Smart City Simulation using JADE

Vasileios Savvas Papagrigoriou
vpapagr@csd.auth.gr
M.Sc. A.I. AUTH

I. Main Function Overview

The Main function in the smart city simulation initializes and manages a complex multi-agent system using JADE. It sets up the simulation environment, deploys various types of agents, and ensures all components interact smoothly within a simulated cityscape.

A. System Configuration

The function starts by setting up the city map and user interface. This configuration defines the space where agents operate, influencing their behaviors and interactions.

B. Agent Container Creation

A key part of the Main function is creating an agent container, which is a JADE runtime environment hosting all agents. The function ensures the container is created correctly to avoid any errors that could affect the simulation.

C. Agent Deployment

Once the agent container is ready, the Main function deploys various agents such as citizens, hospitals, nurses, police, and thieves. Each agent has specific roles and responsibilities within the smart city framework.

D. Dynamic Agent Initialization

Agents are dynamically generated based on predefined roles and distributed across the city map, allowing for scalability and flexibility in the simulation.

E. Concurrency Management

Managing concurrency is crucial in a multi-agent system. The Main function uses a CountDownLatch mechanism to synchronize the initialization of all agents, ensuring they are fully operational before the simulation starts.

F. Simulation Readiness

The final step is to ensure all agents are initialized and ready to perform their tasks. This involves waiting for all agents to signal their readiness before the simulation can begin.

II. CityMap Class Analysis

The CityMap class is responsible for creating and managing the virtual city environment. It uses a two-dimensional array to represent the city grid and includes a traffic management system for agent movements.

A. Map Initialization and Configuration

The CityMap class initializes a square grid divided into blocks containing roads, sidewalks, and buildings, providing a realistic urban layout. This structured approach allows agents to navigate the city effectively.

B. Building and Road Layout

The setupBlock method defines each city block, placing roads and sidewalks to ensure connectivity and accessibility. Buildings are placed randomly within each block, with special consideration for essential services like hospitals and police stations.

C. Agent Navigation and Task Execution

Agents use the city map to perform their tasks. For example, police and emergency service agents find the shortest paths to incidents using the map's layout. The getDistance method calculates the distance between two points, helping agents determine the quickest route.

D. Scalability and Customization

CityMap's design supports scalability and customization, allowing adjustments in city size and block configuration without major changes to the agents' logic. This flexibility is useful for testing different urban scenarios and agent densities.

III. MapFrame Class Analysis

The MapFrame class is part of the user interface, extending JFrame to visualize the city environment defined by the CityMap. It shows the layout of the city, including roads, buildings, and agent positions.

A. Visualization Capabilities

MapFrame uses Java Swing to render the city grid dynamically, displaying roads, buildings, and agents. Each tile on the grid represents a city block type, colored appropriately for visual distinction.

B. Agent Movement and Interaction

Agent positions are updated on the map in real-time, showing their current locations. This functionality is crucial for tracking agent movements and interactions within the city.

C. Traffic and Path Drawing

MapFrame illustrates traffic patterns and the paths agents take. Methods like drawPath visually represent the routes agents follow, useful for simulations involving navigation challenges.

D. Customizable Graphics for Roads and Directions

The class includes methods such as drawArrows to depict road directions, helping simulate traffic rules. This feature is vital for realistic traffic simulations, where agents must follow specific road rules.

E. Scalability and Extensibility

MapFrame supports scalability in visual representation. The tile size can be adjusted to represent larger or more detailed areas, making it suitable for different scales of simulation.

IV. StaticColors Class Analysis

The StaticColors class manages the colors for various elements in the smart city simulation, serving as a dictionary where keys represent different entity types and values are corresponding Color objects.

A. Design and Implementation

The StaticColors class uses a HashMap to store color values for specific keys representing various entities, such as roads, sidewalks, and agents. By centralizing color information, the class ensures consistent application of the UI theme across the application.

B. Utility and Flexibility

- **Ease of Use:** Other components can retrieve color settings via a simple method call, improving code readability and maintainability.
- **Flexibility:** New color settings can be easily added or modified in a single location without altering the broader codebase.

C. Consistency and Standardization

Predefined color codes for different entity types ensure consistent visual representation throughout the application, crucial for usability and professional appearance.

D. Methodology

The class provides a getColor method that accepts a key and returns the corresponding color. If the key does not exist, a default color (Color.BLACK) is returned, preventing runtime errors due to missing color definitions.

V. AgentRegistry Class Analysis

The AgentRegistry class manages the lifecycle and accessibility of agents within the smart city simulation. It simplifies agent interactions by providing methods to register, deregister, and retrieve agent details using unique identifiers.

A. Agent Registration

Agents are registered with the AgentRegistry upon creation, associating each agent with a unique AID (Agent ID) and a human-readable name. This process is vital for efficient lookup and interaction among agents.

B. Agent Lookup

The AgentRegistry offers methods to retrieve agents by their AID or name, enhancing system flexibility. This dual method approach allows agents to be referenced in a manner most suited to the context.

C. Deregistration of Agents

To maintain system integrity, agents must be deregistered when no longer active. The AgentRegistry removes agents from the registry, ensuring efficient resource management and an up-to-date system state.

VI. CitizenAgent Class Analysis

The CitizenAgent class extends the JADE framework's Agent class, simulating the behaviors and interactions of a citizen in the smart city. It encapsulates various behaviors and states relevant to daily life, from mobility to handling emergencies.

A. Initialization and Setup

Each CitizenAgent is assigned a home location, potential vehicle ownership, and initial financial resources upon initialization. The agent is integrated within the city's graphical representation through the MapFrame.

B. Agent Mobility and Pathfinding

The CitizenAgent uses a pathfinding algorithm (A*) to navigate the city, calculating paths based on the city layout. The class handles different transportation modes—walking and driving—ensuring with respect to movement rules.

C. Daily Activities and Behavioral Simulation

The agent's daily activities are simulated through a ticker behavior, which randomly determines scenarios such as getting injured. These events trigger decision-making processes, such as seeking medical help.

D. Interaction with Healthcare Facilities

When injured, the CitizenAgent seeks the nearest hospital and requests assistance, involving interactions with hospital and nurse agents.

E. Emergency Handling and Recovery

The agent's behavior in emergencies includes requesting help, waiting for hospital responses, and receiving treatment, ensuring the agent can recover and resume normal activities.

F. Agent Communication and Service Discovery

Communication with other agents is facilitated through JADE's messaging system. The CitizenAgent uses service discovery to locate hospitals, illustrating dependency on city infrastructure.

G. Customization and Extension

The CitizenAgent class is designed to be extensible, allowing integration of additional behaviors such as interaction with public services or customization of daily routines.

VII. Agent-Specific Functionalities

This section explores the specialized functionalities of key agent types within the smart city simulation: the Citizen, NurseAgent, PoliceAgent, and ThiefAgent.

A. Citizen Agent

1) Description and Role: The Citizen class represents the general populace. Citizens report crimes or medical emergencies and have attributes like fear influencing their interactions.

2) Specialized Behaviors: Citizens can observe and report criminal activities, with the likelihood of reporting influenced by their fear level.

B. Nurse Agent

1) Description and Role: NurseAgents respond to medical emergencies, navigating the city to reach citizens in distress.

2) Specialized Behaviors:

- **Task Requesting:** Nurses periodically check with hospitals for new tasks.
- **Task Handling:** Nurses evaluate task urgency and reward, deciding whether to accept tasks based on their "greediness."

C. Police Agent

1) Description and Role: PoliceAgents enforce law and order, responding to crime reports, patrolling, and apprehending suspects.

2) Specialized Behaviors:

- **Patrolling:** Police agents patrol the city to detect and deter criminal activity.
- **Crime Response:** Police agents respond to crime reports and coordinate with other units.
- **Interaction with Citizens:** Police agents interact with citizens during incidents and community policing efforts.

D. Thief Agent

1) Description and Role: The ThiefAgent engages in stealing from citizens and avoiding capture, providing challenges to police and citizens.

2) Specialized Behaviors:

- **Stealing Money:** Thieves check for targets and attempt thefts based on a probability.
- **Avoiding Capture:** Thieves use evasive maneuvers to escape police.

E. Communication and Coordination

Agents communicate using ACL messages, allowing them to send and receive information about events, tasks, and statuses. This is critical for task assignment and emergency responses.

VIII. Critical Infrastructure Agents

This section explores the Hospital and PoliceStation agents, crucial to healthcare and security services in the smart city simulation.

A. Hospital Agent

The Hospital agent manages healthcare services, responding to emergencies and coordinating nurse agents.

1) Operational Setup: The Hospital agent configures its settings, including location and interaction mechanisms, and registers in the Directory Facilitator under the hospital service type.

2) Task Management: The agent manages and prioritizes medical emergencies through a task queue system, processing requests for assistance.

3) Nurse Coordination: The sendTasksToNurses behavior dispatches tasks to available nurse agents based on severity and proximity.

4) Response Handling: The ManageTaskResponses behavior tracks nurse responses, ensuring all emergencies are addressed promptly.

B. Police Station Agent

The PoliceStation agent manages public safety, handling security incidents and coordinating police activities.

1) Initialization and Setup: The PoliceStation agent initializes with specific parameters and registers in the Directory Facilitator as a PoliceStation.

2) Security Task Management: The agent receives and manages security tasks through its casedReportListener, adding incoming tasks to a queue.

3) Police Agent Coordination: Tasks are distributed to police agents through the sendTasksToPoliceAgents behavior, based on proximity and urgency.

4) Task Response Management: The ManageTaskResponses behavior monitors police responses, ensuring effective resolution of incidents.

C. Inter-agent Communication and Coordination

Both agents use ACL messages for standardized communication, ensuring effective coordination and transparency.

IX. Proposed Future Improvements

Future improvements aim to enrich interactions and capabilities in the smart city simulation.

A. Citizen

- **Role Diversification:** Allow citizens to perform dual roles, such as being both regular citizens and potential thieves, for more complex interactions.
- **Deceptive Behavior:** Enable citizens to lie to police to gain advantages, adding trust and verification dynamics.

B. City Infrastructure

- **Traffic Management:** Implement traffic lights and a more robust traffic system for realistic urban traffic scenarios.

C. Police

- **Advanced Coordination:** Develop systems for police agents to coordinate responses, especially in criminal pursuits or large public events.
- **Traffic Control:** Allow police to manage traffic dynamically, controlling lights during emergencies to optimize routes.

D. Hospital

Allow hospitals to delegate tasks to other hospitals or emergency services to distribute responses and avoid bottlenecks.

E. Theft and Security

Enable thieves to form gangs and perform organized crimes, requiring coordinated police responses.

X. Overview of ACL Messages

A. Types of ACL Messages

- **INFORM:** Used to notify other agents about certain events or states. For example, the ReceiveHealingConfirmation behavior in CitizenAgent uses INFORM to notify that healing has occurred.
- **REQUEST:** Utilized when an agent needs a particular service or response from another agent. Citizens use REQUEST messages to alert police stations in the event of robberies.
- **PROPOSE:** This message is sent to propose a specific course of action. Hospitals use PROPOSE messages when assigning tasks to NurseAgents, as seen in the HandleTaskResponses behavior.
- **REFUSE:** Sent in response to proposals or requests when the agent cannot comply. An example is when a NurseAgent refuses a task due to high workload.
- **ACCEPT_PROPOSAL:** Confirms willingness to undertake the proposed task. NurseAgents send an ACCEPT_PROPOSAL message when agreeing to take on a task.
- **REJECT_PROPOSAL:** Used to formally decline a proposal. Police agents might send REJECT_PROPOSAL if they are already committed to a more urgent task.

XI. Analysis of Agent Behaviors

A. CyclicBehaviour

- **Usage:** This behavior runs continuously and is ideal for tasks that require constant monitoring or listening.
- **Examples:** - casedReportListener in the PoliceStation agent continuously listens for emergency calls from citizens. - hearingCopVoices in ThiefAgent listens for police alerts to evade capture.

B. TickerBehaviour

- **Usage:** Executes actions at regular intervals, suitable for tasks requiring periodic execution.
- **Examples:** balanceCheck in Citizen, which checks for financial changes at regular intervals. RequestTasksFromHospital in NurseAgent, requesting new assignments periodically.

C. WakerBehaviour

- **Usage:** Activates once after a specified delay, used for tasks that need to be executed only once but after a delay.
- **Examples:** In PoliceAgent, WakerBehaviour might be used to delay the response to a non-urgent call, allowing the agent to prioritize more urgent tasks.