Chapter 3

Multiagent Systems in Automotive Applications

Raul Campos-Rodriguez, Luis Gonzalez-Jimenez,
Francisco Cervantes-Alvarez,
Francisco Amezcua-Garcia and
Miguel Fernandez-Garcia

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.69687

Abstract

The multiagent systems have proved to be a useful tool in the design of solutions to problems of distributed nature. In a distributed system, it is possible that the data, the control actions or even both, be distributed. The concept of agent is a suitable notion for capturing situations where the global knowledge about the status of a system is complex or even impossible to acquire in a single entity. In automotive applications, there exist a great number of scenarios of distributed nature, such as the traffic coordination, routes load balancing problems, traffic negotiation among the infrastructure and cars, to mention a few. Even more, the autonomous driving features of the new generation of cars will require the new methods of car to car communication, car to infrastructure negotiation, and even infrastructure to infrastructure communication. This chapter proposes the application of multiagent system techniques to some problems in the automotive field.

Keywords: multiagent systems, automotive applications, traffic coordination, automobile negotiation, car-2-X communication

1. Introduction

One of the primary goals of the artificial intelligence field remains open; this is the development of autonomous systems capable of performing self-directed tasks in a similar way that humans do. Challenges and issues involved in the development of autonomous systems deployable in dynamic and open environments have led to fields as multiagent systems [1]. It is a discipline that forms a profound interdisciplinary study of fundamentals such as autonomy, agency, negotiation, communication, interaction, and cooperation. The major objective of this field is to develop autonomous systems capable of coexisting and cooperating with people and other

systems in the real world. The principal motivation of this effort to develop autonomous systems is related to how people live in a digital and interconnected world, where new challenges and opportunities are arising (e.g., Internet of Things (IoT), smart cities, and big data [2]) as consequence of technology is strongly embedded in our daily life. Thus, we are near to see in our local environment, autonomous systems like smart environments (rural and urban scenarios), humanoid robots, unmanned vehicles (aerial and ground), among other autonomous systems capable of supporting people in their daily life. An important feature of these systems is the autonomy because they must be capable of embodying self-governance and decision-making. In this sense, to ensure that the autonomous systems are useful, they should be endowed with the ability to exhibit a smart negotiation to achieve its goals through the cooperation. It is supposed that these properties enable distributed systems to improve their performance.

Negotiation enables multiagent systems to achieve their goals. Although there are several research achievements that concern to strategies and protocols in the field of negotiation nowadays, its implementation in applications in real world scenarios is still far to reach. In a general sense, the multiagent system (MAS) is a paradigm in the computer sciences and related areas where a system of interest is conceived as a set of autonomous entities called agents, as well as its interaction mechanisms. The agent is an autonomous entity with the ability to "sense" the environment through a set of physical or logical sensors and to "interact" or "modify" such an environment by a set of physical or logical actuators, as well. A kind of "intelligence" or "inference" mechanism is also conferred to an agent. Thus, actions to the environment are based on the sensors and the inference machinery.

1.1. Multiagent systems

The MAS approach has proved to be a suitable solution for problems of distributed nature, where the information, the control, the processing, or all of them are not centralized but rather distributed. Thus, a set of problems has been well studied and useful solutions have been obtained. The interaction among agents is generally considered as message passing based on a well-structured interaction protocols. The content of the message is "information" that may lay in a context called ontology. **Figure 1** depicts a general layout of a MAS accordingly Foundation for Intelligent Physical Agents (FIPA) [3, 4].

The Foundation for Intelligent Physical Agents (FIPA) is an IEEE organization promoting the technology and standardization of multiagent systems. FIPA defines a set of specifications in the basic layer for the agent communication, management, and message transportation, as well as specification for the abstract architecture and applications layers. The interaction protocols, communicative acts, and the content of the messages interchanged between the agents are covered by the specifications defined by FIPA. For example, the auction and call for proposal mechanisms among a set of agents are defined as interaction protocols in the FIPA specifications [5].

1.2. Automobile applications

The automotive industry is moving toward the automated mobility. To achieve the goal of making mobility safer and having an optimized system for moving people in the world, a visionary technology is needed. The approach followed in this chapter is based on MAS

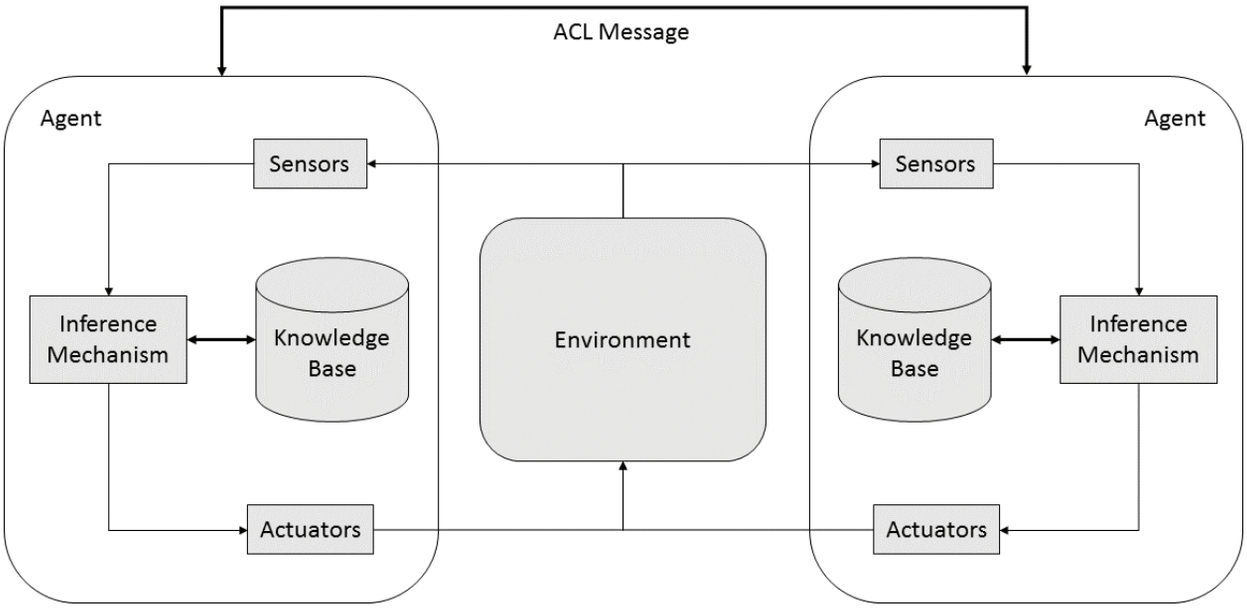

Figure 1. General layout of a multiagent system.

applied to the automotive scenarios [6]. The internet of things (IoT) is part of the design, as it is a trending technology for the connected cars and smart cities.

The applications of MAS to automotive applications, like traffic management and load balancing problem, include multiple possibilities, since the agents could represent different actors in the implementation of the solution. For example, in [6], the authors identify five types of agents: pedestrians, vehicles, traffic lights, streets, and parking lots. In this chapter, we consider the use of coordinators, route agents and traffic light cycles (phases), as an extension to the entities involved in the traffic manipulation.

The use of pedestrians as agents suffers from the problem to manage the communication with other agents. For example, other agents like vehicles and traffic lights can be incorporated with electric source and wireless link that helps power sensor systems or technology to help them accomplish that purpose of communication. Though, pedestrians normally do not have the facilities to perform those functions, however, the benefits to consider pedestrians as agents can be substantial due to obvious reasons. An approach to incorporate pedestrians into the system is to use a mobile device, such as the smart phones. By using these devices to identify pedestrians, its sensors may allow to monitor the position of the pedestrians, among other cases.

Other examples of vehicles as agents are reported in [7] and [8]. These works consider the communication between vehicles to coordinate the routes, every vehicle should take to reach its destiny. Within this approach, every vehicle has information that helps them to accomplish their goal, which deals with moving from point A to point B in the shortest time possible. The agents or cars can share or keep this information according to its heuristics which are the rules they use to make any decision that push them closer to complete their goal. Making local individual decisions based on information gathered by themselves or cooperating with other agents help they accomplish a global goal of coordination between vehicles in such a way that every agent can reach their destiny in less time than picking the common fast routes, and sometimes creating bottlenecks on those streets or avenues.

Another popular approach is to focus on traffic lights since they are typically the most common points where traffic loads are introduced into the system. There are several papers focusing on intersections like [9–12]. These works focus on coordination of phases between different intersections. The hypothesis is that creating local solutions in each intersection will produce a better performance overall in the system as a whole.

In [13] the authors propose using a set of Q-learning iterations to approach the optimal solution of load balancing. They also mentioned several methods to control the traffic lights and intersections using different techniques of the artificial intelligence, such as fuzzy rules, predefined rule-based systems, and centralized methods. An important feature of this approach is that when controlling traffic lights and intersections, the phases that control traffic in different roads are a key element for the success of the goal of the system. Indeed, the coordination between changes of lights and what streets have preference before others are crucial to get a good traffic flow in the right direction. This feature is considered in this chapter.

Other important example that implements multiple agents as a solution to automotive scenarios is [14]. In this work, the focus is to make buses arrive on time to their stops. The system uses four agents: the bus vehicle, the bus route, the intersections, and the stages. The bus vehicle drives through the route informing the route agent their times, the route agent checks the time between the buses in the same route and if the buses are late or early, it communicates with the main agent, the intersection. The intersections analyze what to do; if the bus is to early then the stages where the bus is not currently transiting have priority to be set in the traffic light. On the contrary, if the bus is too late, the stages where the bus is going should have more probability of appearing in the traffic light. One important aspect to notice is the priority, having a greater priority does not mean that automatically that stage will be next. It only gives to the agent more tools to coordinate with other stages to be the one at the top, which is a goal. The stages need to coordinate and from that process, the next stage in the traffic light is selected. The coordination is selected by multiple factors, the number of buses in the lane, the green time required by the stage, the velocity of the vehicles in the lane, etc.

1.3. Technology used

Based on the specifications defined by FIPA, several implementations provide frameworks for the development of MAS. For example, the JAVA Agent Development (JADE) Framework is a platform for the development of agent-based applications. JADE is fully compliant with the FIPA specifications and provides a basic class for agent instantiation, communication protocols, ontology implementation, and graphical management tools. **Figure 2** provides a reference model for the management of agents within the platform [15].

When working with automotive traffic, it is difficult to find a real environment for testing. For example, closing a group of intersections and sending vehicles in a predefine pattern, are desired features for the experimentation process in MAS applied to automotive scenarios. Fortunately, there are some computer traffic simulators that, with some sort of work, could be coupled to MAS development frameworks such as JADE, which is one of the target environments of this chapter.

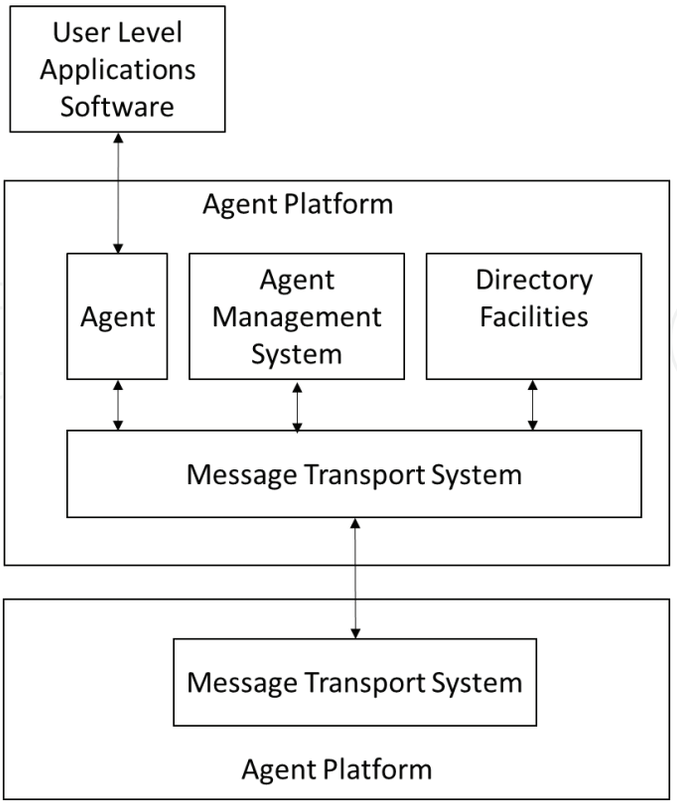

Figure 2. Agent management reference model in JADE.

To mention some simulators, consider for example VISSIM, Paramics, Aimsun, Dynameq, MITSIMLab, Simulation Urban Mobility (SUMO), DRACULA, DynaMIT, MEANET, and MATSim [14]. The simulators provide different characteristics that made them ideal for multiple scenarios. However, SUMO [18] seems to be used more often because it is microscopic, free, and easy to use. This chapter will focus on SUMO to simulate required traffic patterns and to interconnect these results with the JADE development framework in the section devoted to the negotiation and coordination applied to traffic load balancing with the use of intuitive ideas and common sense decisions [19].

SUMO stands for Simulation Urban Mobility, and is an open source project to create a portable traffic simulator. This simulator provides a lot of characteristics that made it ideal for the experiments of the last scenario considered in this chapter. First, the interfaces are visual and easy to use, the way to create routes and export them to be used is very much like a city simulation game. In the interface, multiple lanes can be created for a single street, intersections can be configured to set the phases of the traffic light, and the behaviors that vehicles can perform. SUMO provides an API to manipulate the simulation and obtain information about the same, making it ideal to work with other systems like the JADE framework, which was successfully used in the construction of Multiagent systems [20].

SUMO provides the user with tools to easily represent real streets and roads, then insert into the simulation elements like vehicles, which try to behave as their counter parts in the real world. In this way, the simulations are quicker and cheaper than the real-time events and allow to test the same rules in different environments in a practical way.

Within the SUMO simulator, some designs have been taken to fit with the scenarios required in this chapter. Basically, there are two kinds of simulation processes, macroscopic and microscopic. The macroscopic simulation focuses on the system as a whole. It considers the state of the system at every moment, density, speed, and volume of vehicles. On the other hand, the microscopic simulation focuses on the actions of individual members of the systems. Thus, the approach followed in this chapter is the microscopic simulations, since the actions of the agents can be easily applied to members of the simulation, and within the approach proposed in this chapter, it corresponds to vehicle and infrastructure actions.

The most common simulation scenarios of interaction between agents considered in this chapter are the intersections. Among the two most common intersections where vehicles interact are the crossroad and the T, as depicted in **Figure 3**.

A simulation is composed by several elements but mainly defined by two principal configurations, the network configuration and the traffic demand configuration [11, 12]. This configuration is done through xml files. The network configuration contains multiple components starting with the nodes and edges. A node represents a joint point between edges, while edges represent the roads through which traffic will be circulating. A node is simply a representation of a point in the map that only requires three elements, an identifier and a pair (x, y) of coordinates.

1.4. Contributions of this chapter

This work presents application scenarios that take advantage of the MAS in the automotive field. In this work, the cars and infrastructure devices, like semaphores, are considered to be agents. The agents are communicating with each other by using a wireless network, through the usage of well-structured ACL messages. The agents send messages to know the status of the system, and based on that information, they can make decisions on how to use the available resources, for example, the roads.

In the approach proposed in this chapter, the infrastructure devices have information about routes they are managing. When a vehicle agent requests information about a specific route, the infrastructure device informs the status of the variables of such a route. Once the vehicle

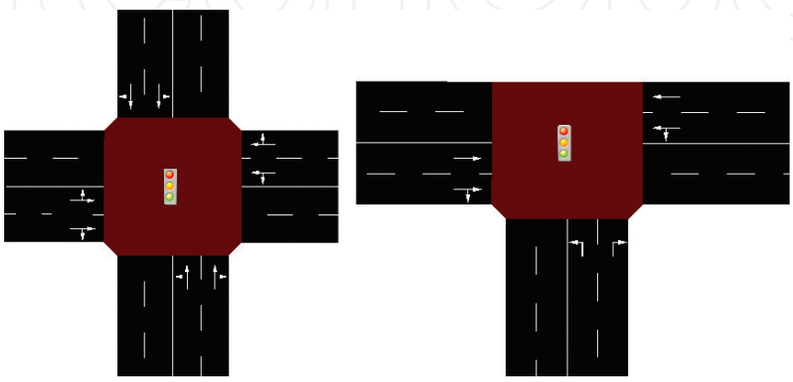

Figure 3. To the left, a crossroad and to the right, a T form intersection.

has the information, it evaluates which route is the best based on its goals and in some cases, the individual agents consider information about the preferences of other agents to get a suitable global solution.

In this way, a cooperative, distributed multiagent system can be used to improve dynamic routing and traffic management. Distributed artificial intelligence techniques, those applicable to MAS, could be used to solve decision-making problems to solve city mobility issues with the new technology cars.

2. QoS approach applied to traffic balancing

The QoS approach considers a method to calculate the best route for a vehicle based on a set of requirements, of the drivers as well of the infrastructure. It was firstly proposed in the telephony and computer network industries to measure the requirements of different users. To quantify the service of the network, several aspects of the service are considered such as the bit rate, mean of errors in the transmissions, throughout, jitter, transmission delays, or availability, among others. In QoS, a weight is assigned to each of the goals of the user, depending on the importance assigned to each aspect of the service they require. Then, a negotiation process is executed between the clients and the service network.

In the context of the automotive field, such interaction helps to find a better route for an agent, or rather the driver it represents. On one hand, as far as information of different routes is shared, the traffic management system (the network) tries to maintain a balanced traffic accordingly to its own goals. On the other hand, the vehicle agents have their own priorities. For example, it could be possible that for a specific type of driver, the distance it will travel is quite important; while for the another one, the number of turns it will make on its travel is the key parameter. Consider, the case of a big cargo truck versus a utilitarian car, for example.

In this approach, the information about traffic is currently used to decide whether to use a certain route or not. However, infrastructure typically does not take part in a system to keep the traffic balanced. It is supposed that the infrastructure could play an important role in the load balancing strategy. In this approach, the infrastructure may consider information about building constructions in certain areas. Thus, an objective of the infrastructure could be to reduce the traffic flow in those areas.

The implementation described in this chapter explains how a distributed system changes the perspective of the traffic in a city, and how important is to see it as part of a smart infrastructure where all agents play an important role. The definition of the objectives of the drivers and the infrastructure play a key role in this approach.

2.1. Goal definition by a utility function

The car agents must define in a quantitative way, the goals and preferences of the drivers they represent. Based on the received information, vehicle agents may calculate the utility as follows:

$$U^{ip} = \sum_{g} W_g^i N_g^{ip} \quad (1)$$

such that:

$$\sum_{g} W_g^i = 100 \quad (2)$$

Where i stands for the $i-th$ agent, p is the specific path, g is the specific goal, U is the overall utility function, W is the weight conferred to specific goal by $i-th$ agent, and N is the normalized score for goal g by $i-th$ agent.

The goals that a vehicle agent considers are based on the driver preferences. For example, but not limited, to the following goals:

a. Minimize Travel Time, g_1

b. Minimize Travel Distance, g_2

c. Minimize/Maximize Arterial Streets, g_3

d. Minimize Number of turns, g_4

e. Minimize/Maximize Roadway classification changes, g_5

Thus, for example, a cargo truck may confer big weight to the number of turns in the selection of its best route, as follows:

$$U^{cargoTruck} = 10\,g_1 + 10\,g_2 + 0\,g_3 + 80\,g_4 + 0\,g_5 \quad (3)$$

In a similar way, the other types of cars can define the preferences of their drivers in the negotiation of the best route based on the QoS approach. For additional information about the goal-based QoS.

2.2. Architectural design

Figure 4 provides a conceptual diagram of the agent interaction proposed in this chapter for the architecture implementing the QoS approach. In the figure, the car agents "request" information about the "status" of the infrastructure is done by asking to the proper agent. With the information of the nearby lanes, the car agents can decide which one provides the best solution for the goals of the driver they represent. The diagram is supported in the JADE framework [15].

2.3. Experimentation

In this approach, the implementation considers the following aspects:

- The number of car agents in the MAS is arbitrary. That is, it could be from two agents, i.e., one car and one infrastructure or route, to an open number of cars and routes.

- The agents of the system are implemented on an embedded board, e.g., the Intel Galileo Board Single hardware **Figure 5**, based in the JADE framework [15] where human decision of driving agents are tried to be programmed algorithmically [16, 17].
- The architecture distinguishes two types of agents: unsteady (e.g., routes) and steady (e.g., cars).
- The architecture considers a load balancing algorithm among the car agents and route agents based on QoS.
- The architecture considers that the route agents shall send their parameters of interest to all the car agents that request them. The parameters of interest are automatically updated in every 1 min.
- The car agents use the information provided by the route agents to calculate its best route.
- The distributed load balancing algorithm considers the infrastructure requirements, for example, to keep some route under some peak value of traffic density.

For illustrative purposes, **Table 1** summarizes the parameters for the experiments in the QoS approach. There are four routes available, each one known by an infrastructure agent. There are two vehicles that would receive information from such routes. According to the MAS, they will be "born" with some attributes that will receive through the arguments, which are described in **Table 1**.

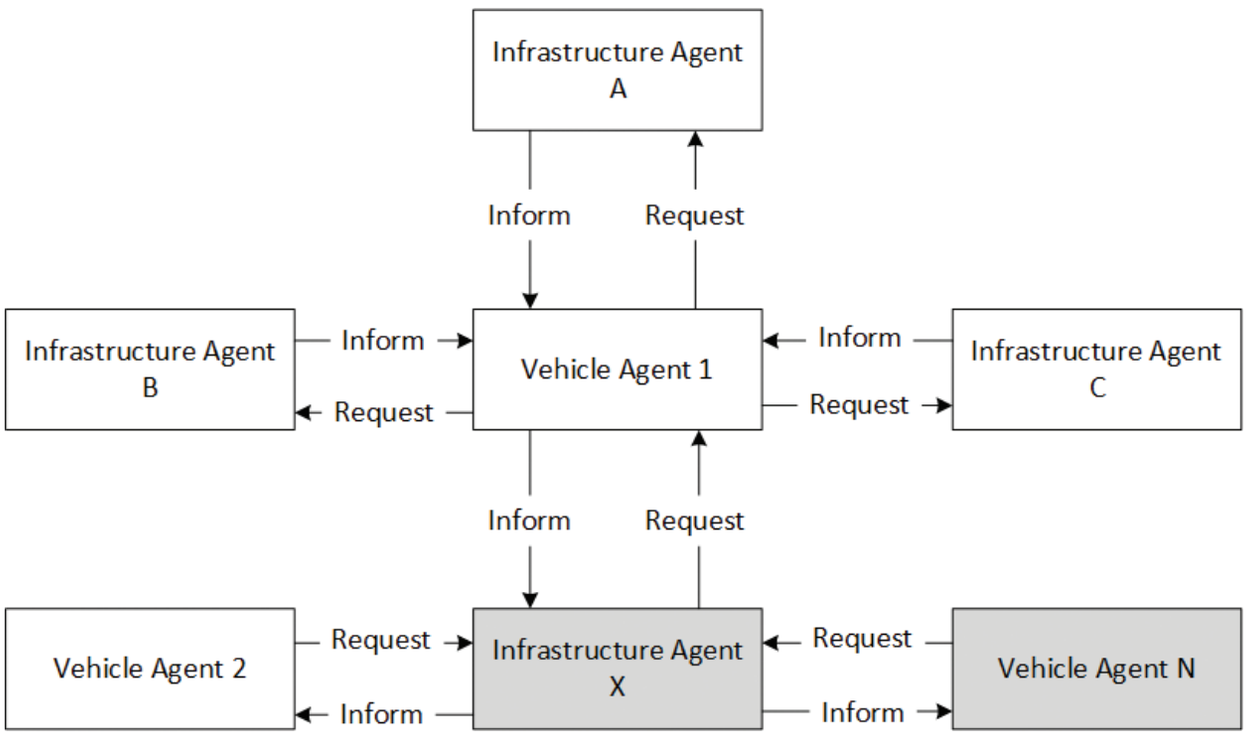

Figure 4. Conceptual diagram of the agent's interaction.

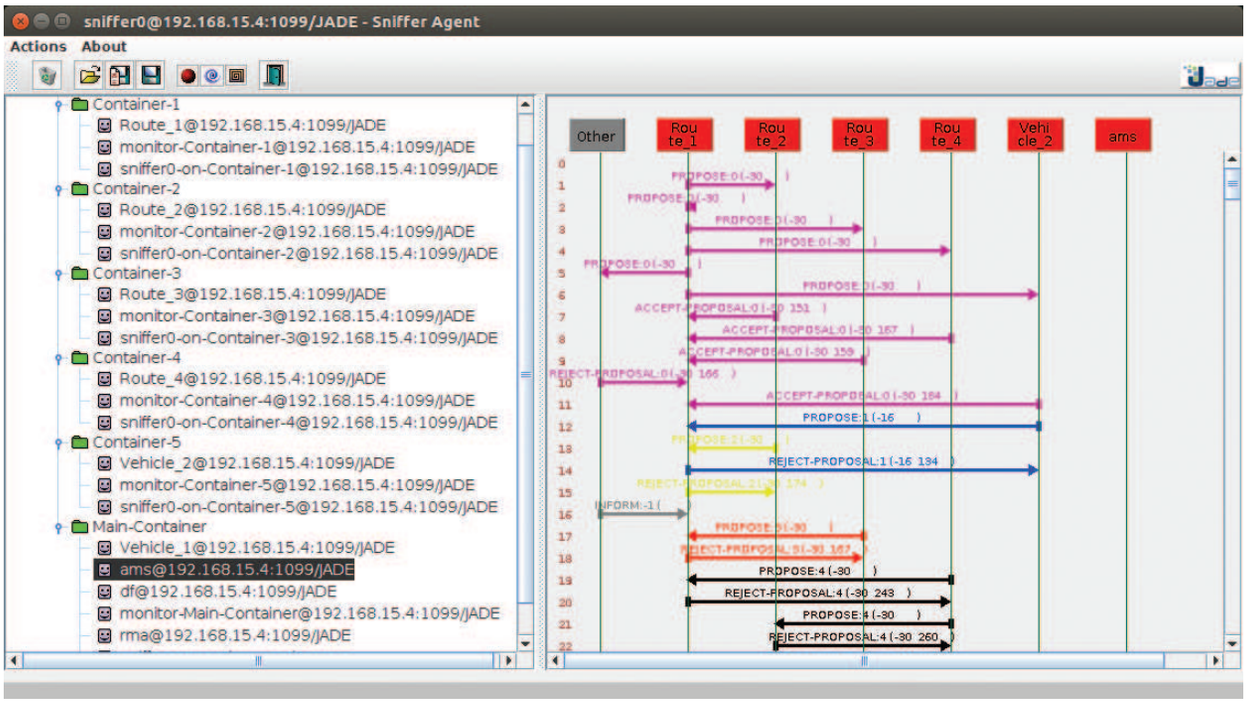

Figure 5. Example of agents' communication.

	Arguments						
Agent name	**Travel time**	**Travel distance**	**Arterial streets**	**Turns**	**Roadway changes**	**Vehicle?**	**Route number**
Route_1	0	0	0	0	0	False	1
Route_2	0	0	0	0	0	False	2
Route_3	0	0	0	0	0	False	3
Route_4	0	0	0	0	0	False	4
Vehicle_1	55	10	5	5	25	True	0
Vehicle_2	5	5	80	5	5	True	0

Table 1. Agents parameters for the scenario.

The first group of five arguments (columns) represents the weight (importance) that each agent gives to that goal. The first group of four agents (rows) represents routes. The routes have a zero value on those arguments because they do not have such goals. Rather, their job is to inform those conditions to the vehicle agents.

The sixth argument, i.e. *Vehicle?*, it is only used to indicate whether the agent is a vehicle or an infrastructure agent, since they share the same base class that the JADE framework provides for every implementation of an agent. Finally, the seventh argument indicates the route number. This value only concerns the infrastructure agents and its objective is to have a unique ID for each route.

Priorities/goals	Vehicle_1 weights (%)	Vehicle_2 weights (%)
Travel time	55	5
Travel distance	10	5
Arterial streets	5	80
Number of turns	5	5
Roadway classification changes	25	5

Table 2. Vehicle weights.

In the current implementation, the vehicle agents ask for the normalized values of each condition every 10 s. As soon as it receives the values of each route, it calculates the best route based on the weights (driver preferences) given when it was born. The vehicle has a list of routes, in case there is a new one, it will add such route to the array list *routes List*, defined as a global variable in the agent class.

The calculations were made analytically to compare against the results computed by the car agents. **Table 2** shows the weights that each agent assigns to each goal.

Tables 3–6 show the selection of the results of the utility of the routes in the experiments. These results agree with the expected values accordingly with the weights of the agents.

Route_1	Vehicle_1 utility	Vehicle_2 utility
0.11	0.0605	0.0055
0.52	0.052	0.026
0.69	0.0345	0.552
0.88	0.044	0.044
0.45	0.1125	0.0225
	0.3035	0.65

Table 3. Route_1 utilities.

Route_2	Vehicle_1 utility	Vehicle_2 utility
0.88	0.484	0.044
0.12	0.012	0.006
0.12	0.006	0.096
0.73	0.0365	0.0365
0.99	0.2475	0.0495
	0.786	0.232

Table 4. Route_2 utilities.

Route_3	Vehicle_1 utility	Vehicle_2 utility
0.23	0.1265	0.0115
0.22	0.022	0.011
0.88	0.044	0.704
0.43	0.0215	0.0215
0.25	0.0625	0.0125
	0.2765	0.7605

Table 5. Route_3 utilities.

Route_4	Vehicle_1 utility	Vehicle_2 utility
0.44	0.242	0.022
0.43	0.043	0.0215
0.19	0.0095	0.152
0.25	0.0125	0.0125
0.81	0.2025	0.0405
	0.5095	0.2485

Table 6. Route_4 utilities.

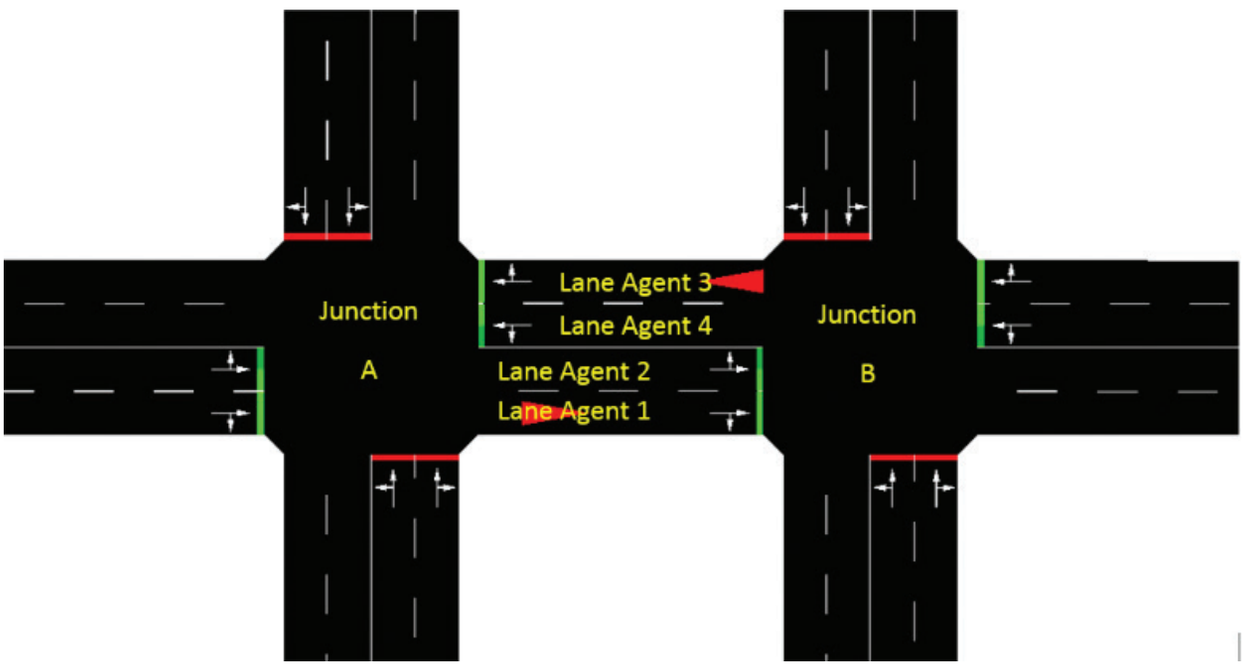

Figure 6. Lane and junctions.

The agents obtain the maximum of all the routes, the result will be the best route. In this case, Route_2 will be the best for Vehicle_1 with a total utility of 0.786 and Route_3 will be the best for Vehicle_2 with a total utility of 0.760. **Figure 6** shows a screenshot of the GUI of the Sniffer agent capturing the ACL messages of the interaction between the cars and routes. The main container with the administrative tools of the JADE platform, including the sniffer agent, is running in a laptop. The agents are running on an Intel Galileo development board.

3. Agent negotiation and coordination

The coordination of agents is a key element in the MAS field. This coordination can be accomplished by using multiple methods. For example, if the agents are competing to obtain a resource, an auction can be a good mechanism.

3.1. Design description

The approach considered in this section is like the one provided in Ref. [21]. However, instead of focusing only in the buses, we will focus on all the vehicles going through an intersection. The proposed design has three main agents: the lane agent, the junction agent, and the phase agent.

3.1.1. Lane agent

The lane agent represents one of the lanes of an edge. More precisely, let say there is a street section that goes from junction A to B, and that street goes in both directions A to B and B to A. Then, the lane agent 1 will be the lane closer to the right in the section that goes from A to B, while the lane agent 2 is the second closest to the right. A similar approach is applied to the section that goes from B to A. The lane closest to the right will be lane agent 3 and finally, the second closest lane is the lane agent 4. This approach could be followed incrementally. That is, the street can have one or more lanes going in the same direction which means that a street can have multiple lane agents assigned to them, as previously described. **Figure 7** provides an illustration of the junctions and lane agents.

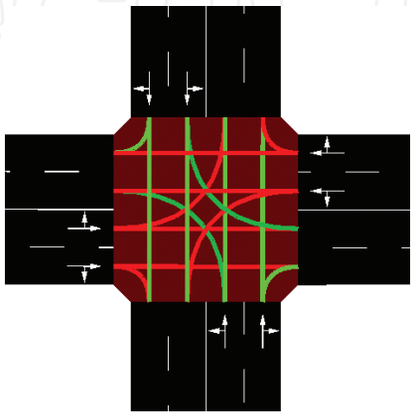

Figure 7. Connections and phase representation.

The objective for the lane agent is to keep the lowest number of automobiles in the street at any time. To accomplish that aim, the lane has a priority that is related with the capacity of the street and how close it is to reach its limit. This limit is when the lane reaches the priority 1, which means the street is almost empty. On the other hand, when the lane is at the priority 5, it means the street is at full capacity.

To calculate the lane capacity, multiple parameter are in play, for example, the length of each vehicle (C), the space between each vehicle (S), the number of vehicles (N), the length of the lane (L), and the maximum number a priority can reach (M). The following equation captures these parameters:

$$P = \frac{\sum_{i=0}^{N}\left(S_i + C_i\right)}{L} M + 1 \tag{4}$$

3.1.2. Junction agent

This represents an intersection in a real scenario, which is a junction between two or more streets, which also may contains a traffic light in it. The objective of this agent is to manage the traffic light cycles, which for the systems are called the phases, in such a way that the streets can allow traffic to move through the intersection. This agent is responsible for keeping the phases in a stack to inform what the current stage is and rotating the phases according to that stack.

3.1.3. Phase agent

This agent represents a traffic light cycle. The objective for this agent is to negotiate with other phases to go up in the stack from the junction agent. The phase has a priority to know what kind of actions it needs to negotiate with other agents and tries to stay as much time as possible at the top of the stack. To accomplish that aim, every phase agent has several seconds that can be used to negotiate with other agents.

A phase contains two arrays of elements, one with the time of the cycle and the other with a string representing the behavior that vehicles can have during that phase. These elements are represented as follows:

[31, 6]

[“GGGgrrrrGGGgrrrr”, “yyygrrrryyygrrrr”]

The array of string represents the behavior of the traffic lights during the cycle. For example, starting from the first lane at the top left in **Figure 6**, the vehicles can turn right in first lane and go straight, in the second lane. The same vehicles can go straight and turn right with precaution (this represents the lowercase g in the above character string). All the red lines in **Figure 7** represent connections that vehicles cannot use during this phase.

3.2. Agent coordination in QoS

This process starts when the lane agent calculates its priority. This may happen every certain quantum of time depending on the configuration of the system. The lane agent calculates its priority by checking the lane capacity with the formula seen in section describing the lane agent. That calculation returns the priority level the lane should have and if it is different from the current priority, then it sends a message to the junction agent notifying the priority change. A diagram representing this interaction by means of ACL messages is depicted in **Figure 8**. The junction agent receives the message and notifies the affected phases to calculate its priority. The phase agent will use the largest priority of the lanes that require the use of such a phase.

This simple system of three agents allows us to experiment with different methods of coordination. The proposed implementation method is to create a trade system where one phase

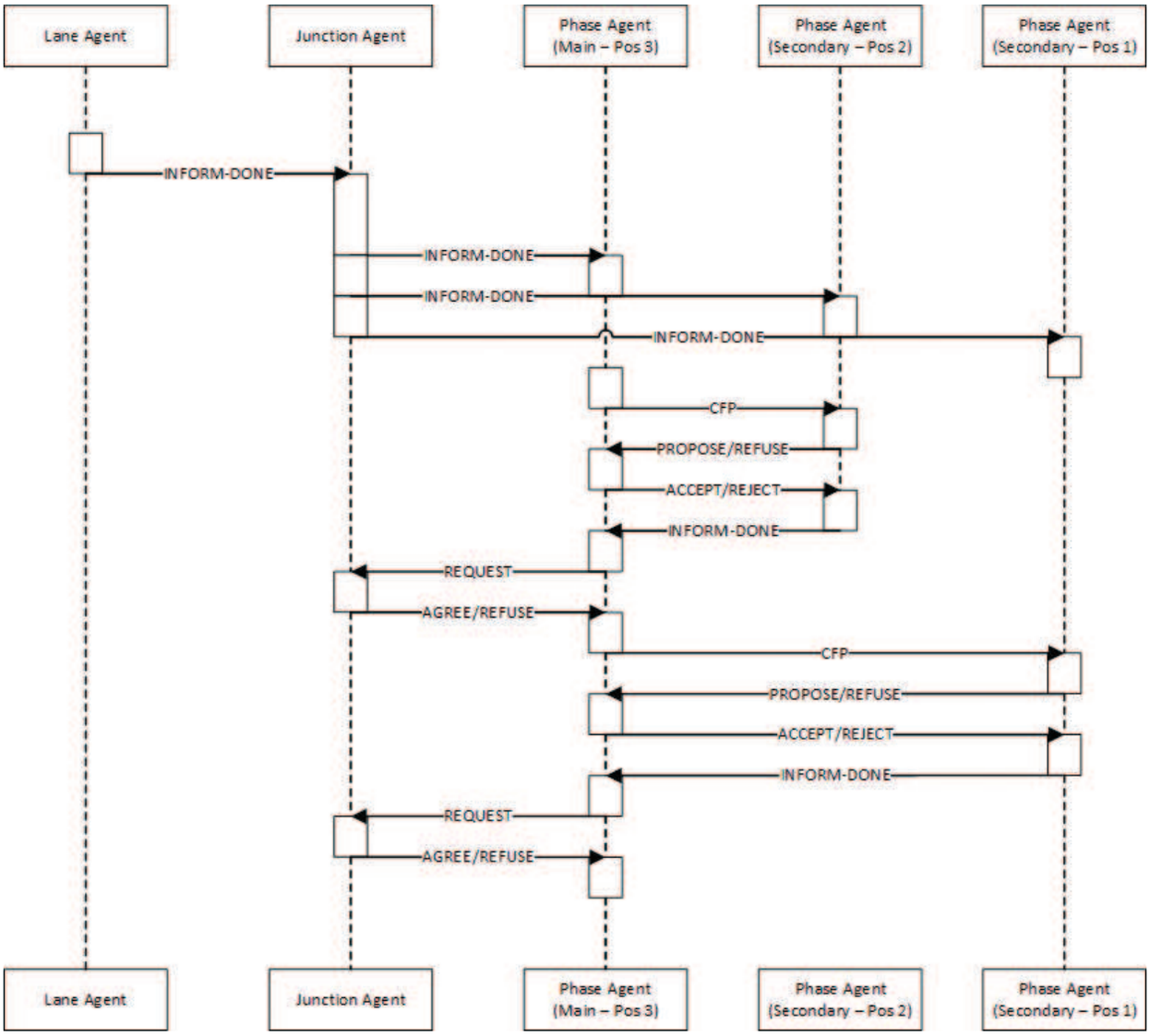

Figure 8. Diagram of negotiation process.

exchanges time for the possibility of the get up in the queue of priorities. Each phase calculates the priority as per the total number of vehicles each lane supports and the current number in the lane at a specific stage. With that information, the lane can setup a priority from one to five, where five means it is critical for that phase to be next one in the cycle.

One important aspect to consider is the fairness of the system. That is, some traffic light phases will have more seconds to negotiate than others. The rule of the tomb is a strategy for a system that can be beneficial for the phases with lower number of seconds and the phases with more seconds to spend. For this reason, instead of using the second as a raw currency, this work proposes to use the concept of a unit.

A unit may represent several seconds. However, the units may vary depending on the phase. That is, the unit will be the expected time of the phase divided by five. In this case, five is the number of columns we want our agents to work with. Thus, the expected value will be in any case that corresponds to the middle column. Accordingly, in the negotiations, any phase will have the unit value of two columns to the left to spent, and the unit value of the two columns to the right to gain. **Table 7** shows the unit values of the phase agents that it uses in the offering stage of the negotiate process.

The offer table contains the priority number as row and the number of units to gain, or lose, as columns. If the phase is at a certain priority and at a certain column, then with a simple lookup process, it is possible to determine the value that one phase agent should offer to take in the queue of another phase.

The accept table works in a similar fashion as the offer table. However, in this case when the phase receives an offer for its position in the queue, then it should check the accept table to decide whether to accept or reject the offer. The minimum value that the phase agent should accept is at the column and row of this table. Notice that there are some infinite symbols in the entries of the table. It means that for those situations, it does not matter the number of units offered, the phase will reject any offer, since that phase agent is at a situation where it is required to get into the junction cycle as soon as possible. **Table 8** shows the unit values of the phase agents that it uses in the accepting stage of the negotiate process.

For example, if the phase agent has 15 s of green time, then the unit value is 3 s (15 divided by 5).In this mechanism, the phase is not allowed to get lower than two units (6 s) and not bigger

Time (s)					
Priority	**−2**	**−1**	**0**	**+1**	**+2**
1	0	0	0	0	1
2	0	0	0	1	1
3	0	0	1	1	2
4	0	1	1	2	3
5	1	1	2	3	4

Table 7. Offer table.

Time (s)					
Priority	**−2**	**−1**	**0**	**+1**	**+2**
1	2	1	1	1	8
2	3	2	1	1	8
3	4	3	2	1	8
4	4	3	2	8	8
5	8	8	8	8	8

Table 8. Accept table.

than two units (6 s again). In this case, the negotiation units are in **Table 9**. Thus, if the phase is in priority 5, for instance, and currently has 15 s, then it will offer two units to the phase at the top in the queue. If that phase does not accept the offer, then the negotiation ends. However, if the phase at the top of the queue accepts, then the offering phase will take the place of the accepting phase in the queue. Accordingly, the offering phase will lower two units of time, with 9 s of green light, but up in the queue. The phase that accepted the offer will increase its time by two units, i.e., with 21 s of green light, but lower in the queue.

3.3. Experimentation

To test the negotiation strategy described in the previous subsection, the first step is to simulate the basic scenario when coordination may occur.

Figure 10 shows a four-road intersection in the SUMO simulator, which is used to simulate the negotiation process. The implementation of the four roads needs four phases to be fully functional. To represent the states of the phases, four cardinal points, North, South, East, and West, are considered **Figure 9**.

In phase 1, the cars can move in both directions, North-South and South-North. In phase 2, the cars go from North-East and South-West. In phase 3, the cars are allowed to move from East-West and West-East. Finally, in phase 4, the vehicles can go from West-North and East-South. With these four phases, all vehicles can move from one direction to all other different

Time (s)					
Priority	**9 s**	**12 s**	**15 s**	**18 s**	**21 s**
1	0	0	0	0	1
2	0	0	0	1	1
3	0	0	1	1	2
4	0	1	1	2	3
5	1	1	2	3	4

Table 9. Offer table for green time of 15 s.

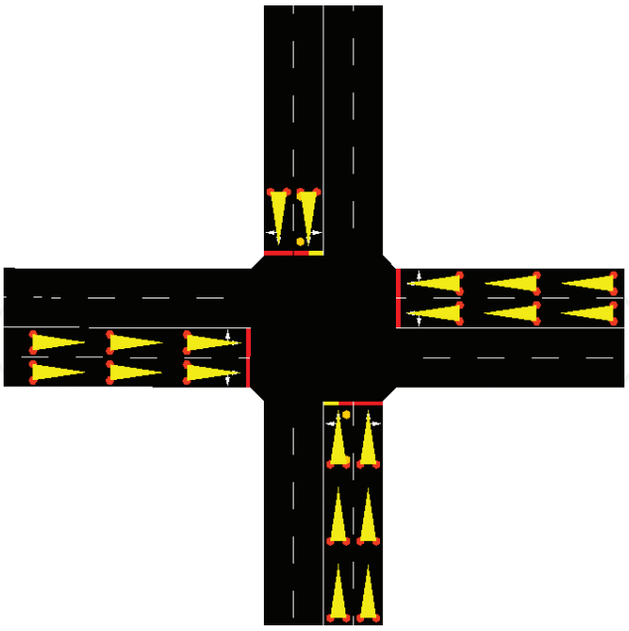

Figure 9. Four roads intersection.

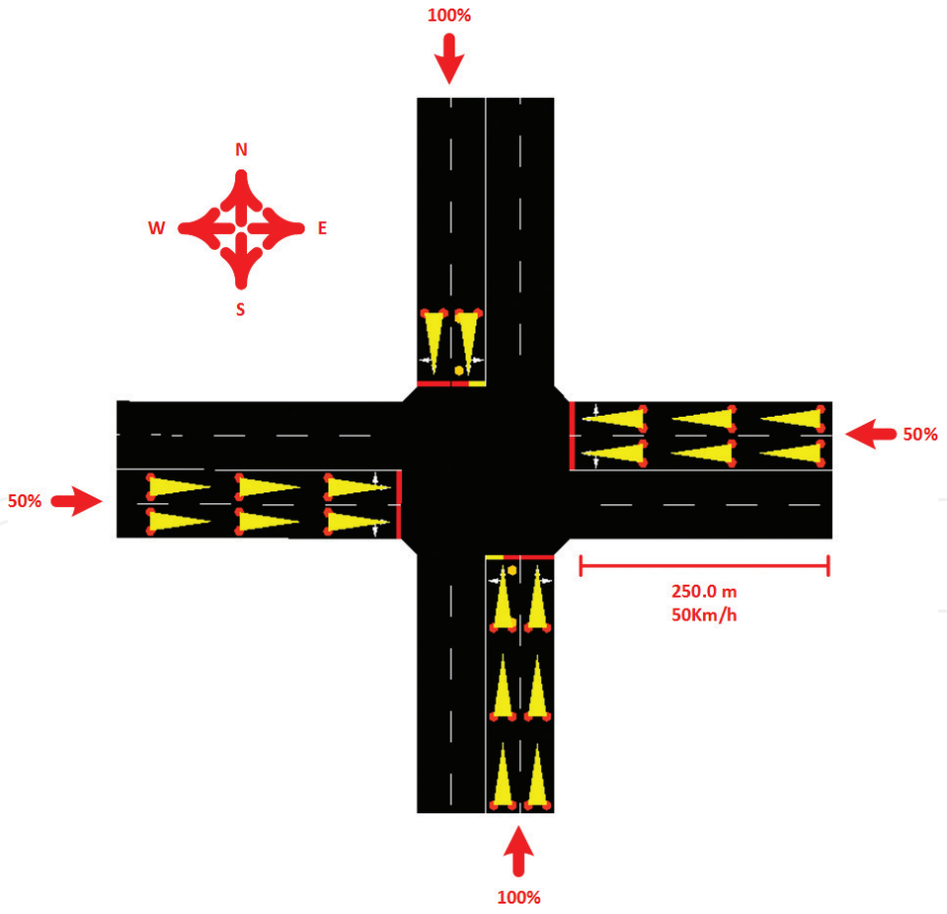

Figure 10. Simulation values.

locations they require to fully travel the intersection, even considering right turns allowed at any moment with precaution.

3.3.1. Integration of JADE agent platform and TraSMAPI

To implement the negotiation system, the SUMO simulator is interfaced with the JADE development framework. JADE requires a JVM to be executed. To execute the runtime environment, a simple command can be used to accomplish that goal:

java -cp <classpath> jade.Boot

where the classpath is the place where the jade.jar file lives in the system. However, it will create and empty the platform and the container, solely with the basic structure and no other than the default agents.

JADE provides a set of classes in the JAVA language that can be used to create the agents that implement the different pieces of the negotiation system previously described. The most important class for this purpose is the agent. The agents have a unique identifier, denoted as AID, which is used to uniquely determine a specific agent. The AID can be obtained using the method getAID. The identifiers in JADE are using a convention like an email address, i.e., <nickname>@<platform-name>; however, it is only a name and should be considered like that.

All the agents in JADE should extend the agent class. This inherits a set of methods to work in JADE framework. The two methods that require more attention are the setup and takedown.

The setup method is the place where the initialization of the agent occurs. It is used instead of the constructor method of a JAVA class. The agent class provides this different method, because it is safer to use and it can warrant that the system is up and running at that moment. This is something that cannot be possible with the traditional constructor. In the setup method, the agent parameters can be read to populate attributes by using the getArguments method.

The takedown method is invoked after the agent is terminated and this can be done by using the doDelete method in any place of the agent. The purpose of this method is to clean up any necessary objects or operations.

The communication between agents is the core functionality that needs to be implemented in JADE. To accomplish this task, JADE provides a behavior class. An agent can have different behaviors and all of them should be included using the addBehavior method. The behaviors are the mechanisms to implement the actions and methods of the agent.

There are a complete set of behaviors in JADE for different objectives. One shot behaviors, cyclic behaviors, generic behaviors, wake behaviors, and ticker behaviors, to mention some. One shot behaviors are implemented using the OneShotBehaviour class, this is meant to be executed only one time and after that delete the behavior from the agent. The cyclic behaviors use CyclicBehaviour class and they return false in the done method all the time, so this behavior repeats and keeps executing. The generic behaviors correspond to the Behaviour class, this is a vanilla class that can be extended and used as the user requires especially with

communicative acts that requires several messages between agents. The waker behaviors relate to the WakerBehaviour class and will be executed after a certain condition is reached, commonly a time set like an alarm. Finally, the ticker behaviors use TickerBehaviour class and are repeated every certain interval of time.

The communication between the JADE agents with the simulator SUMO is required. For example, the lane agent requires to know the number of vehicles in the simulator lane, and the junction agent needs to modify the traffic light in the simulation according to the queue. To establish a communication between the two frameworks, this chapter uses the traSMAPI middleware.

TraSMAPI is a project from the University of Porto, which is an API to communicate with microscopic traffic simulator (like SUMO). This allows to get information and manipulate the different elements of the simulation like vehicles, traffic light, etc. One of the most important aspects is that it is written in JAVA, the same language as JADE allowing to easily integrate the multiagent environment with the Simulation [12].

The way TraSMAPI manipulates the simulation is through an interface created in SUMO, which is called TraCI (Traffic Control Interface) [22]. The interface can be accessed by enabling a remote port in SUMO. By using the command line, this can be done by adding the parameter—remote-port [portNumber] or in the sumocfg gile adding the traci_server section like this:

<traci_server>

<remote-port value=”portNumber”/>

</traci_server>

With this, a series of bytes can be sent through that port to the SUMO simulation, the bytes correspond to the values of the instructions required to interact, first byte reserved for the command and the following, for parameters required to get data or modify any characteristics of the running simulation.

3.4. Experimentation

To test the implementation, two types of traffic light scenarios will be used. One with the traditional static, or fix, times for the lights in the semaphores, and one with dynamic phases negotiation that use agents.

The vehicles will be generated using a fix number per hour. Two combinations will be used to simulate more traffic flowing from north-south lines. In north to south lines, the flow will be a complete load of vehicles and in east to west lines, 50% of the full load. The full load will have values of 500, 750, 1000, and 1250 vehicles per hour. The details of the intersection are depicted in **Figure 11**. For the traffic, light phases, we will be using four phases as described in **Table 10**.

The results using static (s) traffic lights and using dynamic (d) traffic lights are shown in **Figure 11**.

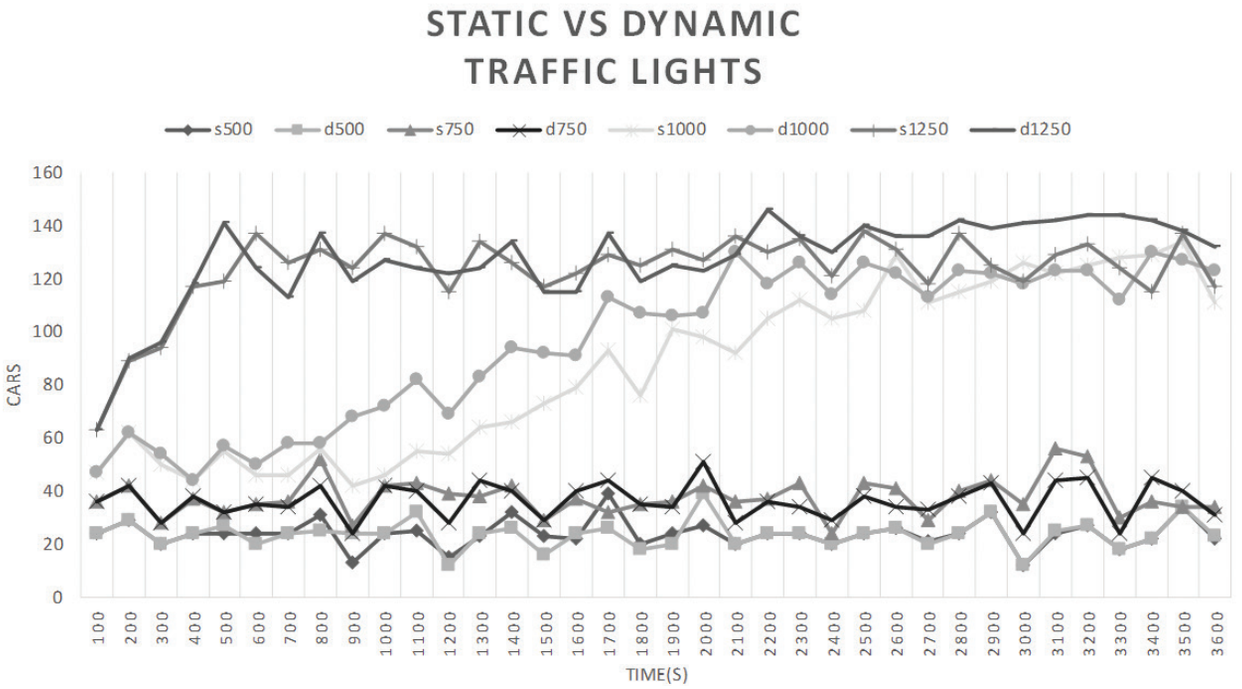

Figure 11. Static versus dynamic results.

Phase	Direction	Green time (s)	Yellow time (s)
1	NS-SN	15	4
2	NE-SW	6	4
3	WE-EW	15	4
4	WN-ES	6	4

Table 10. Configuration of the phase.

The result shows a similar behavior in the static traffic light and the dynamic ones using the negotiation mechanism with agents. At 1250 veh/h, both systems have difficulties to manage the vehicles load. Further experimentation is encouraged with different phases and times in the traffic light. It is clear that other coordination tables may be constructed to improve the balancing of vehicles under different load conditions and street configurations.

4. Conclusions

This chapter proposed the application of the MAS technology and concepts to the solution of problems in the automotive field. The MAS has provided suitable solution to problems of distributed nature, such as those present in the automotive field. The vehicles (both, cargo and utilitarian), the infrastructure (lane, semaphores, etc.), and even the pedestrian are suitable to

be modeled as agents. This simplifies the modeling and simulation, and thus the construction of solution to problems of smart traffic systems. The communication mechanisms of the MAS are well suited to implement with simplicity, complex interaction protocols for the car-2-X communication. In particular, this chapter proposed the application of two mechanisms of the MAS to the automotive field. One the one hand, it proposed the utilization of QoS mechanism to the coordination between the cars and the infrastructure. On the other hand, it proposed the utilization of an auction-based mechanism for the negotiation between faces in lane intersections.

By using the set of tools and techniques described in this chapter, solutions to intelligent traffic systems may be approached from the MAS field. The experimentation with the traffic simulators and the framework for the agent implementation seem to be a new way to design solutions that may be quite complex to implement with other approaches.

Author details

Raul Campos-Rodriguez*, Luis Gonzalez-Jimenez, Francisco Cervantes-Alvarez, Francisco Amezcua-Garcia and Miguel Fernandez-Garcia

*Address all correspondence to: rcampos@iteso.mx

Electronics, Systems and Informatics Department, ITESO University, Tlaquepaque, Jalisco, Mexico

References

[1] Rogers A, Corkill DD, Jennings NR. Agent technologies for sensor networks. IEEE Intelligent Systems. 2009;**24**(2):13-17

[2] Sayed AH. Adaptation, learning, and optimization over networks. Foundations and Trends® in Machine Learning. 2014;**7**(4-5):311-801

[3] Burg B. & FIPA VP. Foundation for Intelligent Physical Agents. Official FIPA presentation, Lausanne, February, 2002. Foundation for Intelligent Physical Agents, http://www.fipa.org/

[4] FIPA00002, S. FIPA Agent Management Specification. 2000. http://www.fipa.org/specs/fipa00023/index.html

[5] Poslad S. Specifying protocols for multi-agent systems interaction. ACM Transactions on Autonomous and Adaptive Systems (TAAS). 2007;**2**(4):15

[6] Bomarius F. A Multi-Agent Approach towards Modeling Urban Traffic Scenarios. Research Report RR-92-47, Deutches Forschungszentrum für Künstliche Intelligenz, September 1992.

[7] Adler JL, Satapathy G, Manikonda V, Bowles B, & Blue VJ. A multi-agent approach to cooperative traffic management and route guidance. Transportation Research Part B: Methodological, 2005;**39**(4):297-318.

[8] Yamashita T, Izumi K, Kurumatani K, & Nakashima H. Smooth traffic flow with a cooperative car navigation system. In Proc. of the fourth international joint conference on Autonomous agents and multiagent systems (pp. 478-485), ACM. Utrecht, Netherlands, 2005.

[9] Enhamza K & Seridi H. Intelligent traffic light control using an adaptive approach, In Proc. Of IT4OD, p. 246-250, Morocco, 2014.

[10] Abdoos M, Mozayani N & Bazzan AL. Traffic light control in non-stationary environments based on multi agent q-learning, In Proc. Of 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 1580-1585. Washington, DC, 2011.

[11] Azevedo T, De Araújo PJ, Rossetti RJ & Rocha APC. JADE, TraSMAPI and SUMO: A tool-chain for simulating traffic light control. In Proc. Of the 13th International Joint Conference on Autonomous Agents and Multiagent Systems, Paris, 2014.

[12] Timóteo IJ, Araújo MR, Rossetti RJ & Oliveira EC. (2010, September). TraSMAPI: An API oriented towards Multi-Agent Systems real-time interaction with multiple Traffic Simulators. In Proc. Of 13th International IEEE Conference on Intelligent Transportation Systems (ITSC), (pp. 1183-1188). Funchal, Madeira Island, Portugal, 2010.

[13] Abdoos M, Mozayani N & Bazzan AL. Hierarchical control of traffic signals using Q-learning with tile coding. Applied Intelligence. 2014;**40**(2):201-213

[14] Tlig M & Bhouri N. A multi-agent system for urban traffic and buses regularity control. Procedia-Social and Behavioral Sciences. 2011;**20**:896-905

[15] Bellifemine F, Poggi A & Rimassa G. JADE–A FIPA-compliant agent framework. In Proceedings of PAAM. 1999;**99**:97-108, p 33. Available online at: http://jade.tilab.com/

[16] Christian B & Griffiths T. Algorithms to live by: The computer science of human decisions. Macmillan, USA, 2016.

[17] Barceló J. Fundamentals of traffic simulation. New York: Springer. 2010;**145**:439.

[18] Krajzewicz D, Hertkorn G, Rössel C & Wagner P. SUMO (Simulation of Urban MObility)-an open-source traffic simulation. In Proc. of the 4th middle East Symposium on Simulation and Modelling (MESM20002). Sharjah, United Arab Emirates, 2002; 183-187.

[19] Vaudrin F & Capus L. Experiment of a common sense based-approach to improve coordination of Traffic Signal Timing System with SUMO. In Proc. Of Intermodal Simulation for Intermodal Transport SUMO 2015. Berlin, 2015

[20] Bellifemine FL, Caire G & Greenwood D. Developing multi-agent systems with JADE. John Wiley & Sons, 2007:**7**.

[21] Adler JL, Satapathy G, Manikonda V, Bowles B & Blue VJ. (2005). A multi-agent approach to cooperative traffic management and route guidance. Transportation Research Part B: Methodological. **39**(4):297-318.

[22] Wegener A, Piórkowski M, Raya M, Hellbrück H, Fischer S & Hubaux JP. (2008). TraCI: an interface for coupling road traffic and network simulators. In Proc. of the 11th communications and networking simulation symposium (pp. 155-163). ACM, 2008

IntechOpen