Simulation and Testing of a Platooning Management Protocol Implementation

Bruno Ribeiro(✉), Fábio Gonçalves, Alexandre Santos(✉), Maria João Nicolau, Bruno Dias, Joaquim Macedo, and António Costa

Department of Informatics, Algoritmi Center, University of Minho, Campus de Gualtar, 4710-057 Braga, Portugal
{b7214,b7207}@algoritmi.uminho.pt, joao@dsi.uminho.pt, {alex,macedo,bruno.dias,costa}@di.uminho.pt
http://algoritmi.uminho.pt

Abstract. *VANETs (Vehicular Ad Hoc Networks)* are networks of moving vehicles equipped with devices that allow spontaneous communication. Developing collaborative applications for *VANETs* has currently an increasing popularity in the *Intelligent Transportation Systems (ITS)* domain. This paper proposes a *Platooning Management Protocol (PMP)*, whose implementation and testing is carried out by means of simulation, using the *V2X Simulation Runtime Infrastructure (VSimRTI)* framework (coupling *Simulation of Urban MObility (SUMO)* and *Network Simulator 3 (ns-3)*). Results show that *PMP* works in a efficient manner: maneuvers happen during an acceptable time interval, the proposed communication requirements are met and the lane capacity is increased.

Keywords: Platooning · ITS · Simulation · VANETs

1 Introduction

ITS consist of an intricate set of technologies applied to vehicles and infrastructures that ensure an efficient and smart usage of the roads in general, which potentially improve safety, efficiency and productivity or even decrease levels of pollution. *ITS* enable the rise of several applications relying on the exchange of information between vehicles themselves and infrastructures, allowing drivers to make smarter driving choices. The goal of this work is to develop and test a *PMP* that defines several maneuvers to allow platooning (create, join, leave, merge and dissolve), including the set of messages that allow their operation. The structure of this paper is as follows: first, the state of the art regarding *ITS* application development is presented. Next, the *PMP* is introduced, analyzed and tested. The simulation environment is also discussed, along with the results obtained from the simulations.

2 Related Work

This section provides a brief overview on available publications that cover subjects related to V2X applications, specially advanced applications. The work in

© IFIP International Federation for Information Processing 2017
Published by Springer International Publishing AG 2017. All Rights Reserved
Y. Koucheryavy et al. (Eds.): WWIC 2017, LNCS 10372, pp. 174–185, 2017.
DOI: 10.1007/978-3-319-61382-6_14

[1] presents a *Cooperative Adaptive Cruise Control (CACC)* system that aims to reduce significantly the gaps between the vehicles, taking advantage from information exchanged using *Dedicated Short-Range Communications (DSRC)* wireless communication. In [2] is presented a *CACC* implementation at the *Grand Cooperative Driving Challenge (GCDC)*, based on *Vehicle to Vehicle (V2V)* communication. In [3], the interference of non-automated vehicles, when a given vehicle is joining a platoon is analyzed. It is defined a protocol that supports the join maneuver and it is validated using *PLEXE* from *Vehicles in Network Simulation (VEINS)*. A *CACC* management protocol based on IEEE 802.11p communication, including three basic maneuvers (merge, split and lane change) is presented in [4]. In [5], communication strategies for *Platooning* are investigated and compared to typical beaconing protocols, resorting to *PLEXE*. In [6], an application that aims to advise danger on emergency situations on *VANETs* resorting to IEEE 802.11p is proposed. Additionally, there are some important projects focused on the study of advanced *ITS* applications, such as *COMPANION*, *iGAME* or *SARTRE*.

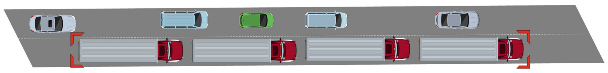


Fig. 1. Platoon of trucks

3 Platooning Management Protocol

Platooning is a solution that allows vehicles to travel very close to each other in groups with automated velocity and steering control. Driving in *platoons* with automatic control enables the enhancement of safety, traffic flow and highway capacities, while providing drivers with a more convenient and comfortable driving experience. Furthermore, it helps to save energy and fuel, while reducing emissions [7,8]. Figure 1 illustrates a platoon of trucks in an highway. The simplest way of implementing *Platooning* is through the use of *V2V* communication, where vehicles only share information with their immediate predecessor. More advanced solutions disseminate information from vehicles that are not in line of sight, providing the driver with situational awareness feedback. Vehicles possessing group information in advance helps to predict the behavior of the platoon. *Platooning* requires a very efficient *PMP* that specifies all the required maneuvers and proper communication behaviors. The proposed *PMP* is described next, including a description of the maneuvers and the specification of their requirements based on European standards.

3.1 Maneuvers

The **Create** maneuver starts when a given vehicles tries to join a platoon but there are no available strings around him. The process of creating a platoon is: (i) *Leader* vehicle starts a new *Platoon*; (ii) *Leader* vehicle propagates the *Platoon* existence, broadcasting its *ID* every second.

The **Join** maneuver is triggered when a vehicle wants to join a *platoon*. An important aspect of the *Join* maneuver is the string ordering. The simplest solution is to make vehicles join the *platoon* tail. Allowing vehicles to join in any position, enables the string to be ordered by several parameters: e.g. braking performance. In these cases, vehicles open a gap that allows the joining vehicle to merge, which requires more coordination. A vehicle is able to join a *platoon* if the string does not exceed its maximum length and if no other maneuver is occurring. The joining process is: (i) *Joiner* sends a periodical *Join Request broadcast*; (ii) *Leader* responds with a *Join Acknowledgment* if it's possible to join. Otherwise, it responds with a *Join Reject*; (iii) *Joiner* moves to the correct position to change lane and informs the *Leader* with a *Distance Achieved* message; (iv) *Leader* notifies the *Followers* to open up a gap, with a *Adjust Gap* message (unless the joining is by the rear); (v) *Followers* notify the *Leader* when the adjusting process is completed with *Adjust Gap Acknowledgments*; (vi) *Leader* sends a *Start Maneuver* message, informing the *Joiner* that the maneuver can be accomplished; (vii) *Joiner* changes lane and enters automatic mode, notifying the *Leader* with a *Maneuver Completed* message. viii) *Leader* sends a *Platoon Update* message for all *Followers* with updated information.

The **Leave** maneuver is initiated when a *Follower* needs to exit the *platoon*. It informs the *Leader* and waits for its response, before assuming manual control. Only one vehicle may leave the platoon at a time and only if the other followers have confirmed to adjust their gap. The maneuver steps are: (i) *Follower* sends a *Leave Request*; (ii) *Leader* orders *Followers* to open a gap with *Adjust Gap* messages; (iii) *Followers* acknowledge their adjustments, resorting to *Adjust Gap Acknowledgment* messages; (iv) *Leader* returns a *Start Maneuver* message for the *Leaver*; (v) *Leaver* shifts to manual driving and changes lane; (vi) *Leaver* notifies the *Leader* with a *Maneuver Completed* message; (vii) *Leader* notifies *Followers* that the maneuver is finished with *Platoon Update* messages.

The **Dissolve** maneuver happens when the *Leader* decides to disassemble the string. The *Leader* may only dissolve after all *Followers* acknowledge the command. The steps are: (i) *Leader* sends a *Dissolve Request*; (ii) *Followers* enter manual driving mode and send a *Dissolve Acknowledgment* to the *Leader*; (iii) When all *Followers* respond (if any), the *Leader* dissolves the *platoon*.

The **Merge** maneuver consists on joining two *platoons*. This maneuver is only possible if the size of the *platoons* is less than the maximum length and the process is initiated by the *Rear Leader*. The following steps show how the *Merge* maneuver is performed. The front *Leader* and *platoon* are referred as *Leader A* and *Platoon A*, while the rear *Leader* and *platoon* are referred as *Leader B* and *Platoon B*: (i) *Leaders* send *Merge Requests* every 10 s; (ii) *Leader A* receives the request and responds with a *Merge Acknowledgment*; (iii) *Leaders* exchange *Platoon Info* messages with information regarding their *platoons*; (iv) *Leader A* sends a *Adjust Gap* message to *Leader B*; (v) *Leader B* moves *Platoon B* to the rear of *Platoon A*; (vi) *Leader B* acknowledges the distance with a *Adjust*

Gap Acknowledgment message; (vii) *Leader B* sends a *New Leader* message to its *Followers*; (viii) *Leader B* assumes a *Follower* role.

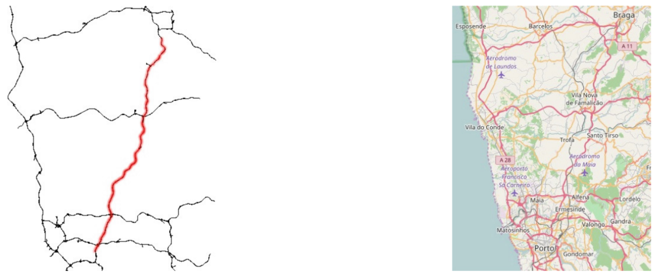
3.2 Platooning Requirements

The most important requirements for the *platooning* application are based on the **ETSI TR 102 638** standard [9], which provides the main requirements for a *Co-operative vehicle-highway automation system (Platoon)* use case. The *latency* is defined to have a maximum value of *100* ms, the relative *position accuracy* should be better than *2* m, and the platoon group messages should have a minimum frequency of *2* Hz. The vehicles should be prepared to transmit *V2V* messages in *unicast* and *broadcast* mode.

4 Simulation Deployment

The development of efficient *VANETs* systems requires the determination of its main properties and consequent evaluation of its performance. Performing field tests is a tough challenge: the large number of existent vehicles and scenarios makes it harder to collect data, the development of prototypes is expensive, etc. Simulation is a popular solution to evaluate the performance of *ITS* systems - tests are easily repeated and researchers are able to control parameters, configurations, conditions and input data. However, it normally assumes the use of simpler models, which may reduce the system realism. To perform a proper simulation of *VANETs*, both a traffic and a network simulator are required. Network simulation is one of the most prominent evaluation methods in computer networks, and *ns-3* and *OMNeT++* are two major tools used to model realistic V2X environments. Some of the most important tools used in to simulate mobility and traffic are *SUMO*, *VISSIM* and *VanetMobiSim*. Additionally, there are some tools that allow their interconnection, which enables them to interact with each other in a transparent way, such as *VEINS*, *iTETRIS* or *VSimRTI*.

The first step towards deployment is the choice of the simulation tools. Among all solutions, the most complete and realistic way is through the use of coupled simulators. According to [10], *iTETRIS*, *VEINS* and *VSimRTI* are strong solutions and there is no clear winner, since they all cover the required aspects for *VANETs* simulation. Despite *iTETRIS* potential, the project is finished and there is no available support. *VEINS* simulator already includes a platooning module denominated as *PLEXE*. However, since *VSimRTI* is more flexible on the choice of the simulators and allows the use of *JAVA* programming, the choice falls for *VSimRTI*. To simulate transportation, the choice is *SUMO*, since it is able to support detailed representations of large scale traffic scenarios. To support accurate simulation of communication for *ITS* systems, the choice is *ns-3*, since it includes all models to reproduce functionalities and protocols for the ITS communications stack. According to [11], a very suitable wireless technology available today to interconnect vehicles is *IEEE 802.11p*. Since it was specifically built for vehicular environments, it was the technology chosen to allow communication. The intra-platoon distance was based on the values used in [4] for homogeneous vehicles. Another important aspect of this *PMP* is that maneuvers can happen at any point but only one maneuver is allowed at a time. Allowing more than one maneuver would make the management task extremely complex. Finally, and regarding security concerns, a basic and simple security mechanism for messaging exchanging was implemented. The vehicles are statically assigned one public key pair and one symmetric key and all public keys are pre-shared between the vehicles. The symmetric cipher algorithm used was *AES* (128 bits key) and for public key scheme it was used a *RSA* (1024 bits key). The exchanged message is composed of the encrypted payload and the signature. The signature is obtained using SHA-256 and the RSA key.



(a) Simulation Map (b) Real Map

Fig. 2. Braga-Porto highways

4.1 Simulation Scenario and Decisions

This subsection describes some important deployment decisions. The selected simulation scenario (illustrated in Fig. 2) area comprises the highways that connect the Portuguese cities of *Braga* and *Porto*, obtained from a *Open Street Map (OSM)* file using the *osmosis* tool. The vehicles in the simulation that are running the *PMP* are *Trucks*, while other vehicles are simple *Passenger* cars. Additionally, there are some *reference Trucks* that do not run the *PMP*, to allow a comparison between them and *Platooning-enabled Trucks*. The application uses only one of the available ITS-G5 service channels and there are no additional applications running, which implies that there are no congestion problems, nor interferences. To broadcast information on *platoons*, *Leaders* use *Geocast* messages, and *V2V* Unicast on requests/replies between *Leaders* and *Followers*. When receiving a Join Request, the *Leader* computes a performance value and the position the vehicle should assume on the string based on its performance, sending this information back to the requester. Before starting dismembering the *platoon*, *Leader A* goes through a phase of velocity fluctuation, to test the *Followers* ability to adjust their velocities based on the messages received from the *Leader* every 100 ms. When the *Dissolve* process is complete and all vehicles have left *platoon A*, the simulation reaches its final state. The *ns-3* configuration was built to include values that match the communication characteristics of the *Cohda MK5 On Board Units (OBUs)*, as seen on Table 1 to allow more realistic results regarding the communications. In [12], the advised size of the string is 15 vehicles. However, two different sizes were defined for the two different *platoon* strings created in the simulation (15 and 3 vehicles). Additionally, the two strings travel with different speeds: 20 and 25 m/s, respectively. This happens so that it is easier to test the cases where the string is already full, and to study the impact of the speed on the results. The vehicles running the application are homogeneous, defined with the following parameters: *Class* - Truck; *Maximum Acceleration* - 1.1 m/s^2; *Maximum Deceleration* - 4.0 m/s^2; *Maximum Speed* - 36.11 m/s; *Length* - 16.5 m; *Width* - 2.55 m.

Table 1. NS-3 WiFi configuration established for Cohda MK5 OBU

Wifi configuration	
Wifi Mac	ns3::OcbWifiMac
Physical mode	OfdmRate6MbpsBW10MHz
Wifi manager	ConstantRateWifiManager
Received signal energy threshold	−99 dbm
Received signal energy threshold (CCA Busy)	−85 dbm
Transmission gain	10.0 dB
Reception gain	−16.0 dB
Maximum available transmission level	23 dbm
Minimum available transmission level	−10 dbm
Transmission power levels	Step 0.5 dB
Signal-to-Noise-Ratio loss	3.2 dB

5 Results and Analysis

In general, the behavior of the vehicles in *ITS* applications simulation tends to be extremely dynamic: typically, the mobility simulator runs with hundreds of vehicles equipped with applications at a given penetration rate and they follow computer generated routes. However, the simulation deployment on this work was proposed to be slightly more static, in the sense that the dissolve and leave maneuvers start timings and routes are predefined. Furthermore, the vehicles running the application and the platoon string they should join is also predefined. This causes the lane capacity results to be almost the same from simulation run to simulation run. Still, given the fact that the remaining maneuvers behavior is random, essentially due to the fact that the joining process will cause vehicles to join random positions on the string, independent runs will generate independent results. Although not being the most desired situation, this happens due to the high difficulty level of controlling and evaluating the vehicles and application behavior during the complex maneuvers that result from the *PMP*. The *PMP* generates result logs regarding maneuver durations, the exchanged messages, vehicles distance, speed values and lane capacity, which are discussed next.

5.1 Lane Capacity

The first results that can be obtained from the use of *Platooning* is increased lane capacity. According to [13], the typical lane capacity value is $C = 35$ and the formula to compute it is:

$$C = v \times \frac{n}{ns + (n-1)d + D} \quad \text{vehicles/lane/min}, \tag{1}$$

where v is the steady state speed (*meters/min*), d the intra-platoon spacing (*meters*), D the inter-platoon spacing (*meters*), s the vehicle length (*meters*) and n the number of cars composing the string. In the *PMP* deployment, the maximum theoretical capacity is $C = 71.88$ vehicles/lane/min. To compare this analytical value to the simulation, the capacity was logged every second and the mean results are presented on Fig. 3. As expected, the capacity is typically lower than the theoretical value, due to the dynamic values of the string (e.g. speed, number of vehicles). Also, these values differ in both *platoons*, which implies that the capacity values are different between them.

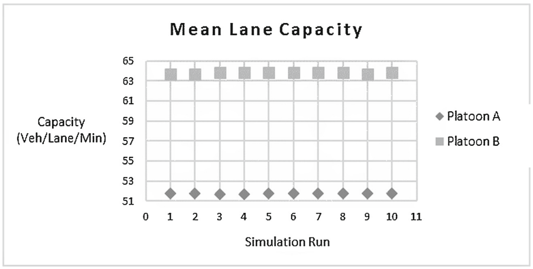


Fig. 3. Lane Capacity during simulation time

5.2 Maneuvers

This subsection describes the behavior of the *Platoons* during the application runtime, focusing and discussing on the duration of the maneuvers.

Join Maneuver. With exception to the *Leaders*, every vehicle must perform a *Join* maneuver in order to become part of a *platoon*. The mean duration of these maneuvers are presented in Table 2. The *Waiting* value represents the amount of time a vehicle awaits until it receives a *Join Acknowledgment* after sending a *Join Request*. The first conclusion one draws is that joining a *platoon* at the rear is faster than joining a *platoon* by the side, where other

Table 2. Join maneuver mean durations

Join Type	Operation (s)	Waiting (s)	Total (s)
Rear	54.6	30.3	69.0
Side	70.6	43.6	114.2
Mean	60.3	37.5	85.1

vehicles are required to adjust their gaps, making the maneuver last longer. Still regarding the *Join* maneuver, it is also important to analyze the duration of a negative response to a *Join Request.* A reject situation happens when a platoon is already at his maximum size. On average, between the request and response, only 62.2 ms elapse. This happens because the *Leader* is able to respond almost immediately if a given requester is able (or not) to perform the maneuver, even if another is already occurring.

Leave Maneuver. In side *Leaves*, vehicles that follow behind the leaving vehicle are required to open up gaps, so the maneuver is safer. In rear *Leaves*, the leaving vehicle simply changes lane and leaves. For these reasons, there is a huge difference in the duration of the maneuver, depending on the type: on average, side *Leaves* last 23.3 s, while rear *Leaves* last 1.0 s (on the simulator, the operation is immediate, making the result unrealistic).

Adjusting Gaps. Although these situations are not qualified as maneuvers, they play an important role on their duration times. On average, the *Adjust Gap* operation lasts 12.6 s on *Joins* and 17.4 s on *Leaves.*

Merge Maneuver. The merge maneuver (exemplified on Fig. 4) can be divided in three steps: exchanging the information between *Leaders*, the adjusting gap operation and the *New Leader* information dissemination. On average, the *Merge* maneuver was accomplished with a similar duration to the operations in *Join maneuvers* (a *Merge* operation can almost be seen as one *Leader joining* another *platoon*). The adjust gap operation takes most of the time (62.7 s on average) while the maneuver set up and finishing steps are very fast (72.0 ms).

Dissolve. The *Dissolve* maneuver duration time is estimated from the moment when the *Dissolve Request* is issued until the last *Dissolve Acknowledgment* is received. The maneuver performs quickly -3.8 s. This happens because, on the simulator, vehicles acknowledge the request and start manual driving instantaneously, making the result somewhat unrealistic. Hence, these durations only concern the communications part of the maneuver.

5.3 Messages

The results regarding the delay of the messages are now presented on Table 3 and Fig. 5. The calculated values present a very satisfactory result. They indicate that the *PMP* is mostly able to deliver the messages on time, and it is able to conform with the communication requirements of the *PMP* - the messages are

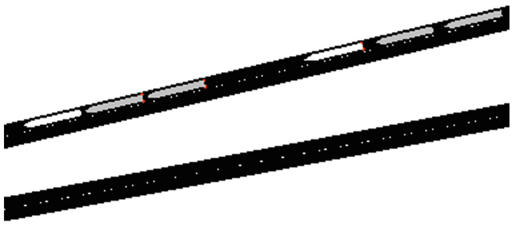


Fig. 4. Merge maneuver

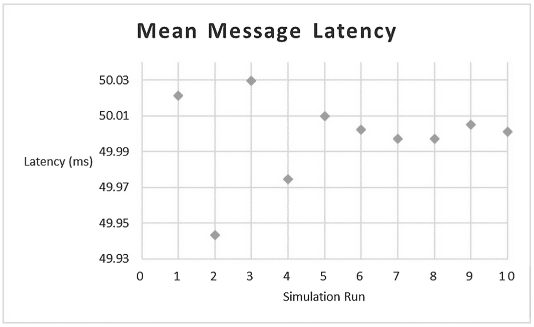


Fig. 5. Mean message latency values

Table 3. Statistic results from messages exchange

Messages latency (ms)	
Mean	50.0
Median	49.1
Standard Deviation	28.7
Minimum	0.6
Maximum	100.9
Mean Number of Messages	
709181	

able to be generated every 100 ms and the requirement for the maximum delay allowed (100 ms) can be fulfilled. This also means that the impact of the security mechanisms used to encrypt and sign the messages is almost unnoticeable (content is secure and the communication is not compromised). However, there are on average 1073 messages that are delivered with a latency greater than 100 ms. Despite the results not being perfect, the results are still acceptable, since these messages represent a universe of only 0,2%.

5.4 Distances and Speed

The *Distances* log allows the analysis of the *distance to go* value - the distance towards the computed minimum gap (when positive, the vehicles should accelerate, and vice-versa). Figure 6 illustrates the mean values of the *distance to go* for all followers in each simulation run (*Leaders* do not keep distances). These values start to be recorded during the forced fluctuation phase, in order to study the platoon stability. The error bars on the chart represent the standard deviation values. During the speed fluctuating phase, the *distance to go* is stable, which means that the vehicles can rapidly adjust their speed to reach the correct position based on the information sent by the *Leader* in the frequent *Platoon Group* messages. The *Speed* log also allows the study of the platoon stability during the same fluctuation phase. Figure 7 shows the average of the difference between the actual speed and the required speed for all followers in each simulation run. Vehicles can smoothly adjust their speed to meet the *Leader* speed. Another important aspect of the speed values that the vehicles assume during the simulation is that they are intimately related to *distance to go* values. This

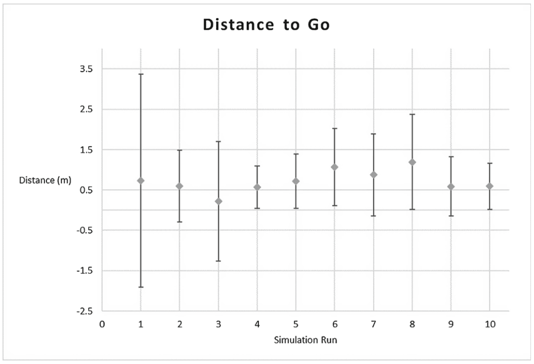


Fig. 6. Vehicle's *distance to go* value during simulation

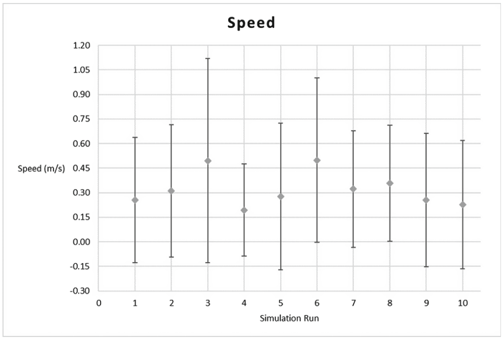


Fig. 7. Vehicle's *speed* deviation during simulation

means that the vehicle must increase its speed when the *distance to go* value its positive and vice-versa.

6 Conclusions and Future Work

ITS are systems that aim to assure a more efficient and improved usage of the roads by controlling traffic operations and drivers behavior. *ITS* enables the creation of many applications that use the information from vehicles and infrastructures to implement better driving practices and to improve traffic flow.

This paper discusses related work and general *ITS* simulation important tools. Then, the *PMP* is presented, containing the description of the maneuvers, general considerations and some important requirements (based mostly on *ITS* standards). The second part details the process of deploying the *PMP* implementation and obtaining the results from the simulation runs. The application was implemented using *VSimRTI*, coupling *SUMO* and *ns-3*. The choice of the tools was not a difficult decision, taking into account that these tools are proven to be very powerful and well-established within the research community. The deployment section describes the simulation scenario and associated decisions as well. The deployment was not an easy task - from the study on the state of the art, it was possible to conclude that the application is immensely complex and sometimes very subjective. For each particular problem that arises from *Platooning*, there are usually a lot of different proposed solutions. This seems to be an indicator that the *Platooning* specification is prone to ambiguity. Also, the deployment of the application required a lot of effort to overcome some lack of "intelligence" the chosen tools present - the constant trade-off between a more realistic application and the simulation performance caused some difficulties to evaluate the application behavior.

From the simulation, it is possible to conclude that the *PMP* works efficiently - maneuvers durations are within an acceptable interval, messages meet the hard communication requirements and the lane capacity is proven to be increased.

Regarding future work, we will consider gathering fuel consumption and emissions information using a more powerful emission model than the open source models available on *SUMO*. Additionally, we will consider using models that take into account road slopes. Finally, it may be interesting to test the protocol on a real implementation.

Acknowledgment. This work has been sponsored by the Portugal Incentive System for Research and Technological Development. Project in co-promotion n° 002797/2015 (INNOVCAR 2015–2018), and also by COMPETE: POCI-01-0145-FEDER-007043 and FCT - Fundação para a Ciência e Tecnologia within the Project Scope: UID/CEC/00319/2013.

References

1. Milanés, V., et al.: Cooperative adaptive cruise control in real traffic situations. IEEE Trans. Intell. Transp. Syst. **15**(1), 296–305 (2014)
2. Guvenc, L., et al.: Cooperative adaptive cruise control implementation of team mekar at the grand cooperative driving challenge. IEEE Trans. Intell. Transp. Syst. **13**(3), 1062–1074 (2012)
3. Segata, M., et al.: Supporting platooning maneuvers through IVC: an initial protocol analysis for the JOIN maneuver. In: 11th Annual Conference on Wireless On-demand Network Systems and Services (WONS), pp. 130–137. IEEE (2014)
4. Amoozadeh, M., et al.: Platoon management with cooperative adaptive cruise control enabled by VANET. Veh. Commun. **2**(2), 110–123 (2015)
5. Segata, M., et al.: Toward communication strategies for platooning: simulative and experimental evaluation. IEEE Trans. Veh. Technol. **64**(12), 5411–5423 (2015)
6. Fazio, P., et al.: Vehicular networks and road safety: an application for emergency/danger situations management using the WAVE/802.11p standard. Adv. Electr. Electron. Eng. **11**(5), 357 (2013)
7. Al Alam, A., et al.: An experimental study on the fuel reduction potential of heavy duty vehicle platooning. In: 13th International IEEE Conference on Intelligent Transportation Systems (ITSC), 2010, pp. 306–311. IEEE (2010)
8. Alam, A.: Fuel-efficient distributed control for heavy duty vehicle platooning. In: KTH Royal Institute of Technology (2011)
9. ETSI: ETSI TR 102 638 V1.1.1 Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions (2009)
10. Sommer, C., Häarri, J., Hrizi, F., Schüunemann, B., Dressler, F.: Simulation tools and techniques for vehicular communications and applications. In: Campolo, C., Molinaro, A., Scopigno, R. (eds.) Vehicular Ad hoc Networks, pp. 365–392. Springer, Cham (2015)
11. Ribeiro, B., Santos, A., Nicolau, M.J.: A survey on vehicular communication technologies. In: Intelligent Environments 2016, vol. 21. Ambient Intelligence and Smart Environments, pp. 308–317. IOS Press (2016)
12. Robinson, T., et al.: Operating platoons on public motorways: an introduction to the SARTRE platooning programme. In: 17th World Congress on Intelligent Transport Systems, vol. 1, p. 12 (2010)
13. Varaiya, P.: Smart cars on smart roads: problems of control. IEEE Trans. Autom. Control **38**(2), 195–207 (1993)