers, speed, acceleration etc., but it does not have the ability or time to learn the type of the vehicle that’s beside it

Also trusting all the three communication channel would make it a trivial problem

Case 3: Forgettable unique ID, with intersection having no database of previous vehicles with trusted I2I **untrusted V2I,** **trusted V2V**

Pros: Untrusted V2I raises the question of rogue vehicle

Cons: V2V may communicate what they see at the moment, lets say visual identifiers, speed, acceleration etc., but it does not have the ability or time to learn the type of the vehicle that’s beside it

Case 4: Forgettable unique ID, with intersection having no database of previous vehicles with **trusted I2I** **untrusted V2I,** un**trusted V2V**

Pros: Untrusted V2I raises the question of rogue vehicle,

Cons:

Assumptions:

Forgettable unique ID, with intersection having no database of previous vehicles with **trusted I2I** **~~untrusted~~ V2I, trusted V2V,** IM has a data base of **emergency events** (database not encouraged)

All trusted channels (I2I, V2I, V2V), but rogue/untrusted vehicle and trusted intersection

Vehicle characteristics can be differentiated based on three factors: braking patterns, acceleration and turning events (Minh 2013). Turn events has a faster prediction rate (Hallac 2017)

**Problems to solve:**

**1. Track a vehicle across intersections**

Can be assessed by turning events, inertial sensors, braking and acceleration patterns

**2. Distinguish between Real emergency vehicles vs Imposter emergency**

How does the IM know:

* + Access of the database from state or city (Not recommended)
  + Identify accidents on the road based on the density of traffic flow. Less dense and slow flow of traffic may indicate accident or emergencies

Vehicle:

Part 1: Trajectory mapping

* + Is it going to an emergency location – trajectory mapping prediction based on known IM data and possible routes it can go
  + Predict future trajectories from vehicle to vehicle interactions: A vehicle receives information, such as Global Positioning System coordinates, about nearby vehicles on the road using inter-vehicular communication. The collected data from vehicles together with GPR models received from infrastructure are then used to predict the future trajectories of vehicles in the scene. (Sepideh 2018) (The IM needs to do this learning)

Part 2: Analyzing doppler effect

* + Visual and audio responses (how long were their sirens and lights on) – doppler effect
    - Cant use this as IM should prevent future emergency vehicles from making noise or light pollution.

Part 3: Analyzing

Increasing acceleration or velocity for emergency vehicle

**Can the real EM and imposter EM be distinguished, without a database of emergency events?**

Rational Approach:

Every time a vehicle passes an intersection, not following priority, it gets fined and the fine is only lifted once it reaches the emergency location.

Rational Vehicle: Does not want harm to itself and abide by the rules

Irrational Vehicle: Willing to pay the money and cause attack in order to:

* Slow down other emergency vehicles
* Or simply to get an earlier timeslot, assuming they don’t care about the consequence.

In both cases, while a single rogue vehicle may not cause significant delay, multiple vehicles can cause a significant delay.

Solutions (updated):

**Distinguish between real and rogue EM**: Trajectory mapping

* + Is it going to an emergency location – trajectory mapping prediction based on known IM data and possible routes it can go
  + Predict future trajectories from vehicle to vehicle interactions: A vehicle receives information, such as Global Positioning System coordinates, about nearby vehicles on the road using inter-vehicular communication. The collected data from vehicles together with GPS models received from infrastructure are then used to predict the future trajectories of vehicles in the scene. (Sepideh 2018) (The IM needs to do this learning)
  + Compare the calculated future trajectories with where they said they are actually going

**Track a vehicle (without ID) across intersections**

* Can be assessed by turning events, inertial sensors, braking and acceleration patterns

**Update October 26**

**Assumptions:**

We have a grid of multiple intersections connecting each other, where vehicle data is stored temporarily for learning and identifying in the next intersection. This also means the vehicles need to pass at least one intersection to analyze its data.

**Primary Threat:**

Rogue vehicles posing as an emergency vehicle to get false higher priority.

Solution:

* 1. Track future trajectory of a vehicle using its previous coordinates, velocity and acceleration using machine learning. This future trajectory calculated by the intersection manager can be compared with the location they are claiming to go to, to see if they are telling the truth. This way there is no need for a database of all emergency events in the area.
  2. Learning the velocity or acceleration pattern of emergency vehicles in relation to normal vehicles. For instance, emergency vehicles may have a constant velocity curve, while regular vehicles may decelerate or stop more abruptly, creating an uneven curve.

**Secondary threat: Adversary capability**

Vehicle can change IDs between intersection making it difficult to track a vehicle:

1. Each vehicle can use its driving pattern as an indicator of its uniqueness. Vehicle characteristics can be differentiated based on three factors: braking patterns, acceleration and turning events (Minh 2013). Turn events have a faster prediction rate (Hallac 2017), since it both accelerates and decelerates.

Next steps:

Collect data from SUMO or CARLA for vehicles movement between intersections.

Game theory:

In the model you've described, we have an Intersection Manager (IM) that operates in a manner similar to a network router or a traffic signal controller, but with more sophisticated logic and real-time decision-making capabilities. To prevent irrational vehicles from gaining an advantage, the IM can implement a variety of game-theoretical strategies. Here is how the model can integrate these strategies:

**1. Priority Assignment:**

* The IM uses a priority queue to manage REQUEST signals. Emergency vehicles are given preemptive priority, meaning their REQUESTs supersede all others irrespective of arrival time.
* For all other vehicles, a first-come, first-served policy is the default, but it can be adjusted for other policies, such as prioritizing public transportation.

**2. Zone-Based Time Slot Allocation:**

* The IM allocates time slots based on zones. Vehicles in Zone A have higher priority over those in Zones B and C. However, time slots are contingent on path availability.
* If an emergency vehicle enters any zone, it is granted immediate access if possible, reassigning time slots of non-emergency vehicles as needed.

**3. Penalty for Rule Violation:**

* If a vehicle attempts to proceed without a CONFIRM signal, the IM logs the violation. Repeat offenders may receive longer wait times or lower priority in the future.
* This "punishment" strategy deters rational but rule-breaking behavior by increasing the cost of non-compliance over time.

**4. Dynamic Path Clearance:**

* The IM can dynamically manage the conflict area by adjusting the path clearance in real-time, taking into account the current vehicle mix and any emergent situations.
* The IM's algorithms should be resilient to attempts by irrational vehicles to "game" the system by sending false REQUEST or CANCEL signals.

**5. Strategic Movement in Storage Area:**

* While vehicles can move at will in the storage area, strategic repositioning (e.g., making way for an emergency vehicle) results in the need to REQUEST again.
* The IM can keep track of these strategic moves and potentially reward cooperative behavior with a higher priority upon reentry.

**6. Communication Overhead Management:**

* To prevent the communication system from being overwhelmed with REQUESTs and CANCELs from irrational vehicles, the IM could implement a rate-limiting function or require a cooldown period between signals from the same vehicle.

**7. Confirmation and Cancelation Protocols:**

* Upon issuing a CONFIRM signal, the IM expects the vehicle to enter the conflict area within a certain time window. Failure to do so without a valid CANCEL signal may result in a penalty.
* CANCEL signals from non-emergency vehicles that are strategic in nature (to allow emergency vehicles to pass) could be recognized and these vehicles can be given priority in re-issuing REQUESTs.

**8. Game Theoretical Enhancements:**

* **Signaling Game:** The IM could use the signals as a type of 'signaling game' where it learns which signals are likely to be credible and which are not, based on past behavior.
* **Nash Equilibrium:** The IM operates to achieve a Nash Equilibrium where no vehicle has anything to gain by changing only their own strategy unless they are an emergency vehicle.

In summary, by adopting a smart, real-time management system for intersection traffic control, we can create an environment where the costs of irrational behavior are increased, and cooperative, rule-following behavior is rewarded, thus pushing the system towards an equilibrium where the majority of vehicles act rationally. The use of such an intersection management system could potentially serve as a deterrent against irrational behavior, thereby reducing the likelihood of such vehicles gaining any advantage.

Game theory enhancements in the context of the intersection management (IM) system involve leveraging game-theoretic principles to inform and enhance the decision-making algorithms that control the flow of traffic. Let’s explore how these could be applied to the IM system you've described:

**Signaling Game:**

In a signaling game, there are typically two players: a sender, who sends a signal, and a receiver, who interprets the signal and makes a decision. Here, the vehicles are senders of signals (REQUEST or CANCEL) and the IM is the receiver.

* **Honest Signaling:** Over time, the IM can develop a model to assess the credibility of signals based on past behavior. If a vehicle frequently sends a CANCEL signal, only to re-REQUEST quickly, it could be interpreted as an attempt to game the system.
* **Penalties and Rewards:** The IM can penalize frequent cancellers by lowering their priority or reward honest signaling by granting a slight priority boost. This creates a self-enforcing mechanism where honest signaling is in the best interest of all players.

**Nash Equilibrium:**

The Nash Equilibrium occurs when players are making decisions that are optimal given the decisions of other players, and no one has anything to gain by changing only their own strategy. In the IM context:

* **Strategic Decision-making:** The IM could be programmed to identify strategies that bring the system towards a Nash Equilibrium, where vehicles are neither incentivized to break rules nor to manipulate their signals.
* **Predictive Modeling:** By predicting the actions of rational vehicles, the IM can optimize traffic flow and minimize the impact of irrational behavior.

**Evolutionary Stable Strategy (ESS):**

An ESS is a strategy which, if adopted by a population in a game, cannot be invaded by any alternative strategy that is initially rare. It is a refinement of Nash Equilibrium to situations with a population of players:

* **Long-Term Adaptation:** The IM can analyze traffic patterns over long periods to identify ESS within the context of intersection management, discouraging strategies that attempt to exploit the system.
* **Feedback Loops:** By creating feedback loops where the strategies of vehicles are monitored over time, the IM can adjust its control algorithms to promote strategies that contribute to efficient traffic flow.

**Tit-for-Tat and Cooperative Strategies:**

These strategies involve cooperation based on reciprocity. In traffic systems:

* **History-Based Allocation:** The IM could employ a tit-for-tat approach where the future allocation of time slots to vehicles could depend on their previous behavior.
* **Cooperation Incentives:** Vehicles that have demonstrated cooperative behavior, such as yielding to emergency vehicles, could be remembered by the system and given preferential treatment in future interactions.

**Mechanism Design:**

Mechanism design is a field in economics and game theory that takes an engineering approach to designing economic mechanisms or incentives, toward desired objectives, in strategic settings where players act rationally. In this system:

* **Incentive Structures:** The IM could create incentive structures to ensure that following the rules is always the best option for rational vehicles.
* **Social Optimum:** Design mechanisms that align individual incentives with the social optimum, ensuring the smooth flow of traffic through the intersection.

By incorporating these game-theoretic enhancements into the IM, the intersection can be managed in a way that discourages behavior that leads to inefficiencies or accidents, while encouraging a smooth and cooperative flow of traffic. The application of these principles requires sophisticated modeling and real-time data processing capabilities, which would be part of the intelligent traffic management system's software infrastructure.

ML

In some cases, a combination of both LSTM and GNN might be the best approach, where LSTM can handle the temporal aspects and GNN can handle the spatial and relational aspects of the data.

* **Data Nature**: If your data is primarily time-series (like historical vehicle positions and movements), an LSTM might be more suitable.
* **Task Focus**: If the task is to predict where a vehicle will go based on its past trajectory, without needing to deeply understand the influence of the overall road network structure, an LSTM would likely serve your needs better.
* **Road Network Influence**: If understanding how the vehicle interacts with the road network (like route choices at intersections, effect of road closures, etc.) is crucial, then a GNN might offer more insights.

To use a combination of LSTM and GNN for predicting future vehicle trajectories, you'll need to prepare a dataset that includes features suitable for both temporal and graph-based analysis. Here’s a detailed procedure including the necessary dataset columns and how they can be used by the LSTM and GNN components:

Dataset Columns

For LSTM (Temporal Analysis):

Global\_X, Global\_Y: GPS coordinates at each time step.

Global\_Time: Time stamps for each recorded position.

Speed: Vehicle speed at each time step.

Acceleration: Vehicle acceleration at each time step.

For GNN (Graph-Based Analysis):

Road\_Segment\_ID: Identifier for the road segment where the vehicle is located.

Intersection\_ID: Identifier for the nearest intersection or relevant graph node.

Adjacent\_Segment\_IDs: Identifiers for adjacent road segments.

Traffic\_Flow: Information about traffic conditions on each road segment (if available).

Data Preparation

Preprocess Data:

Ensure all data is clean, normalized, and structured appropriately for the model.

Split the data into training, validation, and testing sets.

Feature Engineering:

Calculate any additional features needed (e.g., change in direction, relative positioning to other vehicles).

Model Architecture

LSTM Component:

Use Global\_X, Global\_Y, Speed, Acceleration, and Global\_Time as input features.

This component learns the temporal dynamics of vehicle movement.

GNN Component:

Construct a graph where nodes represent road segments or intersections, and edges represent connections between them.

Use features like Road\_Segment\_ID, Intersection\_ID, and Traffic\_Flow for each node.

This component learns the spatial relationships within the road network.

Integration:

Combine the outputs of the LSTM and GNN. This can be done by concatenating the last hidden state of the LSTM with the node embeddings from the GNN, followed by a dense layer for prediction.

Training the Model

Loss Function and Optimizer:

Use a regression loss function like Mean Squared Error (MSE).

Choose an optimizer like Adam.

Training Process:

Train the integrated model on the training dataset.

Regularly validate the model on the validation dataset to monitor performance and avoid overfitting.

Hyperparameter Tuning:

Adjust hyperparameters like learning rate, LSTM sequence length, number of GNN layers, batch size, etc.

Evaluation:

Evaluate the final model on the test set to assess its performance.

Post-Training

Model Testing and Validation:

Test the model on new, unseen data.

Validate the predictions against real-world scenarios or simulations.

Iteration and Improvement:

Based on testing, you might need to iterate on the model architecture or features.

Deployment:

Deploy the model for practical applications, ensuring it integrates well with the existing systems.

This approach leverages the strengths of both LSTM (for learning temporal patterns in trajectory data) and GNN (for incorporating the spatial context of the road network), providing a comprehensive model for vehicle trajectory prediction. The success of the model will depend on the quality of the data, the model's architecture, and the tuning of its parameters.

To use a combination of LSTM and GNN for predicting future vehicle trajectories, you'll need to prepare a dataset that includes features suitable for both temporal and graph-based analysis. Here’s a detailed procedure including the necessary dataset columns and how they can be used by the LSTM and GNN components:

**Dataset Columns**

1. **For LSTM (Temporal Analysis)**:
   * **Global\_X**, **Global\_Y**: GPS coordinates at each time step.
   * **Global\_Time**: Time stamps for each recorded position.
   * **Speed**: Vehicle speed at each time step.
   * **Acceleration**: Vehicle acceleration at each time step.
2. **For GNN (Graph-Based Analysis)**:
   * **Road\_Segment\_ID**: Identifier for the road segment where the vehicle is located.
   * **Intersection\_ID**: Identifier for the nearest intersection or relevant graph node.
   * **Adjacent\_Segment\_IDs**: Identifiers for adjacent road segments.
   * **Traffic\_Flow**: Information about traffic conditions on each road segment (if available).

**Data Preparation**

1. **Preprocess Data**:
   * Ensure all data is clean, normalized, and structured appropriately for the model.
   * Split the data into training, validation, and testing sets.
2. **Feature Engineering**:
   * Calculate any additional features needed (e.g., change in direction, relative positioning to other vehicles).

**Model Architecture**

1. **LSTM Component**:
   * Use **Global\_X**, **Global\_Y**, **Speed**, **Acceleration**, and **Global\_Time** as input features.
   * This component learns the temporal dynamics of vehicle movement.
2. **GNN Component**:
   * Construct a graph where nodes represent road segments or intersections, and edges represent connections between them.
   * Use features like **Road\_Segment\_ID**, **Intersection\_ID**, and **Traffic\_Flow** for each node.
   * This component learns the spatial relationships within the road network.
3. **Integration**:
   * Combine the outputs of the LSTM and GNN. This can be done by concatenating the last hidden state of the LSTM with the node embeddings from the GNN, followed by a dense layer for prediction.

**Training the Model**

1. **Loss Function and Optimizer**:
   * Use a regression loss function like Mean Squared Error (MSE).
   * Choose an optimizer like Adam.
2. **Training Process**:
   * Train the integrated model on the training dataset.
   * Regularly validate the model on the validation dataset to monitor performance and avoid overfitting.
3. **Hyperparameter Tuning**:
   * Adjust hyperparameters like learning rate, LSTM sequence length, number of GNN layers, batch size, etc.
4. **Evaluation**:
   * Evaluate the final model on the test set to assess its performance.

**Post-Training**

1. **Model Testing and Validation**:
   * Test the model on new, unseen data.
   * Validate the predictions against real-world scenarios or simulations.
2. **Iteration and Improvement**:
   * Based on testing, you might need to iterate on the model architecture or features.
3. **Deployment**:
   * Deploy the model for practical applications, ensuring it integrates well with the existing systems.

Certainly! The provided Python script uses SUMO's Traffic Control Interface (TraCI) to implement custom intersection management logic, focusing on prioritizing emergency vehicles and applying FIFO (First-In-First-Out) rules for other vehicles at intersections. Here's a breakdown of the key parts of the script:

Importing Modules

python

Copy code

import os

import sys

import traci

traci: The module for interacting with the running SUMO simulation.

Function Definitions

is\_emergency\_vehicle(vid)

python

Copy code

def is\_emergency\_vehicle(vid):

return traci.vehicle.getVehicleClass(vid) == "emergency"

This function checks if a given vehicle (identified by vid, its vehicle ID) is an emergency vehicle. It returns True if the vehicle class is "emergency".

run()

python

Copy code

def run():

intersection\_approach\_distance = 30 # Distance to check for vehicles approaching intersection

handled\_vehicles = set() # Keep track of vehicles that have been managed

...

The main function where the simulation steps are processed.

intersection\_approach\_distance: A threshold distance to determine if a vehicle is approaching an intersection.

handled\_vehicles: A set to keep track of vehicles that have already been managed at the intersection.

Main Loop

Within run(), the script executes a loop that processes each simulation step:

python

Copy code

while traci.simulation.getMinExpectedNumber() > 0:

traci.simulationStep()

...

This loop continues as long as there are vehicles expected in the simulation.

traci.simulationStep(): Advances the simulation by one step.

Identifying Vehicles at Intersections

python

Copy code

vehicles\_at\_intersection = set()

for lane in traci.lane.getIDList():

vehicles = traci.lane.getLastStepVehicleIDs(lane)

for vid in vehicles:

...

The script iterates over all lanes and checks the positions of vehicles to see if they are near an intersection.

Handling Emergency and Non-Emergency Vehicles

python

Copy code

for vid in vehicles\_at\_intersection:

if is\_emergency\_vehicle(vid):

...

elif vid not in handled\_vehicles:

...

Emergency vehicles are immediately given the highest priority and allowed to proceed at their maximum speed.

Non-emergency vehicles are temporarily stopped when they approach the intersection.

FIFO Implementation

python

Copy code

if len(handled\_vehicles) > 0:

first\_vehicle = min(handled\_vehicles, key=lambda v: traci.vehicle.getLanePosition(v))

...

The script identifies the first vehicle (in FIFO order) that approached the intersection and allows it to proceed.

Other vehicles remain stopped until it is their turn (based on FIFO).

Running the Simulation

python

Copy code

if \_\_name\_\_ == "\_\_main\_\_":

sumoCmd = ["sumo-gui", "-c", "yourConfig.sumocfg"]

traci.start(sumoCmd)

run()

This part of the script starts the simulation using the SUMO GUI and a specified configuration file (yourConfig.sumocfg).

The run() function is called to apply the custom intersection management logic.