A Tool Offering Steady-State Simulations for VANETs

Miguel Báguena, Sergio M. Tornell, Álvaro Torres, Carlos T. Calafate, Juan-Carlos Cano, Pietro Manzoni
Department of Computer Engineering,
Universitat Politècnica de València
Camino de Vera S/N 46022, Valencia, Spain.
mibaal@upvnet.upv.es, sermarto@upv.es, atcortes@batousay.com, {calafate, jucano, pmanzoni}@disca.upv.es

Abstract— **Without realistic vehicle mobility patterns, the evaluation of communication protocols in vehicular networks is compromised. Moreover, in order to ensure repeatability and fairness in vehicular simulations, researchers require simulation tools that allow them to have a complete control of simulations. In this paper we present VACaMobil, a Mobility Manager for the OMNeT++ simulator which offers a way to create complex scenarios with realistic vehicular mobility by allowing to define the desired average number of vehicles, along with its upper and lower bounds, which are maintained throughout the simulation. We compare VACaMobil against other commonly used methods which also generate and manage vehicular mobility. Results expose some flaws of those basic tools, and shows that VACaMobil behaves significantly better. The harmful impact on communication protocols when using common tools is also quantified, revealing VACaMobil as a necessity for current research.**

Index Terms— **Vehicular Networks, Mobility patterns, Simulation Tool, SUMO, TraCI, Veins, VANET.**

I. Introduction

THE REPRODUCIBILITY of experiments is a major issue when evaluating smart communication protocols and algorithms, especially over Vehicular Ad-hoc NETworks (VANETs). In [1] the authors provide a complete review of the minimum set of parameters that should be identified in order to allow other researchers to reproduce simulation experiments. They pointed out several key parameters, such as the simulated hardware, the network simulator, the scenario, and the road traffic simulator. However, regarding node mobility, there are other parameters that have been mostly ignored by the research community: the traffic density and the traffic demand.

As other authors pointed out in previous studies, mobility models [2] and the chosen scenario [3], as well as the node density, heavily influence the final network performance. However, since mobility generators and road traffic simulators are often difficult to configure, the simulated node density and distribution may depend on complex data that are usually not included in the published academic results, thereby compromising reproducibility.

Several recent patents concerning inter-vehicle communication have been published [20], [21] and [22]. Moreover, the industry needs appropriated evaluation methods, which makes simulators an interesting subject for patent submissions [23] and [24].

In this paper we present VACaMobil (VANET Car Mobility manager), a mobility manager module for the OMNeT++ simulator which, to the best of our knowledge, is the first tool able to generate SUMO [4] driven nodes in a vehicular network while ensuring the stability of certain user-defined parameters, such as the average, maximum, and minimum number of vehicles. These features are especially useful for mid-length simulations (typically one hour) allowing researchers to assume that the vehicle density is stable. At the same time, since our solution is tightly coupled with SUMO through the TraCI interface, it is able to mimic real vehicle behavior. By running in parallel with SUMO, VACaMobil executes the following tasks: (i) it manages when a new vehicle must be introduced in the network, (ii) it assigns a random route from a predefined set to each vehicle, and (iii) it determines which type of vehicle should be added. When using VACaMobil, and given a specific road map, researchers will be able to completely define the network mobility merely by defining the desired average number of vehicles and its upper and lower bounds. Going a step further, our tool also aids researchers at selecting among the different types of vehicles previously defined in SUMO, such as “cars”, “buses”, or “trucks”. This allows researchers to easily define road traffic simulations with heterogeneous vehicles.

The rest of this paper is organized as follows: In section II, we briefly introduce the different methods commonly used by the research community for generating VANET mobility patterns. In section III, VACaMobil is fully described. In section IV, we compare our proposal with the duaRouter and duaIterate.py tools, both included in SUMO. Section V demonstrates the impact of the mobility model in the performance of network protocols. In section VI we present the main guidelines for using the VACaMobil tool

This paragraph of the first footnote will contain the date on which you submitted your paper for review. It will also contain support information, including sponsor and financial support acknowledgment. For example, “This work was supported in part by the U.S. Department of Commerce under Grant BS123456”.

The next few paragraphs should contain the authors’ current affiliations, including current address and e-mail. For example, F. A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305 USA (e-mail: author@ boulder.nist.gov).

successfully. Finally, section VII presents our conclusions and some future plans to improve VACaMobil.

II. A REVIEW OF EXISTING MOBILITY GENERATORS FOR VANETS

Before presenting the details of our proposal, we analyze some of the methods commonly used to obtain suitable mobility patterns in urban vehicular scenarios. We have analyzed several papers published during the last few years, most of them published in conferences and journals related to Intelligent Transportation Systems. Early approaches relied on overly simple mobility models merely based on random mobility. Since these simple models do not represent vehicle mobility properly, other mobility models have recently been developed based on real-world traces, and also on artificial mobility models from the field of transportation and traffic science. In this section, we briefly describe the most relevant works.

A. Random Vehicle Movement

At the beginning of the previous decade, the “Random Way-Point" mobility model was extensively used in Mobility Ad-Hoc NETwork (MANET) research. However, in 2003, the authors in [5] demonstrated how harmful the Random Way-Point mobility model really is in terms of result representativeness. Moreover, the negative effects described in this work become even worse when simulating VANETs. Later on, some other authors extended the “Random Way-Point" mobility model by restricting the mobility of nodes to a map layout, as in [6]. However, this improvement does not solve the majority of the “Random Way-Point” model problems stated previously.

In our research group we developed a tool called “CityMob" [7]. CityMob allows users to create random vehicular mobility patterns restricted to a grid. It also adds support for downtown definition, where a downtown is a region inside the simulated map which concentrates the majority of the selected routes along the simulation. Although CityMob represents a significant improvement compared to non-restricted mobility models and random mobility models, it also presents some problems; the most important one is that vehicular mobility is not influenced by other vehicles, i.e. two different vehicles can be at the same physical location, and no minimal distance between vehicles is required. Moreover, vehicles do not change their speed during a trip. However, in the real world, vehicles continuously change their speed according to traffic conditions and road characteristics. Last but not least, vehicles keep moving throughout the whole simulation, which especially influences the performance of protocols that keep data stored in buffers. The research community quickly realized the problems derived from inaccurate simulation patterns, and started to work using alternative methods to obtain suitable mobility traces.

B. Real Mobility Traces

Compared to the use of random mobility, real traces present a clear improvement. Such traces are usually obtained from a certain set of nodes, e.g. from taxis in the city of Shangai [8]. Mobility traces can be obtained by tracking the mobility of nodes using On-Board units, as in [8], or by using road-side equipment, as in [9]. Although real traces represent the most realistic mobility patterns, we cannot obviate the fact that mobility of tracked nodes is highly influenced by other untracked vehicles, e.g. taxis’ mobility is influenced by other users on the road whose movement is not reflected in the collected traces. Moreover, real traces lack the flexibility to allow for an exhaustive evaluation of VANET protocols, e.g. changing the vehicle density without modifying their speed is clearly unreal.

C. Assisted Traffic Simulation

The restrictions of real traces can be overcome, with almost no loss of realism, by using mobility models taken from the field of transportation and traffic science. Several road traffic simulators are widely used among the VANET research community. One of the most widely used mobility generators is SUMO [4]. When simulating traffic mobility for VANETs, not only the vehicles’ behavior is important, but also the traffic demand. SUMO allows defining traffic demand in two different ways: trips and flows. The former defines only a vehicle, its origin and its destination, while the latter defines a set of vehicles which execute the same trip. SUMO currently provides several tools to generate traffic demand:

- randomTrips.py: A random trip generator. This tool generates a trip every second having a random origin and destination. It does not check if the origin and destination are connected, or whether the trip is possible.
- duaRouter: A Dijkstra router. Given a file with trips and flows, this tool generates the actual traffic demand, expressed in vehicles with an assigned route. Routes are calculated using the Dijkstra algorithm, and every unconnected trip is discarded.
- duaIterate.py: This Python script will produce a set of optimal routes from a trip file, i.e. all the nodes will follow that route which minimizes the total trip-time for all nodes. This tool repeats a routing-simulation loop until optimal routes are found.

Authors have used these tools in order to generate traffic demands for SUMO. The most simplistic one is to define different flows inside the network. Although drivers usually move from certain districts to others, following patterns associated with their working and living places, defining the traffic only by creating fixed flows lacks realism, as we can see in [10] where only a few flows are defined by the user. Another common approach is to generate random trips using randomTrips.py. This approach presents the problem that only one vehicle is introduced every second, which leads to long transitory periods until the network reaches a steady state. A more sophisticated traffic demand generation strategy is presented in [11], where a predefined number of vehicles following random routes are randomly placed at the beginning of the simulation. Following this trend, in previous works we used C4R [12], which is a software developed by our group to

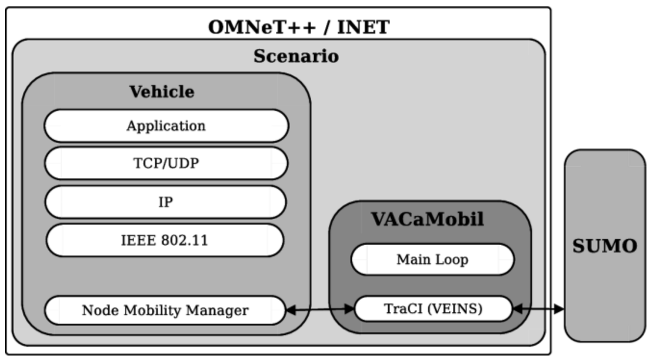


Fig. 1. The VACaMobil module within the OMNeT++ simulation framework.

automate the task of generating random vehicles with random routes at random places. To the best of our knowledge, the work presented in [13] is the only one using the duaIterate.py script to generate a "stable and optimal distribution of flows". This type of traffic definition presents a problem: the trip duration cannot be predicted before running the simulations, and, as a consequence, there is no way to ensure, or even determine, if the road traffic simulation will last until the end of the network simulation. As stated in previous work, this lack of realism and generality in mobility patterns can lead to biased results [2].

D. Bidirectionally Coupled Network and Traffic Simulations

In [14] its authors go a step further and present a new simulation framework called Veins, which includes the TraCI interface to allow the network simulator to interact with the traffic simulator running in parallel. Although it presents much novelty and offers a lot of possibilities for VANET simulation, the authors do not address the traffic demand generation problem. The main characteristics and benefits of this tool were highlighted in [15]; in addition it is one of the main elements of our VACaMobil module as described below.

III. VACaMobil Mobility Manager

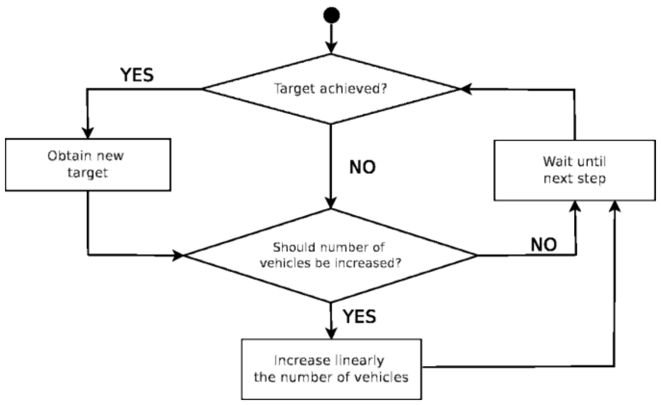
In this section we present our proposed tool to generate realistic mobility for VANETs. We also provide some important implementation details. All of these characteristics were developed having two basic objectives in mind: achieving realistic- mobility scenarios with a user-defined node density, and simplifying the process of simulating vehicular networks under the desired conditions.

A. Implementation Details

We have implemented VACaMobil as a superset of the Veins simulation framework for OMNeT++. This scheme allows us to take advantage of the current TraCI implementation provided by Veins.

Due to the high modularity of the OMNeT++ simulator, VACaMobil is available for two of the most used network simulation environments for OMNeT++: INET [16] and MIXIM [17]. Both implementations of VACaMobil can be downloaded from our github account[1].

[1] https://github.com/grclab



YES
Target achieved?
Obtain new target
NO
Wait until next step
Should number of vehicles be increased?
NO
YES
Increase linearly the number of vehicles

Fig. 2. VACaMobil algorithm flowchart.

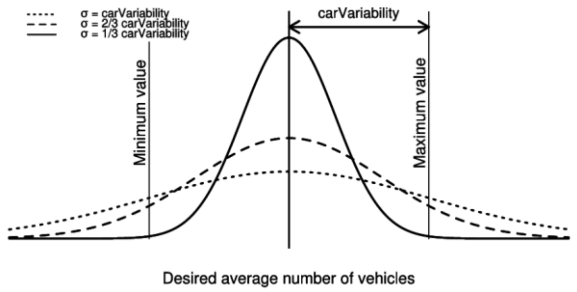


Fig. 3. Distribution of the target number of vehicles.

Figure 1 shows the interaction between the different modules required for VANET simulations when using VACaMobil.

B. Average Number of Vehicles

One of the objectives of VACaMobil is to guarantee a steady number of vehicles throughout the entire simulation. The user can define some degree of variability through the *carVariability* parameter and VACaMobil will ensure that the number of vehicles is always between $average + carVariability$ and $average - carVariability$ values.

To control the current number of vehicles, VACaMobil selects a current target number of vehicles which should be achieved. Then it adds new vehicles, in case the current number of vehicles is smaller than the current target number, or waits until vehicles arrive to their final destination otherwise. A flow chart of this process is shown in Figure 2.

To avoid the insertion of a large number of vehicles in a short period of time, the VACaMobil tool stores the duration of the last period where the number of vehicles decreased, and then takes the same amount of time to insert new vehicles into the network.

The current target number of vehicles is obtained from a normal distribution, whose mean ($\bar{x}$) is the desired average value and whose standard deviation (σ) is equal *to 1/3•carV ariability*.

The value of σ is not arbitrary; it has been selected to guarantee that at least 99% of the values obtained from the normal distribution will be inside the user-defined bounds. Figure 3 illustrates the effect of σ in the normal distribution shape. If we had set σ equal to carV ariability, only 68% of the values obtained from the normal distribution would be inside the bounds defined by the user. On the contrary, by setting the standard deviation to 1/3•carVariability, more than 99% of the values returned by the normal distribution are within the defined bounds. Finally, to deal with those values falling outside the user-defined bounds, we filtered the distribution output, being the final current target number distribution as follows:

$$N = \begin{cases} y = norm(x, \sigma) & if\ x - carVariability < y \\ & \&\, y < x + carVariability \\ x - carVariability & if\ y < x - carVariability \\ x + carVariability & if\ y > x + carVariability \end{cases}$$

By selecting this distribution and setting its standard deviation we obtain a great degree of variability, while avoiding extreme values and ensuring that most of the simulation time the number of vehicles is maintained near the average value desired by the user.

C. *Different Types of Vehicles*

SUMO supports the definition of different types of vehicles, which can have different characteristics such as maximum speed, acceleration and deceleration values. The list of different vehicles can be obtained via TraCI.

VACaMobil allows the user to set different probabilities associated to each type. In this case, every time a new vehicle is generated, we obtain a uniform random value to select the corresponding vehicle type. If no probability is defined for a certain type of vehicle, we assume it is equal to zero. However, if no probability value is assigned to any of the defined vehicle types, only vehicles of the first type obtained via TraCI will be generated. This feature allows users to easily define heterogeneous networks composed by different vehicles.

Although SUMO itself is able to provide this behavior, VACaMobil adds the possibility of easily changing the vehicles' associated probability between different simulation runs. As future work VACaMobil will be able to associate different node roles to different types of vehicles, for example, "buses" could have a bigger transmission power than "cars".

D. *Route Generation*

VACaMobil does not include the ability to dynamically generate random routes. Instead, it includes randomRoutes.py, a script that makes use of two well-known tools included in SUMO (randomTrips.py and duaIterate.py). Thanks to those tools we can generate a large set of random different routes which can be loaded into SUMO.

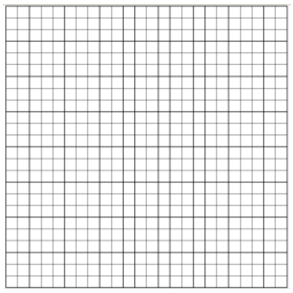
The randomRoutes.py script generates a set of trips between random points of the map by using randomTrips.py, and then it computes the optimal vehicles distribution using duaIterate.py. Finally, it extracts the generated routes and creates a new file containing only routes' definitions.

This method also guarantees that all the defined routes are valid, and that all the vehicles that are inserted into the simulation scenario will eventually arrive to their final destination.

E. *Route Selection per Vehicle*

At startup SUMO loads all the different routes. VACaMobil will retrieve them through TraCI, and randomly selects a new one from the existing set every time a new vehicle is introduced in the network.

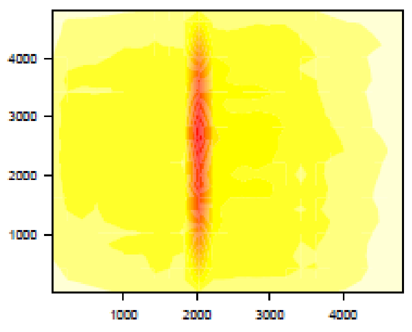
Since routes are defined as a list of consecutive edges, vehicles are introduced in the network at the beginning of the first edge. It is impossible to insert a new vehicle when another previously created vehicle is already located at the beginning of the selected route. To minimize the impact of this restriction, VACaMobil first tries to insert the vehicle in any of the lanes of the route's first edge. If the previous step does not succeed, VACaMobil selects a new route and tries it again until it finds a free place to insert the vehicle. It may occur that none of the loaded routes allows VACaMobil to introduce a new vehicle. In such a case VACaMobil assumes that its main objective cannot be satisfied and the simulation is aborted. This situation typically occurs at the beginning of the simulation, when VACaMobil must introduce a large number of vehicles in a short time period. To avoid interrupting the simulation, the user can modify a variable called warmUpSeconds, which defines the time period at the beginning of the simulation during which VACaMobil requirements are relaxed. During this warm-up time, VACaMobil introduces only a fraction of the desired number of vehicles in every step of the simulation, avoiding the problem previously described. After the warm-up, time VACaMobil ensures that the number of vehicles in the simulation is equal to the value defined by the user.


(a) Synthethic map.

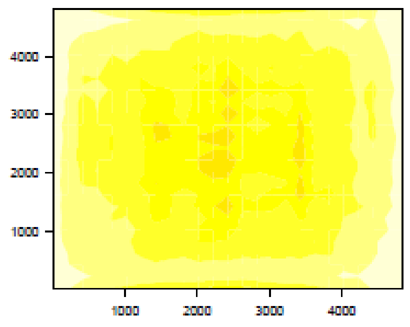
(b) Moscow residential area.

(c) Washington.

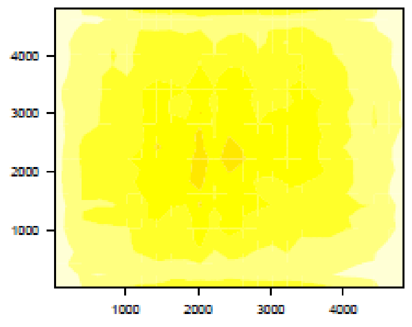
(d) Milano.



(a) duaRouter



(b) duaIterate.py



(c) VACaMobil

Fig. 5. Heat map for the Manhattan scenario

F. *Repeatability, Scalability, and Usability*

Thanks to the use of the standard random number generators available in OMNeT++, VACaMobil ensures the repeatability of the different scenarios including route and vehicle selection.

Despite the goodness of the characteristics previously presented, the best improvement introduced by VACaMobil is the ability to optimize the researcher's work-flow, as will be detailed in section VI. Currently, if a researcher wants to repeat the simulation N times for a certain vehicle density while varying the vehicle routes to decouple the results from the vehicle mobility, the researcher must create N different route files and ensure that the vehicle density is the same along all the simulations. Moreover, the path of those files must be manually introduced into the OMNeT++ configuration file, which is prone to errors.

When using VACaMobil the researcher can take advantage of one of the most important features in OMNeT++, which allows specifying the number of independent repetitions required for every simulation.

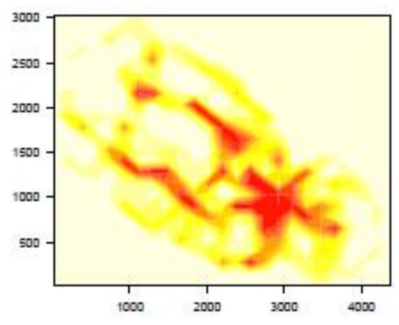
As explained in section VI, VACaMobil also simplifies the process of simulating different amount of vehicles in the network.

IV. EVALUATION

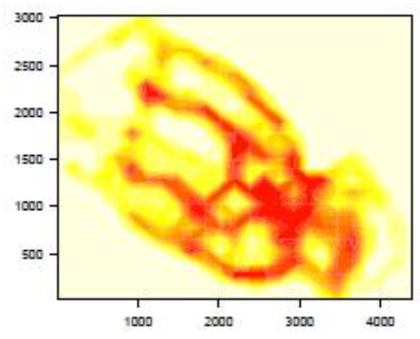
In this section, we compare VACaMobil against the tools currently included in SUMO, i.e. duaRouter and duaIterate.py, that were described in section II. We have selected the following scenarios:

- **Synthetic Manhattan scenario**: We created a road map consisting of a 25 x 25 grid with segments of 200 meters (Figure 4a).
- **Suburban real map scenario:** We extracted a suburban road layout from the OpenStreetMap database. It is a scenario of about 12 km2 from the city of Moscow characterized by long road segments and a low road density (Figure 4b).
- **Urban grid real map scenario:** We extracted an urban road layout from the OpenStreetMap database. It is a scenario of about 6 km2 from the city of Washington DC characterized by long road segments and a high road density (Figure 4c)
- **Urban downtown real map scenario:** We extracted an urban road layout from the OpenStreetMap database. It is a scenario of about 7 km2 downtown area from the city of Milano characterized by short road segments and a high road density (Figure 4d).

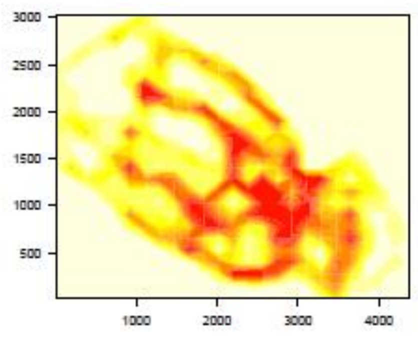
In all the scenarios, the set of random routes provided by VACaMobil is extracted from the traffic demand generated by duaIterate.py. In the following subsection, we compare the vehicle density and its evolution along the simulation time for



(a) duaRouter

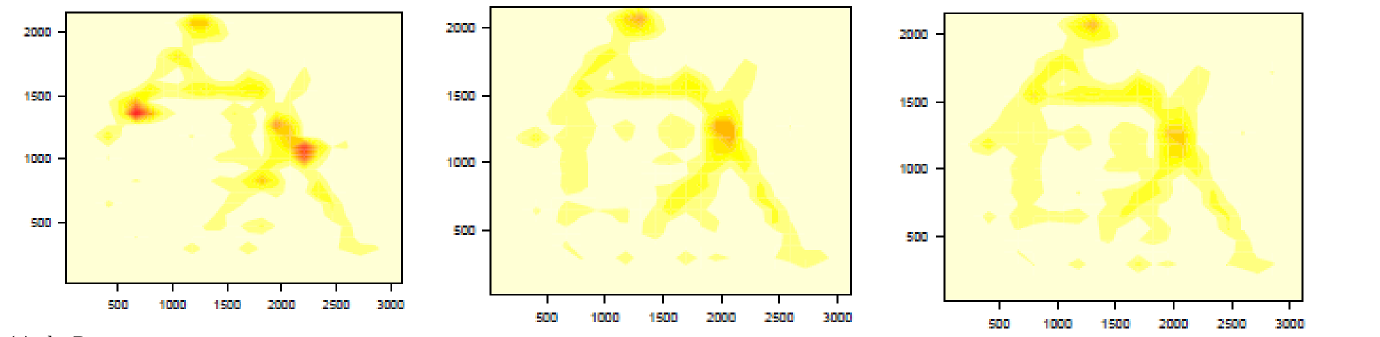


(b) duaIterate.py



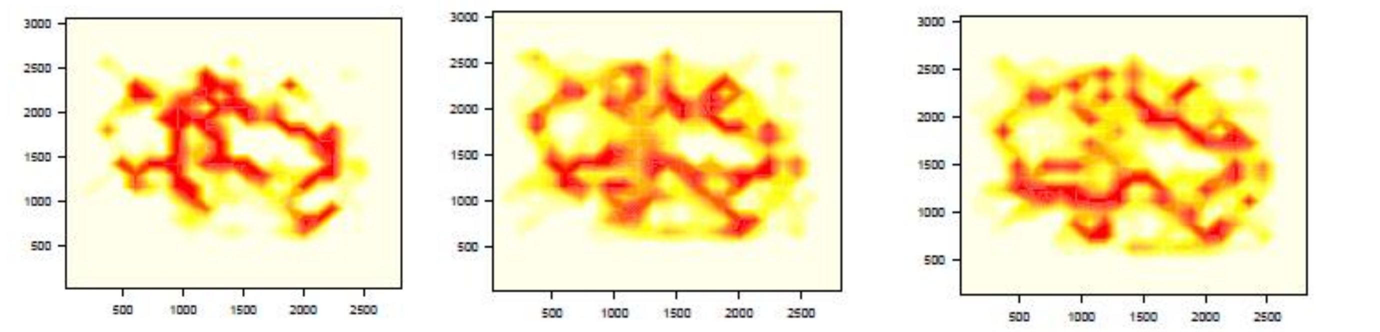
(c) VACaMobil

Fig. 6. Heat map for the suburban scenario (Residential area of Moscow)



(a) duaRouter (b) duaIterate.py (c) VACaMobil

Fig. 7. Heat map for the urban grid scenario (Washington DC)



(a) duaRouter (b) duaIterate.py (c) VACaMobil

Fig. 8. Heat map for the downtown scenario (Milano)

the different tools and scenarios.

A. *Vehicle Distribution Study*

We first evaluate one of the most important issues in vehicular mobility: how vehicles are distributed over the simulated road map. To do so, we have created several heat maps that collect information about the total number of vehicles that reported a certain location during the simulation. The parts of the map colored in red experienced the maximum density values, while parts colored in white experienced the minimum density values. All the heat maps that belong to the same scenario share the same scale.

Figure 5 shows the vehicle distribution in the Manhattan scenario. Due to its lack of randomness, duaRouter is unable to select different routes for vehicles when there are several streets with the same travel-time. This prevents the simulator from properly distributing vehicles, and so all of them are routed through the same street. When using the duaIterate.py script, a better distribution of the vehicles is achieved since many simulations are sequentially executed to optimize vehicle routes. Since the VACaMobil's random routes set is obtained from duaIterate.py, it achieves a similar nodes distribution.

Figure 6 shows the vehicle distribution in the suburban scenario. In this case, duaRouter concentrates all the traffic in the inner roads. Therefore, the typical ring roads around the map are underused, which is not a common driver behavior. When using the duaIterate.py or VACaMobil, vehicles are distributed among the two bigger streets, i.e., the inner and the outer avenues.

Figure 7 shows the vehicle distribution in the urban grid scenario. Since this map includes big roads, few congestion points are present with duaRouter. However, a smarter driver behavior can also be seen when using duaIterate.py or VACaMobil. These few congestion points are also avoided with these tools because drivers can take advantage of alternative paths in this grid to improve their travel times.

Finally, Figure 8 shows the vehicle distribution in the urban downtown scenario. In this case, duaRouter is also unable to spread the vehicles properly. Since some roads are faster than others, all the vehicles are routed through them, even when these streets are congested. Therefore, undesired traffic congestion is created in the fastest inner roads. However, this is an unrealistic scenario because drivers tend to avoid traffic jams whenever possible. When using either duaIterate.py or VACaMobil, vehicles are routed through alternative streets, avoiding traffic jams. This strategy has a higher degree of similitude compared to real road traffic, since drivers prefer faster roads but often change their route to avoid traffic jams.

B. *Vehicle Density Study*

Table II shows the differences in terms of number of

TABLE II
VEHICLE STATISTIC SUMMARY

	Synthetic Manhattan		Suburban scenario		Urban grid scenario		Urban downtown scenario	
	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
duaRouter	313.767	58.8271	319.165	79.4342	214.711	39.0532	880.546	465.716
duaIterate	304.487	55.5174	219.610	34.2232	183.009	36.6651	393.717	96.414
VACaMobil	319.349	6.14267	223.718	8.32201	174.615	6.79294	369.691	7.84640

TABLE I
VACAMOBIL CONFIGURATION

	Vehicle number	carVariability	Std. dev.
Synthetic Manhattan	320	20	6.33
Suburban scenario	225	25	8.33
Urban grid scenario	175	20	6.33
Urban downtown scenario	370	25	8.33

TABLE III
STEPS REQUIRED

Method	#Files	#Commands	#Rep Steps
duaRouter	4•N•K	2•N•K	2•N
duaIterate	4•N•K	3•N•K	3•N
VACaMobil	4	4	0

vehicles for the four different scenarios for each different traffic generation tool. Since neither duaRouter nor duaIterate.py allows defining the number of cars in the simulation, we configured VACaMobil according to values in Table I to mimic the average values obtained previously when using duaRouter and duaIterate. In the simplest scenario (Manhattan), the three methods can achieve a stable value for the mean vehicle density with a low standard deviation. However, neither duaRouter nor duaIterate allow to a priori configure the value of this parameter. On the contrary, VACaMobil is not only able to quickly populate the network with the desired number of vehicles, but it also allows defining a maximum and a minimum number of vehicles, which will bound the standard deviation value. In complex maps like the urban scenario, VACaMobil is the only tool able to maintain the number of vehicles within the predefined bounds. To better understand the aforementioned values, Figures 9 to 12 show the evolution on the number of vehicles in the different scenarios against time for each tool. Since duaRouter and duaIterate.py are able to add only one vehicle per second, the user cannot predict when vehicles will arrive to their destination and disappear from the network. Therefore, for the Manhattan scenario, the number of vehicles when the simulation reaches the steady-state is not known a priori, turning protocol analysis based on the number of vehicles into a mere act of faith. Moreover, in maps where traffic jams are common, as in the urban downtown scenario, it takes more time for vehicles to reach their final destination and leave the network, which causes a constantly increasing number of vehicles in the network when not using VACaMobil. Comparing the configuration in Table I and the results in Table II, we can conclude that both the target number of vehicles and the expected standard deviation are achieved only with VACaMobil.

V. IMPACT ON NETWORK PROTOCOLS

In this section we evaluate how nodes' mobility can impact network protocols performance. To demonstrate its effects, we have simulated the same network topology using duaRouter, duaIterate.py, and VACaMobil to manage the mobility.

A. *Mobility Specification*

To perform this test, and for sake of simplicity, we chose the synthetic Manhattan scenario presented before. In contrast to previous evaluations, we have manually modified the configuration files generated by duaRouter and duaIterate.py in order to simultaneously introduce 100 vehicles with random routes at the beginning of the simulation. VACaMobil was configured to maintain 100 nodes in the network during the whole simulation. Figure 13 shows the number of nodes inside the network. As can be appreciated, when using both, duaRouter and duaIterate.py, the number of cars in the network decreases, whereas when using VACaMobil it is kept at its desired value along the entire simulation.

B. *Network Traffic*

This simulation test aims at evaluating the connectivity between two Road Side Units (RSUs). In order to maximize the impact of mobility, we placed both RSUs in the most-traversed road of the scenario. The distance between them was 900 m and the communication range was approximately 200 m. Each RSU sends a 512 Bytes UDP packet to the other RSU every second. The vehicles were configured to use the DYMO routing protocol [18].

C. *Results*

To assess the impact of mobility we compared the connectivity along the simulation when using different mobility alternatives. To do so, we have counted the number of received packets for every 30-second slot. Figure 14 shows the results we obtained after aggregating the values from 10 different simulations. Notice that, when using duaRouter or duaIterate.py, the connectivity of the network experiences an initial period where it is higher than when using VACaMobil. However, as long as the number of vehicles decreases according to Figure 13, the network gradually loses its connectivity. On the contrary, when using VACaMobil, the connectivity of the network fluctuates throughout the

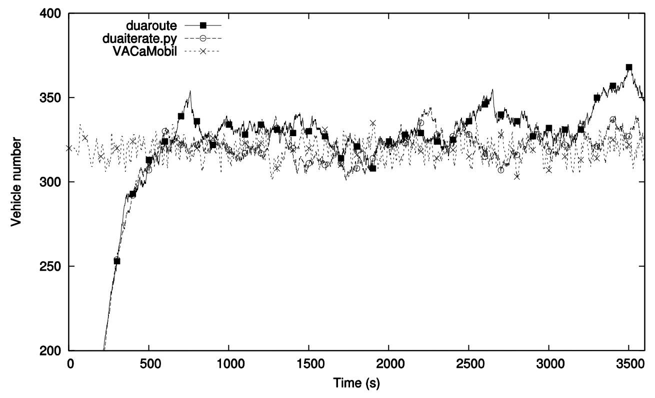


Fig. 9. Evolution of the number of vehicles against time for the Manhattan scenario

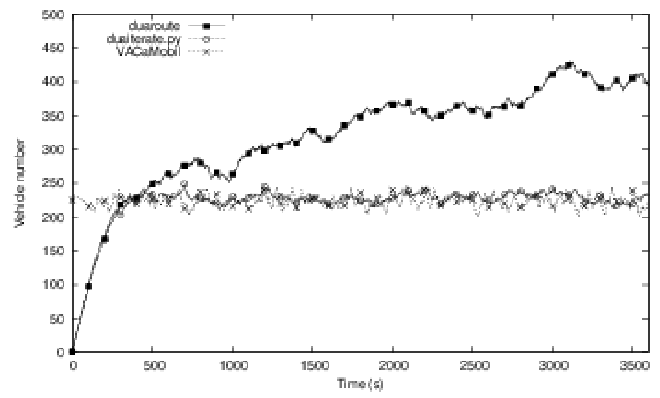


Fig. 10. Evolution of the number of vehicles against time for the suburban scenario

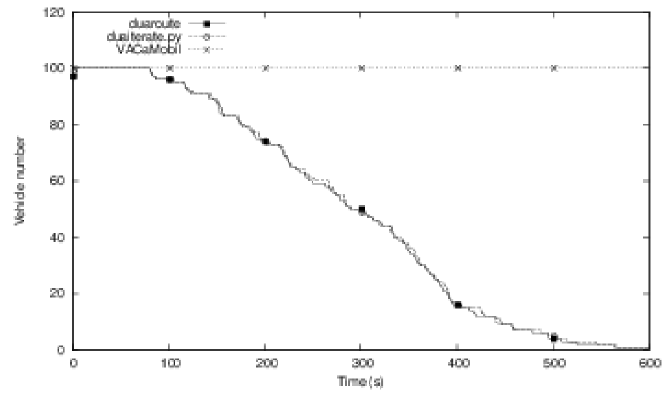


Fig. 13. Number of nodes in the network after introducing 100 cars at the beginning of the simulation

simulation with peaks and valleys due to the random nature of cars mobility. This experiment highlights one of the effects of mobility on network protocols. When evaluating more

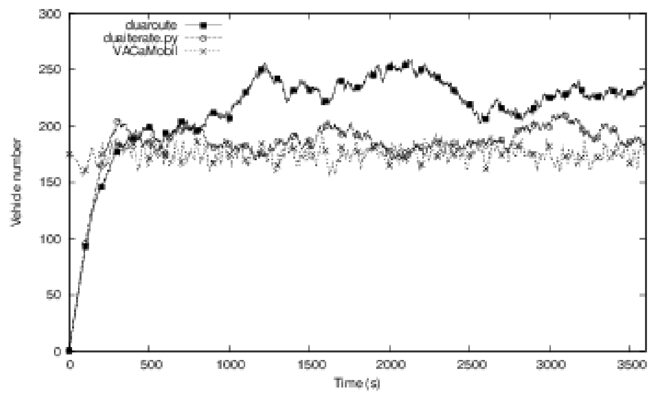


Fig. 11. Evolution of the number of vehicles against time for the urban grid scenario

complicated scenarios, such as diffusion protocols, delay-tolerant networks, or cooperative sensors, new problems related with the lifetime of the nodes arise [2]

VI. Workflow Comparison

In the previous section we have shown how VACaMobil can guarantee that the number of nodes in a simulation is within certain bounds. In this section we describe the typical steps required to obtain conclusive simulation results when using VACaMobil, and compare them against duaRouter and duaIterate.py. This section does not aim to be a tutorial or a guide, but instead to illustrate how VACaMobil makes it easier to avoid biased results due to mobility effects. To do so, we reproduce the process to obtain N non-correlated simulation runs.

A. Network Map

The first step to simulate our network is to obtain or generate our road map. It can be downloaded from OpenStreetMap [19] and then converted to SUMO format using the tools included with SUMO (netconvert). This process was the one we used to generate the urban scenarios presented in previous sections. The road network can also be synthetically generated, as we did for the previous Manhattan scenario. This step is always required in order to simulate a vehicular network using SUMO to manage nodes' mobility. We assume that the same map is used for our N non-correlated repetitions; however, it is always a good practice to vary the scenario using different city maps.

B. Traffic Demand

Once we have our road scenario, we need to generate the traffic demand. The way to model it varies between duaRouter, duaIterate.py, and VACaMobil.

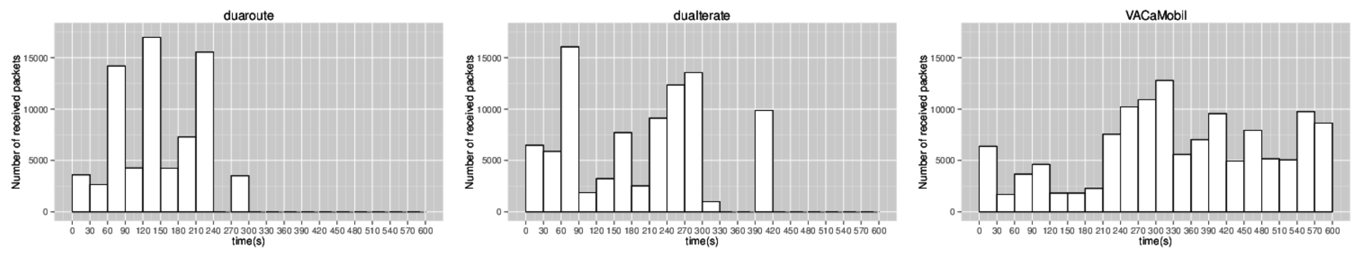


Fig. 14. Number of received packets

1) Duarouter: To generate random traffic demand for duaRouter we need to invoke the randomTrips.sh script presented in section II, and call duaRouter to compute the shortest routes. This process will produce a file containing a set of vehicles that will be simulated by SUMO. If our objective is to repeat N non-correlated simulations we need to repeat these steps N times, once for every simulation, and generate N configuration files that will be made available to SUMO through the OMNeT++ configuration file. In the not so atypical case that we want to vary the number of nodes in our simulation and simulate K different scenarios, we need to repeat these steps K ∗ N times in order to generate N different SUMO configuration files for the K possible scenarios.

2) DuaIterate.py: As referred above duaIterate.py refines the traffic demand obtained with duaRouter to minimize the time cost of the routes. As a consequence, it adds a new step to the duaRouter procedure. This new step must also be repeated for every simulation. As well as duaRouter, duaIterate.py generates a file containing a set of vehicles that will be simulated by SUMO.

3) VACaMobil: In VACaMobil, the traffic demand consists of a set of routes that can be generated immediately after obtaining the network map. VACaMobil includes a script, called randomRoutes.sh, that generates a large-enough set of random routes to be used.

C. *OMNeT++ Configuration*

1) Duarouter and duaIterate.py: In order to configure OMNeT++ to use the mobility generated by duaRouter or duaIter- ate.py, we need to specify the path for each one of the different SUMO configuration files. As seen before, this path should change for every simulation.

2) VACaMobil: In order to simulate the desired scenario with VACaMobil we only need to specify the number of repetitions and an array containing the different number of nodes that should be introduced into the network. OMNeT++ and VACaMobil will manage the whole set of simulations, storing their results in different files, ready to be parsed.

D. *Total Number of Steps*

In this subsection we detail the number of commands required to repeat N times the simulation of K scenarios which differ in the number of nodes introduced into the network. Table III shows the number of files we need to create, as well as the number of steps required to configure our simulations. The "#Rep Steps" column contains the number of steps we need to repeat in case we decide to add a new scenario K + 1. It is clear that VACaMobil does not only provide a more "realistic" and steadier mobility generator for our simulations, but it also simplifies the complexity of the mobility specification, which makes it less prone to errors.

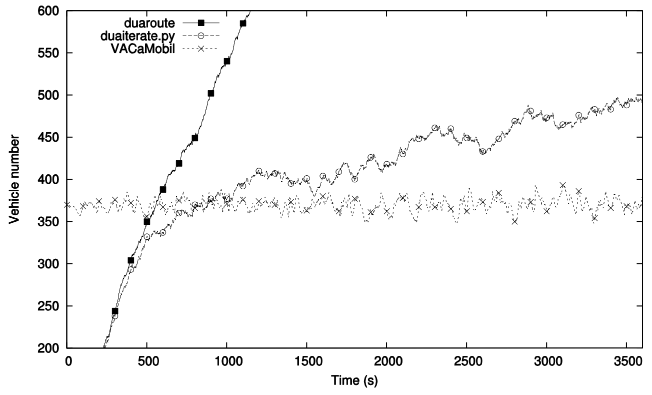


Fig. 12. Evolution of the number of vehicles against time for the urban downtown scenario.

VII. Conclusions and Future Work

In this paper we presented VACaMobil2, a new mobility manager for the OMNeT++ simulator which simplifies the definition of mobility for VANET simulations. The main novelty of VACaMobil is that: to the best of our knowledge, it is the first tool able to generate SUMO [4] driven nodes in a vehicular network while ensuring the stability of certain user-defined parameters, such as the average, maximum, and minimum number of vehicles. In particular, we added critical features to previously existing tools, such as ensuring a constant number of vehicles during the entire simulation time, disseminating vehicles throughout the whole route-map, and offering the possibility of defining different vehicle types with different probabilities. In contrast to other existing tools, VACaMobil is able to keep the mean number of vehicles and the standard deviation value within user-defined bounds. To the best of our knowledge, this is currently the only tool that allows studying a vehicular network in a steady-state situation without losing the realistic vehicle behavior provided by SUMO. As future work we plan to improve VACaMobil by offering downtown definition and automatic placement for Road Side Units (RSU).

ACKNOWLEDGEMENTS

This work was partially supported by the Ministerio de Economía y Competitividad, Spain, under Grants TIN2011-27543- C03-01 and BES-2012-052673, by the Ministerio de Educación, Spain, under the FPU program, AP2010-4397, AP2009-2415, and by the Universitat Politècnica de València under project ABATIS (PAID-05-12).

REFERENCES

[1] S. Joerer, C. Sommer, and F. Dressler, "Toward Reproducibility and Comparability of IVC Simulation Studies: A Literature Survey," IEEE Communications Magazine, vol. 50, no. 10, pp. 82–88, October 2012.

[2] C. Sommer and F. Dressler, "Progressing toward realistic mobility models in vanet simulations," Communications Magazine, IEEE, vol. 46, no. 11, pp. 132 –137, Nov. 2008.

[3] M. Fogue, P. Garrido, F. J. Martinez, J.-C. Cano, C. T. Calafate, and P. Manzoni, "An adaptive system based on roadmap profiling to enhance warning message dissemination in vanets," Networking, IEEE/ACM Transactions on, vol. PP, no. 99, p. 1, 2012.

[4] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo - simulation of urban mobility: An overview," in SIMUL 2011, The Third International Conference on Advances in System Simulation, Barcelona, Spain, October 2011, pp. 63–68.

[5] J. Yoon, M. Liu, and B. Noble, "Random waypoint considered harmful," in INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, vol. 2, March-3 April 2003, pp. 1312 – 1321 vol.2.

[6] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking, ser. WDTN '05. New York, NY, USA: ACM, 2005, pp. 252–259.

[7] F. Martinez, J.-C. Cano, C. Calafate, and P. Manzoni, "Citymob: A mobility model pattern generator for vanets," in Communications Workshops, 2008.ICC Workshops '08. IEEE International Conference on, May 2008, pp. 370 –374.

[8] X. Li, W. Shu, M. Li, H. Huang, and M.-Y. Wu, "DTN Routing in Vehicular Sensor Networks," in Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE. IEEE, Nov. 2008, pp. 1–5.

[9] M. Gramaglia, M. Calderon, and C. Bernardos, "Trebol: Tree-based routing and address autoconfiguration for vehicle-to-internet communications," in Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd, May 2011, pp. 1 –5.

[10] L.-J. Chen, Y.-Y. Chen, K. chan Lan, and C.-M. Chou, "Localized data dissemination in vehicular sensing networks," in Vehicular Networking Conference (VNC), 2009 IEEE, Oct. 2009, pp. 1 –6.

[11] C.-C. Lo, J.-W. Lee, C.-H. Lin, M.-F. Horng, and Y.-H. Kuo, "A cooperative destination discovery scheme to support adaptive routing in vanets," in Vehicular Networking Conference (VNC), 2010 IEEE, Dec. 2010, pp. 202 –208.

[12] M. Fogue, P. Garrido, F. J. Martinez, J.-C. Cano, C. T. Calafate, and P. Manzoni, "Using roadmap profiling to enhance the warning message dissemination in vehicular environments," in 36th IEEE Conference on Local Computer Networks (LCN 2011), Bonn, Germany, October 2011.

[13] C. Sommer, O. Tonguz, and F. Dressler, "Traffic information systems: efficient message dissemination via adaptive beaconing," Communications Magazine, IEEE, vol. 49, no. 5, pp. 173 –179, May 2011.

[14] C. Sommer, R. German, and F. Dressler, "Bidirectionally coupled network and road traffic simulation for improved ivc analysis," Mobile Computing, IEEE Transactions on, vol. 10, no. 1, pp. 3 –15, Jan. 2011.

[15] C. Sommer, O. Tonguz, and F. Dressler, "Adaptive beaconing for delay-sensitive and congestion-aware traffic information systems," in Vehicular Networking Conference (VNC), 2010 IEEE, Dec. 2010, pp. 1 –8.

[16] "Inet framework website," http://inet.omnetpp.org/, last visit Jul 2013.

[17] "Mixim framework website," http://mixim.sourceforge.net/, last visit Jul 2013.

[18] C. Perkins, S. Ratliff, and J. Dowden, "Dynamic MANET on-demand (DYMO) routing," draft-ietf-manet-dymo-26 (work in progress), 2013.

[19] "OpenStreetMap website," http://www.openstreetmap.org, October 2012.

[20] D. Bain and J. R. Paseur, "Inter vehicle communication system," U.S. Patent 8 307 037 B2, Nov. 6, 2012

[21] M. Nagura, "Inter-vehicle communication apparatus and method capable of detecting packet collision," U.S. Patent 8 451 732 B2, May 28, 2013

[22] E. B. Lofton, "System, method and apparatus for inter-vehicle communication," U.S. Patent 2013/0157576 A1, Jun. 20, 2013

[23] K. Oguchi, et al., "Simulator for vehicle radio propagation including shadowing effects" U.S. Patent 2007/0271079 A1, Nov. 22, 2007

[24] C. L. Barret, et al. "Population mobility generator and simulator," U. S. Patent 2004/008392 A1, May 6, 2004