Simulating Communication in a Service-Oriented Architecture for V2V Networks

João F. B. Gonçalves, Edgar F. Esteves,
Rosaldo J. F. Rossetti, Eugénio C. Oliveira

Department of Informatics Engineering
Artificial Intelligence and Computer Science Laboratory (LIACC)
Faculty of Engineering, University of Porto (FEUP)
Rua Dr. Roberto Frias, S/N • 4200-465 Porto • Portugal
{ee02123, edgar.esteves, rossetti, eco}@fe.up.pt

Abstract. A framework based on the concept of service-oriented architectures (SOA) to support the assessment of vehicular *ad-hoc* networks (VANET) is herein presented. Concepts related to SOA, as well as technologies that allow real-time data acquisition and dissemination within urban environments, and simulation tools to aid the simulation of VANET were preliminarily studied. A two-layered architecture was specified on the basis of the requirements for our simulation framework resulting in the specification of a multi-agent system formed of vehicle entities that are able to communicate and interact with each other and with their surrounding environment as well. A prototypical application was implemented, which was used to demonstrate the feasibility of the approach presented through experimental results.

Keywords: Intelligent Transportation Systems, Service-Oriented Architectures, Vehicle-to-Vehicle Communication and Simulation, Multi-Agent Systems.

1 Introduction

Most urban areas all over the globe have increased considerably in the last few decades, giving rise to important mobility issues. Unfortunately, virtually all people tend to opt for using private car instead of public transport. This trend has turned infrastructures unable to cope with the ever increasing demand, which work most of the time in saturation regime. This problem motivates the scientific community that strives to devise different solutions. Some approaches suggest the physical improvement of infrastructure by means of increasing road capacity. Others try to enhance the control systems responsiveness, with relative success. More recently, though, researchers have experimented promising innovative information technologies to aid driving tasks and traveller decision-making. The latter underlie the concept of intelligent transportation systems (ITS), which attempts at integrating contemporary and breakthrough computer and information technologies to better managing and controlling modern urban transportation systems. ITS is intended to

turn future urban transport (FUT) into greener, cheaper, reliable, secure, and sustainable systems, functionally, energetically and environmentally efficient.

Indeed, high-class vehicles will very soon be equipped with short-range wireless communication interfaces, bringing about major concerning issues as well. Thus, simulation tools will need to be adapted to support the assessment of V2V communication infrastructures and performance. This work is based on our study of the communication technologies underlying vehicle-to-vehicle interactions then. We aim at studying concepts related to SOA and ways in which such concepts can be adapted to VANET. The work started by the specification and implementation of a simulation framework on the basis of the concept of multi-agent systems, and preliminary results allowed us to gain further insight into such motivating and challenging new arena.

2 Related Technologies

ITS-based technologies have proven a great impact and influence on future urban transport scenarios. As the automotive industry starts marketing vehicles equipped with wireless communication capabilities, some technologies are being deemed be potentially applicable and beneficial.

2.1 Service Oriented Architectures

Service-oriented architectures are devised as software architectures whose main goal advocates that application functionalities must be made available in form of services. Thus, services from SOA's point of view are functions of a computational system that are made available to other systems. Such a novel approach might be well represented by the "find-bind-execute" paradigm, which is analogous to the Deming's cycle applied to services, involving planning, execution, monitoring and pro-active decision-making phases to improve systems' performance. The "find-bind-execute" paradigm allows a consumer of a service to ask for registering in a service provider that suits its criteria and requirements. If the service of interest is found, the provider sends the consumer a contract and the address in which the service can be found. Such a mechanism is depicted in Figure 1.

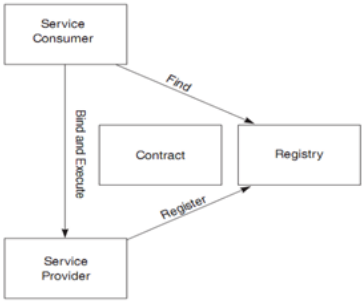

Fig. 1. Find-bind-execute paradigm (adapted from [1])

According to [1], the entities that support such a service infrastructure are service consumers, providers, registry and contract.

Service consumers are basically an application, a service, or another type of software requiring a service. This is the entity who initiates the process, looking for a service capable of supplying its requirements. The consumer executes the service by sending a request formatted in accordance with the contract.

Service providers are addressable entities that accept and execute services. These entities might be a mainframe, a component or another type of software that executes a requested service. Service providers publish their contract in the service registry for other potentially interested consumers to access them.

The *service registry* is a sort of network directory that "knows" the available services. It accepts and stores contracts of providers and provides consumers with these service options. Finally, a *service contract* is a kind of specification of the way in which a service consumer must interact with the provider. This entity rules the protocol for requests and respective answers from services. A contract may require a set of pre- and post-conditions representing the state of a service to be deemed acceptable to execute certain functions. Contracts may also contain information on quality of services, as well as conditions to which consumers must comply.

2.2 Vehicle-to-Vehicle Networks

Vehicle-to-vehicle communication networks are, as its designation suggests, networks formed by several vehicles equipped with wireless communication devices that can communicate with each other. In V2V networks, each vehicle analyses, within a certain radius, other vehicles that are in range, and can inform its position, velocity, direction and other characteristics. This kind of communication has been one of the fields of interest in telecommunications that grew very quickly lately. Thus, vehicles with such capabilities can form a special type of mobile *ad-hoc* network with particular applications, known as Vehicular Ad-hoc Networks (VANET). VANET are a special type of Mobile Ad-hoc Networks (MANET) that supports communications between vehicles. According to [2], VANET inherit some characteristics from the MANET, but also improve the former with new features, which differentiate it from other *ad-hoc* mobile networks. These characteristics include high mobility, open network with dynamic topology, limited connectivity, potential to achieve larger scale, all nodes are providers, forwarders and consumers of data and the wireless transmission can suffer from much noise and interferences.

These characteristics make VANET sufficiently different from other networks and significantly affect their properties. For example, in [3] it is demonstrated that the movement of vehicles has a significant effect on the latency of messages delivery. Most applications that can be implemented on a MANET require a certain type of data dissemination. Studies argue, however, this must be implemented through specific routing protocols, as long as VANET are concerned, basically due to their particular characteristics, such as in [4]. Nonetheless, there are other studies that suggest it is possible to use the available MANET protocols. Some of those protocols are designed to support dedicated short range communications (DSRC), which is

already implemented in USA, such as VITP and PAVAN. Due to its topology, the MANET already have a large set of protocols.

There are several ways to classify routing protocols for such networks, some of which are listed below:

- According to the range, they can be either unicast or multicast.
- According to the route discovery, they can be either pro-active, reactive or hybrid.
- According to the search algorithm they are based on, they can be either Distance Vector, Link State, based on geographic information, or Zone based.

In Figure 2, the protocols previously mentioned are related, as well as are their classification according to the characteristics presented above.

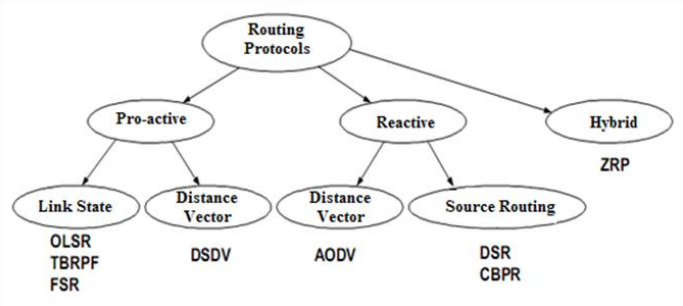

Fig. 2. MANET routing protocols classification

2.3 Simulation Tools for VANET

The development of applications and protocols associated to VANET can be studied through simulation, especially when a real traffic network in urban environments, which must involve a large number of nodes, is subject to study. Through simulation models, the performance of V2V networks, as well as other characteristics can be assessed and improved. Indeed, conducting experiments and studies on such a large scale within the real scenario has proven extremely difficult and expensive. Thus, simulation has become an indispensable and even an imperative tool.

Basically, simulation of V2V networks requires two different components, namely a communication networks simulator, capable of simulating the properties of a wireless network, and a vehicular traffic simulator, able to monitor and represent the kinematic aspects of mobility through the VANET nodes. Recent studies [5] suggested that the vehicular mobility model is very important to obtain significant results and should be well integrated with the wireless communication networks model. Other authors further suggested that the use of an inappropriate model, such as the popular "random waypoint model" (which can work very well for some mobile *ad-hoc* networks, but very likely is not an appropriate representation of mobility in wireless vehicular networks) can lead to erroneous results [5], [6].

Nevertheless, traffic simulators have been subjected to enormous developments and greatly improved in recent years to include communication between vehicles. Several ways to achieve such an advanced feature have been proposed and actually

implemented. The greatest trend in most studies moves towards the creation of simulators that include traffic and wireless communication simulation models in one single simulation tool. Some examples of such tools include GrooveNet and Divert.

However, other studies prefer to use two independent simulators, combining stand-alone traffic and communication networks simulators, which are interconnected by an application that ensures the exchange of information between them. Among traffic simulators used for this purpose, one can mention the popular SUMO, as well as VISSIM and CARISMA. Wireless communication networks simulators include GloMoSim, QualNet and the NS2. The interested reader is referred to [7] for a more detailed discussion on the above simulation tools, including their respective references.

3 Coupling SOA and V2V Communication

Currently, services that both drivers and travellers in general can use on-board in journey time demand a great deal of hardware and software. Each new feature must be implemented in a new device to be embedded in the car. Such an approach has proven very expensive, with little flexibility, which contradicts the increasing trend of services made available on-demand, for instance. On the other hand, services made available as software to be executed in an on-board unit (OBU) based on an embedded computer with considerable processing and memory, as well as communication capabilities seem to be very promising and present great potentials and advantages. According to such not-so-much futuristic scenario, we intend to specify and implement an extensible architecture based on OBU computers and featured with communication capabilities to the level of services.

Bearing in mind that such architecture was intended to promote services throughout V2V communication networks, we opted for a layer-based structure. Thus, the proposed architecture is divided into two main levels, namely the network services level and the end-user services level. This approach is illustrated in Figure 3, in which the two different levels are identified. The first level in such a structure is responsible for network-related tasks, such as building network topology, as well as discovering and exchanging services among vehicles (the nodes of the communication network). The second level, on the other hand, implements the necessary basis underlying the management of the so called high-level services, to be made available to end-users.

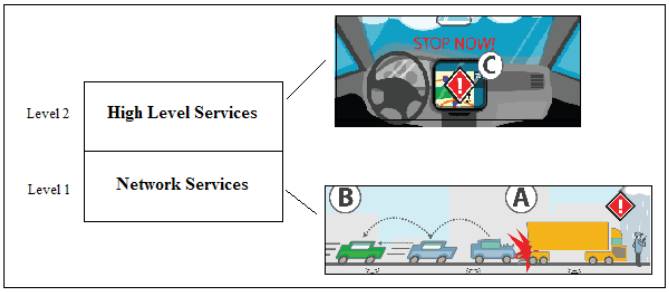

Fig. 3. SOA layered architecture for V2V networks

3.2 The Network Services Layer

For the first level, and accounting for its functions within the proposed structure, we adopted the solution suggested in [8]. The scenario illustrated in Figure 4 is a good representation of the adopted solution, as explained below.

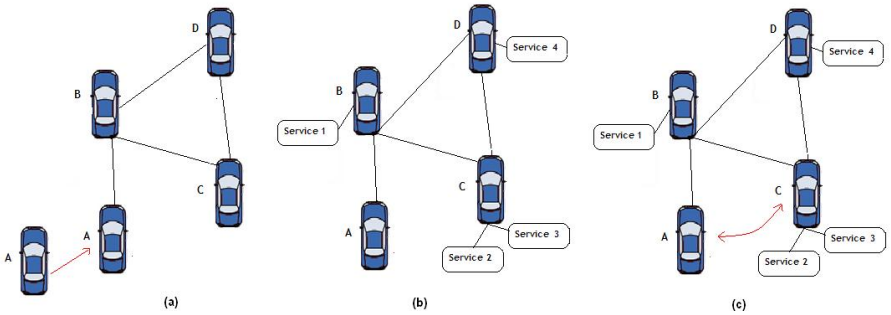

Fig. 4. SOA-based V2V communication interactions

Let us consider the communication network formed up by vehicles as illustrated in Figure 4. In (a), the network is actually set-up. All nodes (A to D) are vehicles equipped with mobile wireless V2V communication devices. Initially, A node is not connected for there is no other vehicle within its range vicinity. As soon as it finds other vehicles within its range, then they get connected. All connected nodes thus form a VANET, allowing V2V communication. After all the necessary automatic configurations are set up, the VANET is ready for routing any messages sent by any node and addressed to any other node of the network.

After the VANET is ready, the SOA-based features must be deployed too. All nodes that provide any service must publish it alongside a comprehensible description, so that other nodes will be able to discover the service and use it as needed. The service discovering capabilities of each node must be implemented in such a way that the node may be able to find the needed service throughout the VANET which every configuration must apply. Thus, in (b), provider nodes publish their services, whereas any consumer will be able to find them whenever necessary.

An illustrative service use example is demonstrated in (c). A node gets connected to the service interface of service whose ID is 3, hosted in C node. All information necessary to message exchange between consumer (A node) and provider (C node) is available in the description of service 3, needing A no further information about C. As A and C nodes are not directly connected to each other, messages must be routed through B node.

Two basic components are necessary for the example presented above to be possible, namely a routing protocol for VANET and the implementation of SOA functionalities. One of the most important SOA features is the service discovering mechanism. Two approaches are then possible for us to implement such a mechanism. First, we can integrate it within a VANET routing protocol or, alternatively, we can implement it on top of the network layer, as a separate functionality. In the current work we opted for the first solution, and integrated it in the VANET routing mechanism. Such a decision proved advantageous as both the routing tables and

protocol routing techniques are used in the service discovering process. For this purpose, we chose the optimised like state routing protocol (OLSR) for it is a pro-active protocol, which facilitates the propagation of services description throughout a network of known topology. Among pro-active protocols, OLSR has the advantage of using multi-points relaying (MPR) and offers an open source application (OLSR *Daemon*) allowing it to be extended.

Routing in MANET is based in the cooperation of participating nodes; all nodes must run the protocol in such a way they collectively achieve routing goals. Such a behavior is also desirable in VANET, as all nodes are expected to collaborate in services discovery as well. In practice, it means all nodes manage message routing even though they may not know its contents. This way all service provider nodes may use the entire network and routing protocol.

A node intending to publish its services must generate and propagate SOA messages periodically, carrying the service information, according to certain transmission interval. Whenever a node receives a SOA message, it must resend it and, if it supports SOA services, process it too.

The format of a SOA message has two basic fields, namely the message length and the service description, which describes the service as a unique service (no other service will have the same description). This sort of message is carried within OSLR packages and propagated throughout the network. Nonetheless, there is no specific format for the content of a service descriptor, which will depend on the type of service and the way it is implemented. XML might be used for this purpose, though.

3.2 The High Level Services Layer

The second layer of the proposed architecture consists of the necessary structure to adequately manage the high level services to users. In other words, high level services can be seen as final applications presenting some degree of interaction with users, most of the time through a graphical user interface (GUI), which can be accessed from inside the vehicle. Thus, all hardware and software components somehow necessary to effectively run such services are specified in this level. Some examples include traffic jams detection, collision avoidance and emergency calls among others.

Once again, we consider that vehicles are equipped with some sort of OBU as discussed earlier. Figure 5 depicts the main components of this level in a UML development diagram, illustrating the distribution of components both embedded in a vehicle and those that are present in the surrounding environment, and eventually in other vehicles.

Before we can go a bit further into the description of each component, the service concept must be clarified in this context. In this architecture, *services* are abstract resources able to carry out tasks that are coherent both from provider and from request entities' point of view, and are classified according to their abstraction level. Thus, *low level* services are basically pieces of software that access physical devices of a vehicle. On the contrary, *high level* services are responsible to carry out tasks that are expected to transform information somehow. Such a hierarchical structure of services has a two-fold purpose. Firstly, it offers a modular programming infrastructure that is extensible and scalable. Secondly, it seems to be adequate to

overcome synchronisation issues while a variety of devices are accessed. Indeed, for the latter case, the *vehicle* component can be actually seen as an agent, whereas multiple vehicles interacting within an urban network environment form a multi-agent system. This metaphor becomes quite intuitive when we consider that vehicles are equipped with sensors, are able to determine their geographical location and to interact with the surrounding environment, as well as with other vehicles. From this standpoint, the OBU is their reasoning kernel that executes tasks of both levels.

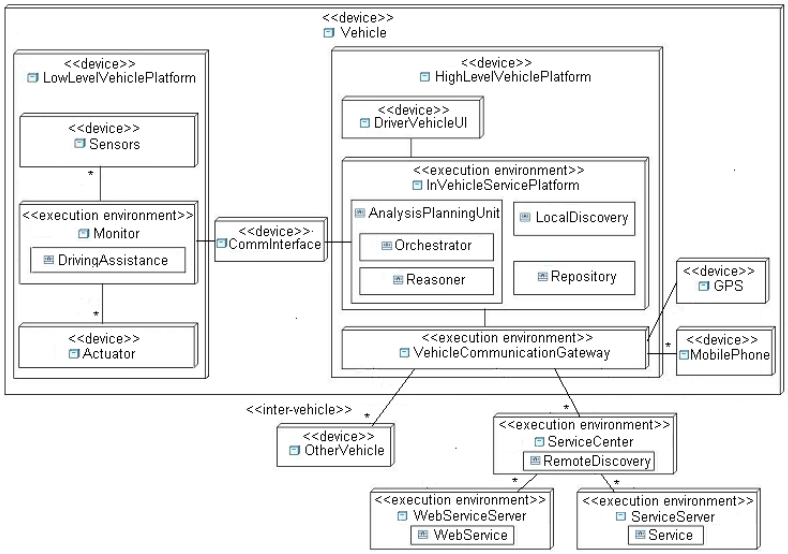

Fig. 5. UML development diagram of a vehicle service-oriented architecture

As for the *low level platform*, it contains the hardware and software elements of a vehicle in charge of critical services such as driving aids (e.g. ABS and ESP). Also, it features a number of diverse actuation software pieces reacting to vehicle's sensors to ensure a minimum answer time and meet real-time constraints. Thus, *sensors* components represent the necessary interface for vehicles to define their current state with regard their surrounding environment and neighbouring counterparts. *Actuators*, on the other hand, automatically trigger assisted driving capabilities whenever they are needed. The relationship between sensors and actuators is basically implemented through *monitors*. These components manage sensors and are constantly observing their activities. Whenever sensors accuse an event, monitors give the appropriate warning and evaluate the situation. Critical events are then reported to actuators so as they can react accordingly, while others are submitted to the *analysis and planning* unit. *Assisted driving* elements contain all functionalities to aid the low level driving tasks triggered by actuators and managed by monitors, after sensors' events have been evaluated.

Drivers and passengers alike interact with active services in the *high level platform*. They can receive information and select commands to trigger, stop or adapt services through the *driver-vehicle interface*. The *service platform* is a subcomponent accessible from the vehicle that contains a service repository and features the necessary functionalities to manage services' specificities, such as service discovery

and orchestration. An *analysis and planning* unit evaluates events reported by other elements such as the monitor, the driver-vehicle interface and the communication gateway, and other current plans to define the appropriate course of actions accordingly. We intend to extend this feature with a BDI-like (belief, desire, intention) architecture, as initially proposed in [9].

The *service repository* is the place where services and other necessary applications are locally stored in the vehicle. This element can be inquired by certain services, local or remote, and asked for a specific function or action. It features functions such as service registry, discovery and update as well. *Local service discovery* allows local services to be found. In this structure, an *orchestrator* is responsible for pursuing and achieving a given goal through the invocation of one or more services combined, whereas a *selector* analyses and selects services based on certain criteria that are set in advance.

Both platforms, namely the low level and the high level as discussed so far, are features of the vehicle element and interact through the *communication interface*. Nonetheless, vehicles can also interact with external services, hosted in other remote elements. The *vehicular communication gateway* is responsible for that, sending messages to external components (e.g. other vehicles or servers), on an abstract basis firstly. Then the most appropriate communication technique is selected (e.g. GSM, GPRS or VANET) and the message is effectively sent. Mobile phones and PDA, for instance, are used to build long range communication connections, and can be embedded in vehicles or borne by drivers and/or passengers that normally use them in other environments. A global positioning system (GPS) receiver, on the other hand, keeps track of the vehicle's position.

Apart from vehicles, services can be executed in other environments, which is a more classical perspective. The beauty behind vehicular *ad-hoc* networks is their great potential to provide transport systems with a great number of diverse services as people move. The *services centre* is a remote execution platform that offers services discovery and other applications that support active services' requests in a more traditional way, but now including requests from VANET. It also features a *remote discovery* function that finds appropriate services throughout the network, which can be present in diverse environments such as web servers (e.g. web services on the Internet) or traditional service providers implementing private client-server environments.

4 Experimental Set-up and Results

The experimental framework carried out was based on a simple network. Despite its simplicity, results showed promising potentials of the proposed approach. The network was coded in the XML format supported in the simulation engine of MAS-T^{2}er Lab, as defined in [10, 11, 12], which was extended to support the current prototypical implementation. Readers are referred to [7] for more details.

4.1 Scenarios Definition

Three different traffic flow configurations were set up, which allowed us to test the system behaviour under different flow regimes. The effects of these flow regimes on the V2V communication performance were analysed then.

A low traffic flow regime was implemented in the *first scenario*, in which source nodes were set to yield 100 to 250 vehicle/hour traffic flows, representing a free-flow regime. The s*econd scenario* is intended to represent an average traffic flow regime. In this case, source nodes were set to yield 250 to 850 vehicle/hour traffic flows, allowing us to analysis V2V performance under average conditions. Finally, in the *third scenario* source nodes were set to yield 850 to 1300 vehicles/hour traffic flows, representing traffic under saturation conditions. Such a scenario is quite common in must urban areas, during peak hours, for instance, or during the occurrence of some incidents strangulating road capacity, such as accidents.

Each simulation run was set to last for 10 minutes' time, corresponding to 30 minutes in real time. To test the communication interaction between vehicles, a car approaching an intersection sends a message (this is done with a 2 minutes' frequency). Intersections are selected sequentially, following a clockwise order. Allowing cars to send messages at intersections is required as an attempt at maximizing the neighbouring cars within the range of the sender's wireless range. Indeed, as intersections are spots of converging traffic streams, then it is very likely the number of cars at intersections, especially in the second and third scenarios, will be considerable.

4.2 Preliminary Results

Preliminary results are plotted in the graph of Figure 6. The graph shows how much of the network is covered by the V2V communication mechanism over time. The results are very interesting and demonstrate the ability of the implemented prototype to cope with both communication and vehicular traffic simulation. As we expected, the results of different scenario set-ups suggest different behaviours for the system under varying traffic conditions.

In the low traffic flow scenario, message dissemination through the V2V infrastructure é quite poor, basically due to the discontinued coverage of the network. Only a small part of the network, *circa* 27%, can be affected by the message propagation. In such circumstances, it is possible that even none of the vehicles will receive the message sent.

Scenarios in which traffic flow follows an average flow regime, such as in the second experimental set-up above, a larger part of the network can be easily covered. In our case, about two thirds of the network was covered in the most appropriate circumstances, meaning the increasing number of cars propitiated a better coverage and improved communication.

Only in the third scenario it was possible to achieve a full coverage of the network, which is quite expectable too. Indeed, in nearly saturated traffic conditions, network links tend to work in their full capacity, meaning the density is very high and vehicle headways tend to the average vehicle unit. In these circumstances, it is most probable

that neighbouring vehicles will be within the range of other vehicles' wireless sensors, improving the connectivity of the network.

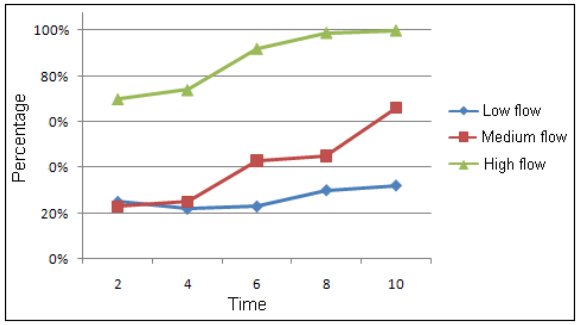

Fig. 6. Comparison of the three different scenarios, relating simulation time and the percentage of network covered by the V2V communication

In all simulation scenarios, nonetheless, coverage increases as time evolves, which can be associated to the small network used and to the semaphore control at all junctions. Indeed, controlled intersections tend to group vehicles together at red lights, especially if traffic flow is higher than the saturation flows of each phase of the control plan. In such a situation, intersections will never be empty improving the connectivity of the V2V communication network.

The three different scenario set-ups were inspired in a typical daily flow profile of a urban network, in which the first scenario might represent a free flow regime, at high night and dawn, the second scenario might represent average situation of off-peak hours during the day, and the third one the morning and afternoon peak hours. Nonetheless, the first scenario might also be associated to rural areas, whereas the third scenarios might well be representative of a bottleneck caused by an incident, such as accident, for instance.

In either case, results corroborate the idea that V2V communication presents a great potential and is a promising technology of future urban transport, whereas implementing SOA beneath such communication infrastructure can contribute a great deal for a better transport service and quality of life in urban areas.

5 Conclusions

In this work, a V2V architecture based on the concept of services was specified. A layered architecture based on different levels of abstractions of services providing SOA in V2V networks was devised in terms of a multi-agent system, as well as was a prototype implemented for testing and evaluating the approach proposed. For the first level of our architecture, we have implemented a SOA-based feature that allows services discovering within the routing protocol of VANET networks. This resulted in the specification of a dynamic and adaptable routing structure compliant with the abstract nature inherent in any SOA-based applications. For the second level, on the other hand, we presented a modular and easily extensible architecture that allows services to be made available in vehicles, based on the concept of multi-agent

systems. We have specified the vehicular architecture that underlies the implementation of on-board services, as well as the adequate means to request services from exogenous sources and other vehicles too. In general terms, the developed prototype and experimental results demonstrated the feasibility of the proposed approach. The simulation environment resulted from the improvements implemented is an important asset for testing and experimenting new generation intelligent transportation systems, which will strongly rely on V2V communication capabilities. Further developments will also include implementation of more complex scenarios and adequate tools to support the assessment of a whole urban network.

References

1. McGovern, J., Tyagi, S., Stevens, M., Mathew, S.: Java Web Services Architecture. Morgan Kaufmann (2003)
2. Mello, H., Endler, M.: Identificação de Região de Congestionamento através de Comunicação Inter-veicular. Monografias em Ciência da Computação, 26 (2006)
3. Chen, Z.D., Kung, H., Vlah, D.: Ad hoc relay wireless networks over moving vehicles on highways. In: 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp.247-250 (2001)
4. Blum, J.J., Eskandarian, A., Hoffman, L.J.: Challenges of inter-vehicle ad hoc network. In: IEEE Transactions on Intelligent Transportation Systems, pp.347-351 (2004)
5. Choffnes, D.R., Bustamante, F.E.: An Integrated Mobility and Traffic Model for Vehicular Wireless Networks. In: 2nd ACM International Workshop on Vehicular Ad Hoc Networks (VANET), pp.65-78 (2005)
6. Saha, A., Johnson, D.: Modelling Mobility for Vehicular Ad-hoc Networks. In: ACM International Workshop on Vehicular Ad Hoc Networks (VANET), pp.91-92 (2004)
7. Gonçalves, J.: Arquitectura baseada em serviços para redes veículo-a-veículo. Master's Dissertation. FEUP, Porto (2009)
8. Halonen, T., Ojaja, T.: Cross-Layer Design for Providing Service Oriented Architecture in a Mobile Ad hoc Network. In: 5th international Conference on Mobile and ubiquitous multimedia, paper 11 (2006)
9. Rao, A.: AgentSpeak(L): BDI agents speak out in a logical computable language. In: 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, LNAI 1038, pp. 42-55. Springer (1996)
10. Rossetti, R.J.F., Oliveira, E.C., Bazzan, A.L.C.: Towards a specification of a framework for sustainable transportation analysis. In: 13th Portuguese Conference on Artificial Intelligence, Guimarães, Portugal (2007)
11. Ferreira, P.A.F., Esteves, F.E., Rossetti, R.J.F., Oliveira, E.C.: A Cooperative Simulation Framework for Traffic and Transportation Engineering. In: 5th International Conference on Cooperative Design, Visualization, and Engineering, pp.89-97 (2008)
12. Ferreira P.A.F.: Specification and Implementation of an Artificial Transport System. Master's Dissertation. FEUP, Porto (2008)