U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

MULTI-AGENT TRAFFIC CONTROL SIMULATION

Introduction to Intelligent and Autonomous Systems

DEVELOPED BY
Catarina Monteiro - 202105279
Gonçalo Monteiro - 202105821
Lara Sousa - 202109782

november 21st 2023

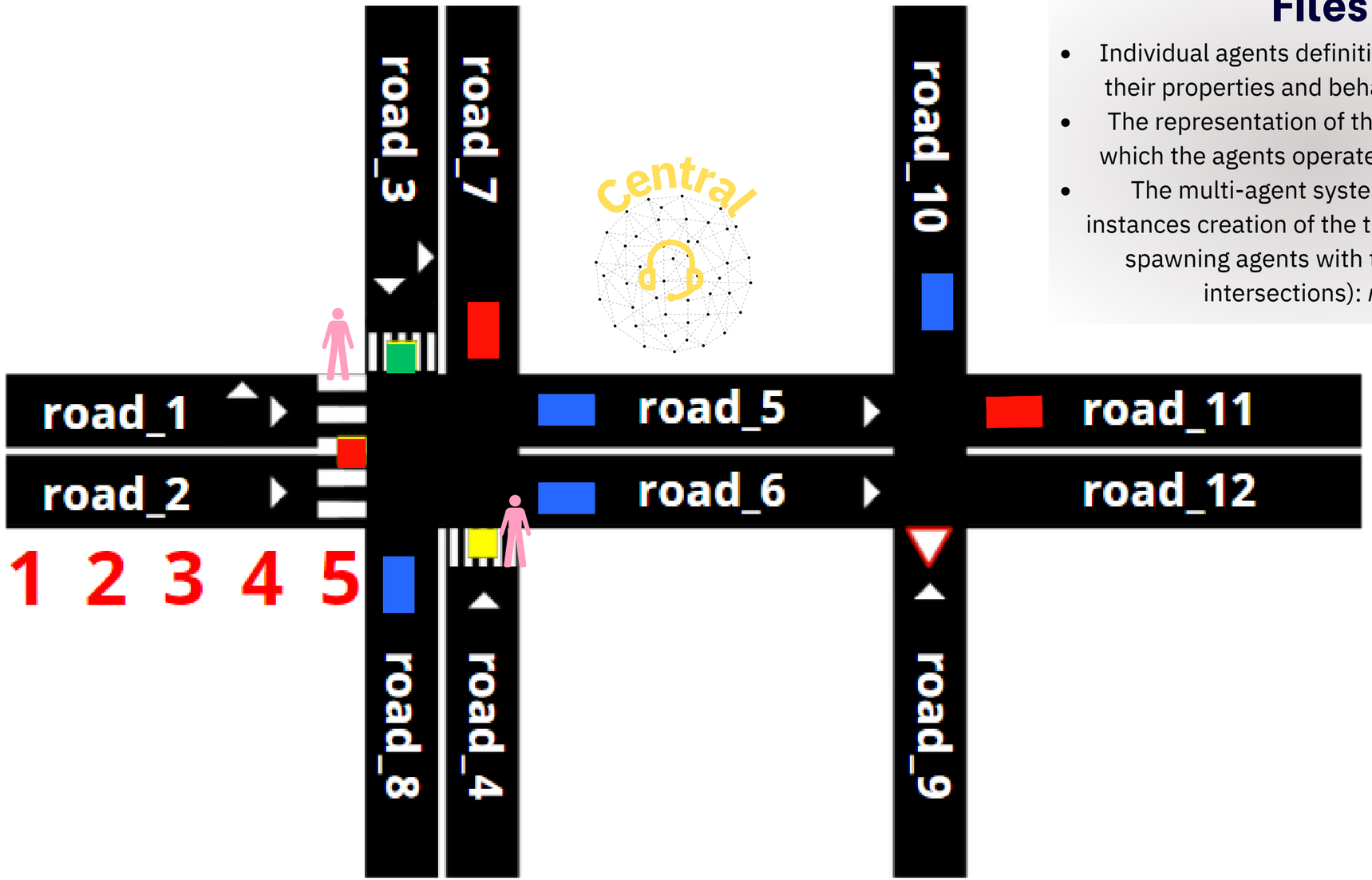
Overview

- The aim is to design and implement a multi-agent system to efficiently manage traffic across multiple intersections. The agents should communicate with each other and coordinate to manage traffic signals, aiming to minimize waiting times and enhance traffic flow.

Objectives

1. Implement intelligent traffic light agents that can adapt to real-time traffic conditions.
2. Usage of vehicle agents that generate simplified but realistic traffic patterns.
3. Integration of the refered agents to work in a coordinated way to optimize traffic flow.

Overall sketch of the environment and agents representation



Files

- Individual agents definition, encapsulating their properties and behaviors: *agents.py*;
- The representation of the environment in which the agents operate: *environment.py*;
- The multi-agent system initialisation(instances creation of the traffic environment, spawning agents with their respective intersections): *main.py*.

Agents

- Three Traffic Light Agents;
- Two Ambulance Agents;
- Four Car Agents;
- Two Pedestrian Agents;
- A Central Agent.

Caption

- Traffic Light 1 Agent
- Traffic Light 2 Agent
- Traffic Light 3 Agent
- 1 2 3 4 5 Possible positions on a road
- zebra-crossing
- Directions
- Priority
- Car agent
- Ambulance agent
- Person agent

Interaction and communication protocols

Spade

(Smart Python Agent Development Environment)

- **Communication Library**

  Employed for communication and coordination among agents.

- **Communication between agents**

There isn't a specific "made protocol." Instead, the communication between agents is facilitated by the SPADE.

Message

Different messages types are employed

- **Inform**

  Used to share information among agents.

- **Request**

  Used to request actions or information from other agents.

Asynchronous

Communication

- **Asynchronous Messaging**

  Agents utilize asynchronous communication, ensuring non-blocking interactions.

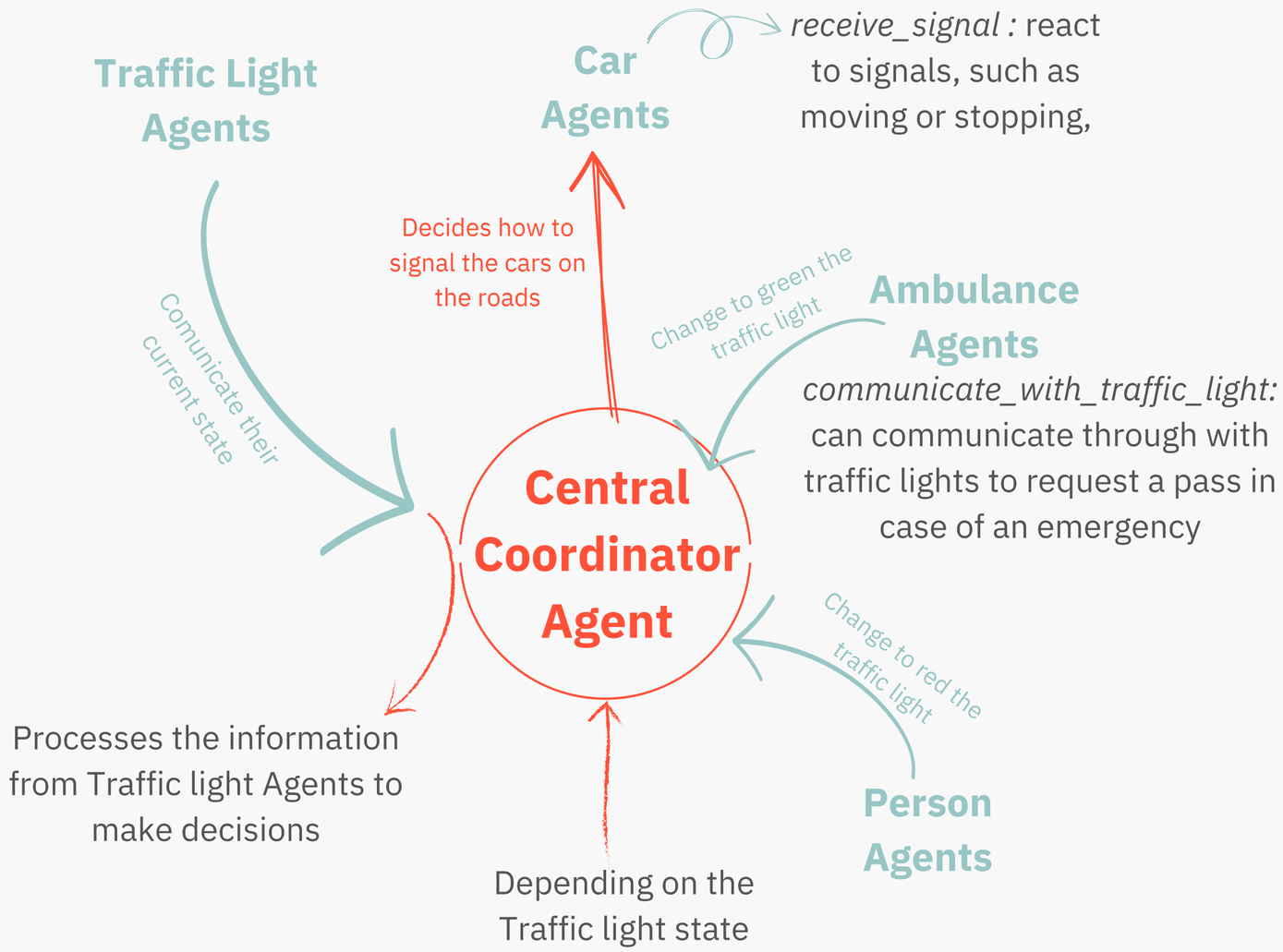
Timeout Handling

- **Message reception includes timeout handling**

  Enhancing the system's responsiveness.

Strategies

Communication Strategy in the environment:



Messages strategy

- **Message Ordering and Concurrency Handling:** Places the messages into a queue;
- **Prevent Race Conditions:** Use the Lock;

Ambulance and priority management

- **Identify and respond to emergency requests:** Directing traffic lights to facilitate quick and safe passage; Decide the appropriate action, like changing a traffic light's color or instructing a car to stop;

Provide Safety on the traffic

- **Elaborate on the safety measures** the central agent enacts, such as stopping all cars on a road for an ambulance;

Dynamic Response to Situations

- Dynamically **responds to real-time situations**, such as an emergency or a pedestrian waiting to cross, and changes the system's behavior accordingly;

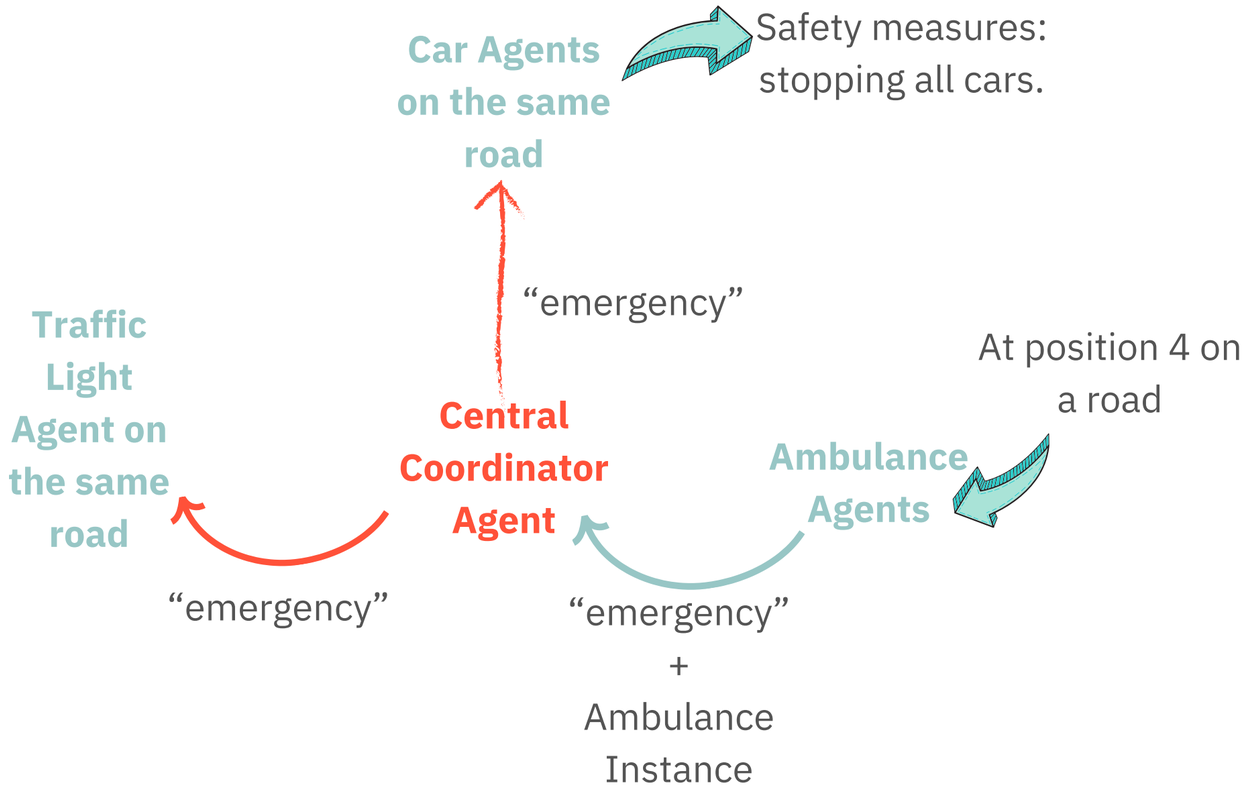
Simulation Environment Interaction

- **Interacts with the simulation environment** to obtain the current state of traffic lights and the location of cars and ambulances.

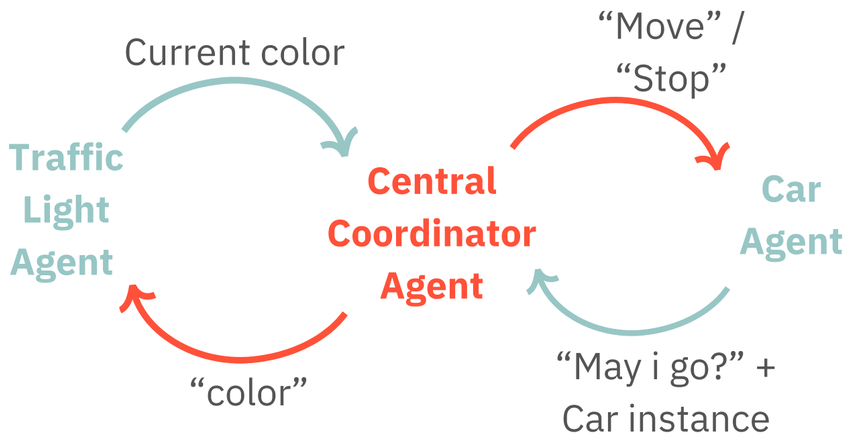
Agents architecture

How does Central Agent coordinate messages?

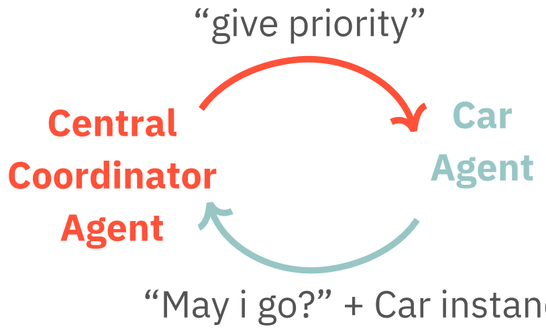
Priority Management: Emergency Communication



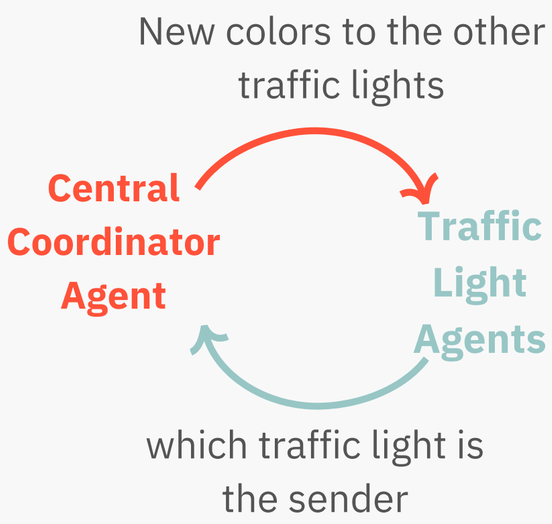
Case 1: If existsTraffic Light Agent on the same road where the Car Agent is

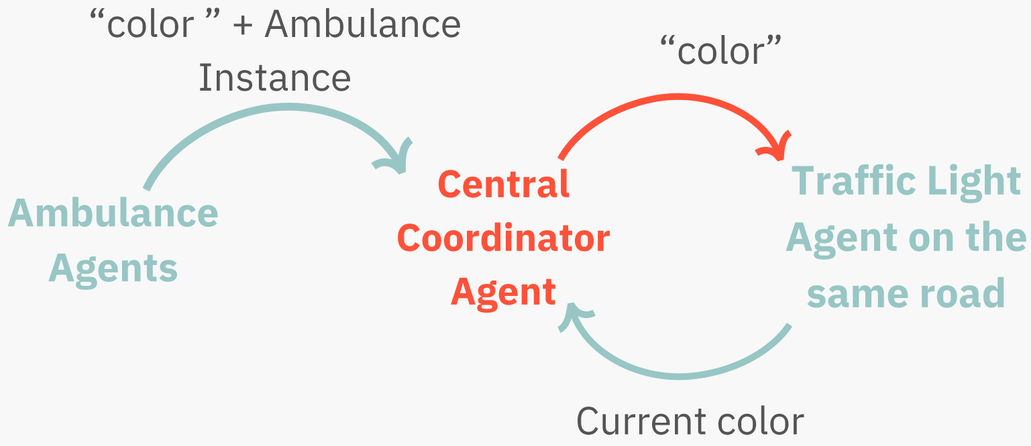


Case 2: If not existsTraffic Light Agent on the same road where the Car Agent is

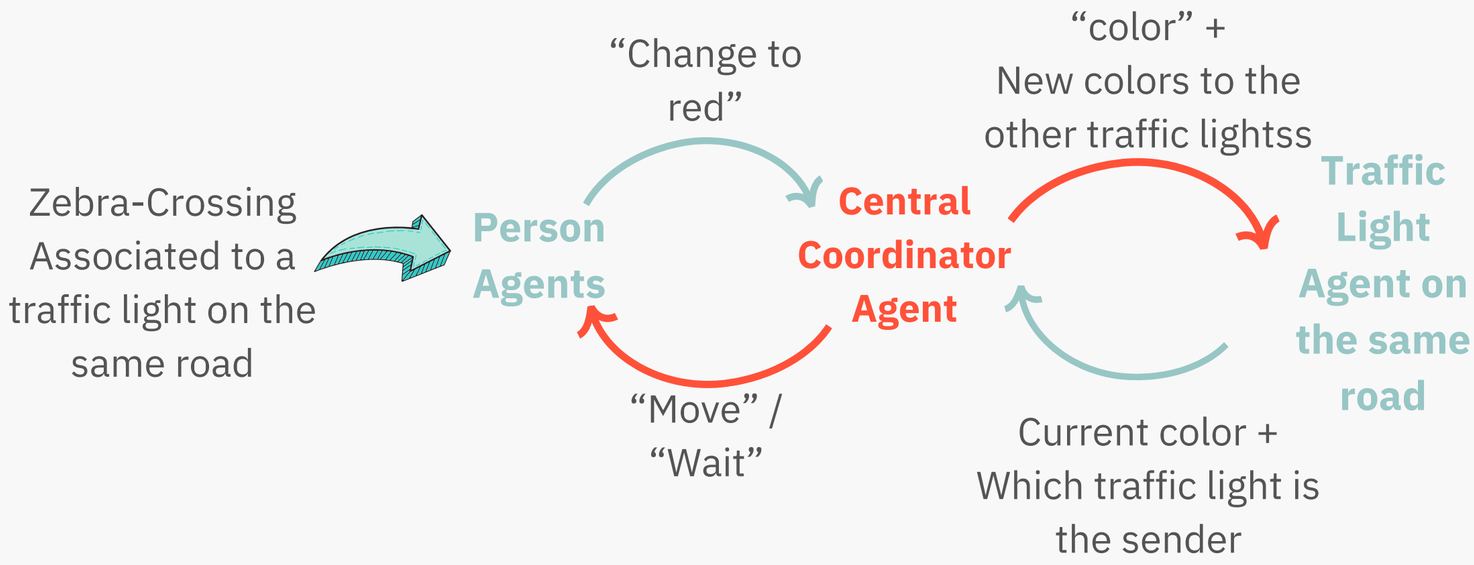


Change to green due to an Emergency





Change to red due to a Person



Experimental part and Analysis of results

How do we visualize the environment and the agent's behavior

Examples:

Here we can observe a traffic light reacting to an ambulance emergency

Here we can see traffic lights reacting to the request of a pedestrian who wants to cross the pedestrian crossing

```
ambulance_2: Moving to position 2.
Traffic_Light_2 is now yellow due to an emergency
Traffic_Light_2 is now green due to an emergency
ambulance_1: Moving to position 3.
Traffic_Light_1 is now red due to the zebra crossing
Traffic_Light_3 is now red due to the zebra crossing
ambulance_2: Moving to position 3.
car_1: Moving to position 3 on road_1
Traffic_Light_1 is now yellow due to an emergency
car_2: Moving to position 3 on road_2
Traffic_Light_3 is now green due to an emergency
car_3: Moving to position 3 on road_3
car_4: Moving to position 3 on road_4
ambulance_1: Moving to position 4.
person_1 approaching a zebra crossing on road_1.
person_2 approaching a zebra crossing on road_4.
ambulance_2: Moving to position 4.
car_1: Moving to position 4 on road_1
Changing Traffic_Light_1 to red.
Central Agent responded with: move. person_1 crossing the street.
Changing Traffic_Light_3 to red.
Central Agent responded with: move. person_2 crossing the street.
car_2: Moving to position 4 on road_2
car_3: Moving to position 4 on road_3
car_4: Moving to position 4 on road_4
ambulance_1: Moving to position 5.
Traffic_Light_1 is now red due to the zebra crossing
```

Experimental part and Analysis of results

We have a problem...

```
ambulance_1: Moving to position 4 on road_1.
mensagem recebida pela central:  <message to="central@localhost" from="ambulance1@localhost" thread="None" metadata={'performative': 'inform'}>
color
</message>
ambulance_2: Moving to position 4 on road_4.
mensagem recebida pela central:  <message to="central@localhost" from="ambulance2@localhost" thread="None" metadata={'performative': 'inform'}>
color
</message>
mensagem processada pela central: <message to="central@localhost" from="ambulance1@localhost" thread="None" metadata={'performative': 'inform'}>
color
</message>
ambulance_1: no loop.
ambulance_2: no loop.
mensagem processada pela central: <message to="central@localhost" from="ambulance2@localhost" thread="None" metadata={'performative': 'inform'}>
color
</message>
ambulance_1: no loop.
ambulance_1: depois do loop.
ambulance_1: if das cores.
ambulance_1:emergency enviada
ambulance_2: no loop.
ambulance_2: depois do loop.
ambulance_2: if das cores.
ambulance_2:emergency enviada
car_2: Moving to position 3 on road_2.
car_3: Moving to position 3 on road_3.
mensagem recebida pela central:  <message to="central@localhost" from="ambulance2@localhost" thread="None" metadata={'performative': 'inform'}>
emergency
</message>
mensagem processada pela central: <message to="central@localhost" from="ambulance2@localhost" thread="None" metadata={'performative': 'inform'}>
emergency
</message>
```

Conclusions

The work developed presents a robust and well-structured solution to the proposed problem.

Critical Analysis

- An error was found in the behavior of ambulance_1, however the system is responsive to emergencies from ambulance_2, as indicated by traffic lights changing to green;
- The system is also responsive to pedestrians crossing the street, as indicated by traffic lights changing to red;
- The system is active and making real-time decisions based on the inputs it receives, traffic light agents are shown to change states, and Car agents are moving to different positions.

Future work

- To solve our error? Directing attention to the central control logic and its handling of individual ambulances. Further investigation into the decision-making process for ambulances prioritisation and the communication protocol between the central agent and the individual traffic lights would be needed;
- To improve the traffic flow? Implementing a more advanced algorithm that predicts traffic patterns;
- To help the system to make informed decidions? Introducing machine learning;
- To adapt the traffic signals to current conditions more effectively? Considering the integration of real-time traffic and weather data.