Inter-module Communications for Rear-end Collision Avoidance Messaging in an IEEE 802.11p Interface

Juan-Bautista Tomas-Gabarron, Esteban Egea-Lopez and Joan Garcia-Haro
Department of Information and Communication Technologies, Universidad Politecnica de Cartagena
Plaza del Hospital, 1, 30202 Cartagena, Spain

Keywords: NCTUns, Event-driven, CCA, VANET.

Abstract: Computer simulation is the preferred approach when investigating Vehicular Ad-hoc Networks (VANET). During the last years different platforms have emerged, regarding the evaluation of next-generation autonomous vehicular mobility and car-to-car wireless connectivity. NCTUns (now Estinet) is a relevant networking platform that provides support for IEEE 802.11p-based connectivity between vehicles. This paper describes the implementation of an inter-module communications' scheme in NCTUns designed to allow reciprocal message transmission and processing between a mobility and a messaging agent in vehicles supporting *Cooperative chain Collision Avoidance* (CcCA) for improving safety even under rear-end collision risky circumstances. Contributions hold in the bidirectional inter-module channel that features the management of mobility in these environments according to the communication's protocol that implements the CcCA application, and viceversa. As an additional characteristic of the implementation, a Nakagami-*m* channel model is implemented to recreate intervehicular communications more realistically.

1 INTRODUCTION

Network simulators are nowadays essential tools for the design, development and evaluation of communication protocols. The importance of simulation is due primarily to the fact that it allows developers to carry out experiments using a model-based approach that saves time and money. It also gives them much more flexibility than real experiments, since it can be used for example to evaluate network performance under different configurations with remarkable easiness. Simulation results are also easier to handle due to the great simplicity when directly collecting and processing digital data.

Network simulators, however, have some limitations. A functional network simulator needs to simulate network devices (eg. routers and hosts) and applications that generate network traffic. It is also necessary to provide monitoring programs and set up the configuration parameters for each test. These facts mean that network simulators work according to a model of reality whose functionality might not correspond entirely with the real processes that it recreates. The associated drawbacks are summarized next:

- Network simulator results are not, for obvious reasons, as realistic as those obtained in real experiments under the same circumstances, since to limit the complexity and production costs, most network simulators implement simplified versions of reality (network protocols), that usually lead to results that do not necessarily match physical phenomena in most situations.
- Most network simulators do not have the property of extensibility because they are exempt from supporting the UNIX POSIX API[1]. As a result, existing network applications as well as those yet to be developed can not be executed properly to generate traffic for a given simulated network under different OS, nor can evaluate its performance under different network settings. On the contrary, they make use of the API provided by the own simulator and are compiled with it to build a unique, modular and quite complex program that only works in the same computer where compilation takes place.

To avoid these problems, the NCTUns/Estinet team (Shie-Yuan and Chih-Che, 2008) devised a sim-

[1]Family of standards which implements a common framework of system calls for extensive compatibility between different operating systems (see reference (Board, 1999)).

Tomas-Gabarron J., Egea-Lopez E. and Garcia-Haro J..
Inter-module Communications for Rear-end Collision Avoidance Messaging in an IEEE 802.11p Interface.
DOI: 10.5220/0004427500820089
In *Proceedings of the 3rd International Conference on Simulation and Modeling Methodologies, Technologies and Applications* (SIMULTECH-2013), pages 82-89
ISBN: 978-989-8565-69-3

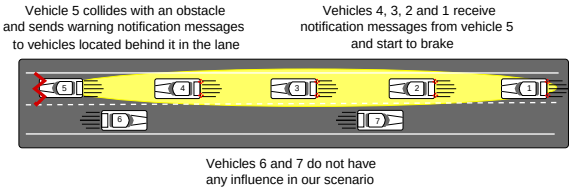

Figure 1: Simulation scenario with 802.11p communication capabilities.

ulation methodology called kernel reentering, which was already used in the first version of the simulator, called Harvard NS. Later, some significant improvements were made to the original methodology to finally implement the simulator and emulator capabilities that NCTUns currently supports. By using the technique of kernel reentering, NCTUns provides a number of advantages over traditional simulators that will be thoroughly explained in next subsections of this Introduction.

For a more practical view of the characteristics of NCTUns and focusing on the particular implementation of intervehicular connectivity in VANETs, we describe in the second section of this paper the core implementation of our inter-module communications' scheme, i.e. the specific functionality of our CcCA application. Since NCTUns addresses mobility and communications as independent processes, we had to implement a specific two-way communications' scheme to allow both of them to work together in general collision avoidance applications. This way it is possible to establish safety policies to reduce the probability of accident in rear-end collision avoidance scenarios like that of Fig. 1 by allowing information exchange for mobility update, and viceversa. Furthermore, making use of the WSMP (WAVE Short Message Protocol) for a more efficient message transmission methodology, as well as the Nakagami model for radio signal propagation, we introduce in depth the main characteristics of our scheme. Actually, this extension has proved flexible and has been succesfully used to evaluate VANET performance in various scenarios (see (Garcia-Costa et al., 2012), for example). Implementation codes and installation guides are available on the Internet at www.juanbatomas.com.

We will finish this paper by making some allusions to the utility of this scheme and possible future worklines of this implementation.

1.1 Kernel Reentering Methodology

Using a tunnel network interface is the basic approach that summarizes the functionality offered by the reentering simulation methodology of NCTUns. A tunnel

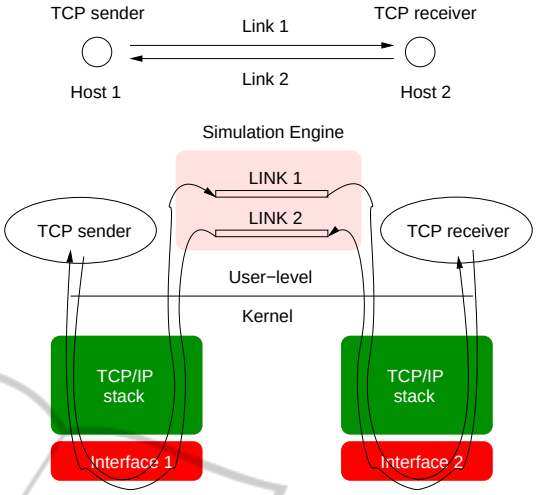

Figure 2: Kernel reentering methodology of NCTUns.

network interface available on most UNIX machines, is a pseudo network interface that does not have any physical network interface assigned to it. However, the functionality of this interface is not different from the one offered by the Ethernet network interface, i.e. an application program to send and receive packets on this pseudo network interface would offer the same functionality as if it did through a physical interface.

Each tunnel interface has a special device file assigned to it, which any application running on NCTUns has access to, basically for writing information related to packets that are sent to other network entities for the purpose of simulation. From the perspective of the kernel module, the packet seems to come from an external communications network, and will come trough the TCP/IP protocol stack just like normal Ethernet packets would do (Shie-Yuan and Chih-Che, 2008). Moreover, if an application program reads a packet from the special device file of the tunnel interface, the first packet in the output queue of the kernel tunnel interface will be copied to the application program. This situation is interpreted by the kernel as if a packet had been sent to a real link without knowing if this transmission had taken place with real Ethernet packets (see Fig. 2).

1.2 Main Characteristics of NCTUns

In the following list we present the most important features that characterize NCTUns as a network simulation platform, and the reasons we use to justify its choice as our simulation environment for implementing the rear-end collision avoidance scheme (CcCA).

- Thanks to the Kernel reentering methodology of NCTUns, this platform can use the UNIX protocol stack in its own network interface for the

generation of very realistic results. By the time we carried out the implementation of our bidirectional module, the WSMP protocol was not supported by the Linux Kernel yet, so we had to use the application-layer version of this protocol that came along with the general API of the platform. Now that preliminary versions of the WAVE stack are already supported in different linux-based platforms, it is possible to design applications relying on communication stacks implementing functional versions of the WSMP protocol (see for example (Hernandez-Jayo et al., 2012)).

- In NCTUns real UNIX applications can be used as traffic generators (teletraffic) for network performance evaluation. In our case, we will make the implementation of a specific application based on the timely transmission of chain collision alerts (CcCA).
- It is possible to use configuration tools and real network monitoring services such as *ifconfig*, *netstat*, *tcpdump*, *traceroute*, etc. available on all UNIX operating systems. In our case we will use a diagnostic tool designed exclusively by us, since our main concern is not to analyze the teletraffic utilizing the network, but until to which extent vehicle's safety can be improved.
- NCTUns supports the simulation of a wide variety of different types of network protocols and network devices. In our case we simulate a Vehicular Ad-hoc Network for emergency braking in CcCA (with the protocol stack architecture WAVE 1609/IEEE 802.11p).
- Due to the discrete event simulation approach of NCTUns, simulation runs execute very quickly (in the general case). Also, simulation results are repeatable, thus bringing us the possibility of evaluating network performance under particular scenarios that might need further analysis.

Despite the plug-and-play methodology of NCTUns for the evaluation of linux-based user services, this platform also suffers, in our opinion, from some drawbacks that are mentioned next:

- Although NCTUns is open source, it contains a proprietary GUI (Guided User Interface) that the end user can not modify. This somewhat reduces the flexibility of the programmer to set up complex scenarios and develop new protocols.
- NCTUns does not provide general simulation scripts for the automatic setup of network configuration files. Normally, it is necessary to make a great effort to program a series of Bash/Python scripts that were used to set up network parameters in the VANET under consideration.
- The WSMP protocol we use for the transmission of information in the network layer of the WAVE architecture is not implemented in the Linux kernel. This implies that the advantage of directly using the TCP/IP stack from the network interface (basically, the reentering methodology) is lost due to the fact that WSMP is simulated as if it was a process of the application layer.
- The generation of a large number of sockets (for each virtual interface, each one for each vehicle) to perform simulations and the deficient memory release scheme of NCTUns when a simulation ends, specially after making a remarkable deal of system simulations, occasionally implies that the socket creation process takes longer time than simulating the network under consideration.
- The NCTUns network simulator suffers from a drawback that could negatively reduce the number of potential users: the lack of an API (Application Project Interface) documenting all the code generated for the networking platform, greatly increases the programmer's effort to understand the functionality offered by the simulator. Actually, authors had to investigate the details of the code without having a sufficiently clear description of the architectural implementation of the NCTUns modules.

Fortunately, all these issues have already been solved by offering a commercial version of NCTUns, renamed Estinet, which provides the same functionality of NCTUns, but includes a very useful documentation of the API that greatly reduces the effort of the network programmer. Whatsoever, it has become commercial, thus requiring to pay a license that might decrease the number of potential new users. NCTUns, on the other hand, can still be found (in its original open source version) for download in http://nsl.csie.nctu.edu.tw/nctuns.html.

2 CcCA COMMUNICATIONS MODULE FOR NCTUns

Once we have presented the most important features of NCTUns, we describe the main module implemented to support communications in applications for the transmission of chain collision alerts.

In the present case, we differentiate between two complementary modules:

- **Mobility Module:** although NCTUns already includes the implementation of different types of

mobility models, in our case we developed our own motion dynamics for vehicles circulating in convoy under risk of chain collision. We preferred to do this because already available mobility managers in NCTUns are designed to govern vehicular motion on preestablished roads which must be configured beforehand in the setup files. In our case, we are not interested in completing a certain path marked by a road, but analyzing how vehicles can react when following dynamical routes according to the reaction to Cooperative chain Collision Avoidance (CcCA) notifications received in destination.

- **Communications Module:** Although not implemented in the Linux kernel (like the TCP/IP stack), NCTUns allows us to use WSMP (WAVE Short Message Protocol) as an application service (instead of a network layer implementation) for the transmission of safety information between vehicles. Because of the timing requirements set by this safety-related application, using this protocol is advisable in this case. In this context it is necessary to carry out some internal modifications of the original WSMP module[2] in order to enable a bidirectional link between the communications module and the mobility module, as we will see next.

In our study, the implementation of two-way communication between the two modules is required, mainly for the following reasons:

- In a hypothetical situation where a vehicle collides with an obstacle along its path it is necessary to notify the communications module that it has the responsibility for sending information to the following vehicles (in the form of WSMP packets) in order to warn them about this event.
- In the same situation, if a vehicle receives WSMP packets, a reaction to this information is foreseen. This requires informing both the communications module and the mobility module, so that the latter can respond to the received information by, for example, reducing the vehicle speed as appropriate.

All vehicles will contain these two modules, coded by using the NCTUns APIs and the UNIX sockets libraries that hold the generic functionality of the AF_UNIX sockets, available for implementing communications between different processes on the same machine.

[2]The one that came already installed in the NCTUns distribution.

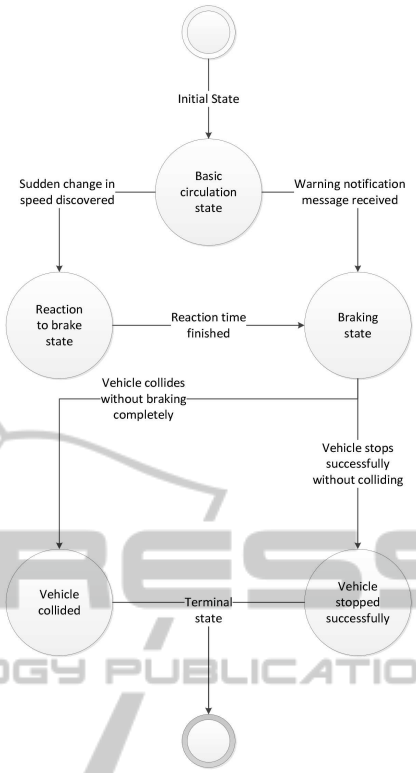

Figure 3: System's state diagram.

2.1 Mobility Module

Due to the enormous complexity of developing mobility models for generic vehicular traffic, we focus on a particular implementation of a mobility model that is adapted to the specific situation of emergency braking for CcCA. Fig. 3 represents the different transition states that each vehicle can contemplate during motion, whereas Fig. 4 shows a flowchart with the behavior of each vehicle according to the specific traffic state in which it might be found, and the particular update of state variables that takes place.

All vehicles will behave according to five possible states:

1. **Basic Circulation:** the vehicle moves at a constant speed along its path on the road.
2. **Reaction to Brake:** in a normal braking situation (human assistance), the driver will need a small time interval (between 0.5 and 1 s) to start to slow down (from the moment the danger is noticed until the brake pedal is pushed).
3. **Braking:** once the brake pedal is pressed (in human assisted driving) or due to automatic braking in an autonomous driving scheme, the vehicle begins to decelerate so as to reach null speed in the driveway.

4. **Vehicle Collided:** the vehicle does not stop in time to avoid a collision with the vehicle ahead.
5. **Vehicle Stopped Successfully:** the vehicle has successfully managed to stop and therefore it avoided the collision.

As can be seen in the flowchart presented in Fig. 4, first we configure a set of parameters that represents the essential characteristics of mobility of vehicles in this particular scenario: initial driving direction, initial position and initial speed. In our implementation this is achieved by the API implemented in NCTUns for developing user-specific mobility agents, by means of a socket communication scheme that allows sending mobility management commands to the simulation engine (which handles mobility).

Then we can observe the logical loop that determines the mobility patterns of the designed model (see the corresponding state's diagram in Fig. 3). First we evaluate whether the speeds of all vehicles are equal to zero. In this case, the simulation ends to avoid consuming more time. Afterwards it calculates for each vehicle the position of the nearest neighbor in the direction of transit (either ahead or rear). If the distance is less than a certain threshold, we consider this to be a collision, so a variable that stores the state of the vehicle (specifically *colEvent*) is set to 1 (0 otherwise). In the following step we analyze the value of this variable. If it is equal to 1, it means that the vehicle is in a state of collision, thus we tell the process governing the mobility of the collided vehicle to stop to save simulation resources. We simply set the vehicle speed to zero and advance the simulation time.

If the vehicle is found in a normal traffic state, that is, neither stopped (by braking successfully without colliding), nor collided, first we evaluate whether there is any message in the mobility queue. Because in our case the only sent message corresponds to alert emergency braking, there is no problem in terms of packet prioritization. Particularly, all sudden brake warning messages in this implementation contain the following information:

- Position of the collided vehicle that issued the alert: to be used to check whether it is possible to stop in time.
- Timestamp at which the message was sent: to check end to end delay of messages.
- Damaged vehicle identifier: to verify the identity of the vehicle that was involved in the accident.

Should we find such messages in the mobility queue, the receiving vehicle automatically calculates the braking distance that it needs to reach a full standstill stop, according to the equation presented below, obtained from reference (de Fomento., 2011):

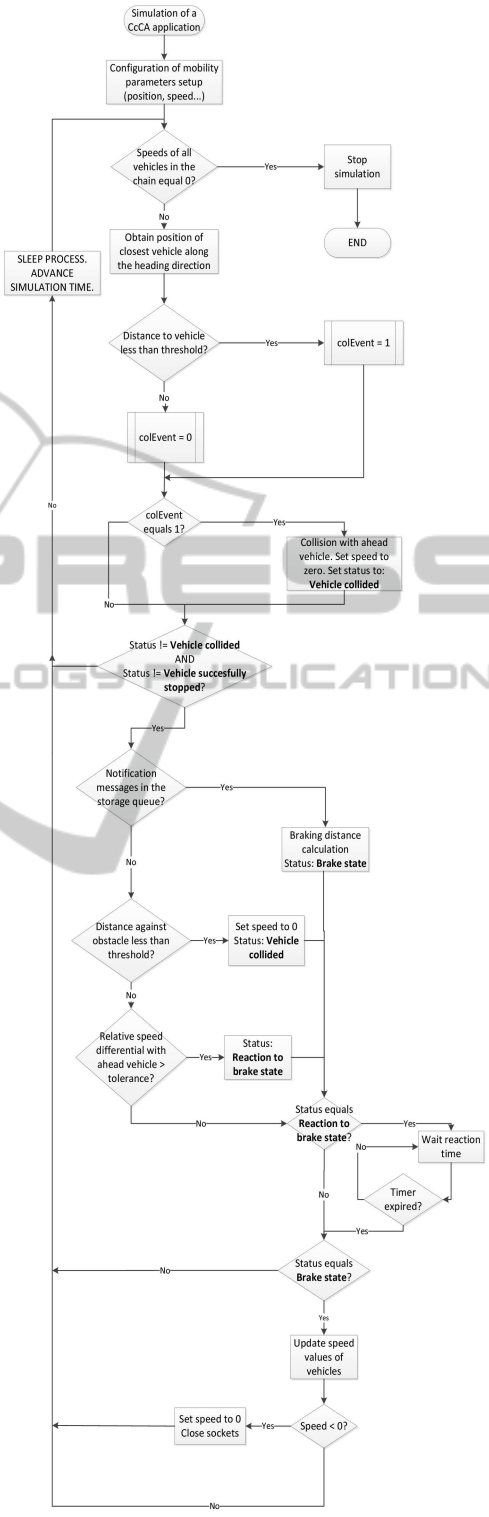

Figure 4: Communications' module flowchart.

$$D_p = \frac{v_I \cdot t_p}{3.6} + \frac{v_I^2}{254\,(i + f_i)} \quad (1)$$

Where v_I denotes the initial speed in km/h, i the

road slope coefficient, t_p the sum of the perception and the reaction time, and f_i representing the longitudinal friction coefficient. The resulting distance (in meters) will be the sum of the distance traveled by the vehicle during the reaction time (first term) and the distance traveled while braking (second term). In our case, we skip the first term in the calculation of D_p, since we will only consider that the stopping distance is calculated from the time instant the vehicle starts to brake (the reaction time is also taken into account, but not for the calculation of D_p). As we can see, the relationship between speed and braking distance is squared, implying that two times the speed requires a braking distance multiplied by four. Another important factor to consider is the road friction, if f_i is very low due to poor conditions of wheels or because of bad weather conditions, the stopping distance may increase considerably, in fact on a icy pavement (e.g. $f_i = 0.05$, on a plane surface i=0) a car could have a stopping distance of a mile approximately at an average speed of 100 km/h.

In the case in which the vehicle finds no queued messages in the mobility queue, it checks whether there is any standing obstacle ahead on the road that could become a potential risk of accident. If a vehicle discovers an obstacle blocking the road along its path of movement, it will start to brake. If it collides, it will set the speed to zero, simulating the accident, and send an alert information to upcoming vehicles. Now it will transition to the collision status.

The last part of the algorithm corresponds to a situation where a particular vehicle begins to brake to avoid colliding with the vehicle directly ahead, either because of a too high speed differential, or due to the reception of accident warning notifications. In this case, according to our braking model, we assume that the vehicle decelerates with a speed that decreases linearly with time (constant braking deceleration). Due to the specific implementation of the NCTUns process manager (to switch between different machine processes and advance the simulation time), in our program we have to necessarily set an idle time (S) that the core engine of NCTUns uses to advance in the simulation time (during which the mobility/communication processes remain asleep). For this purpose, we have derived a mathematical equation that provides the speed of the vehicle at any time after starting the braking process, regardless of the time duration of the idle period. The expression is:

$$v(t_k) = v(t_{k-1}) - \frac{v^2(t_{k-1})}{2D_p} \cdot \left(t_{k-1} + \frac{S}{2} - t_{stop} \right) \rho \tag{2}$$

Where $v(t_k)$ denotes the speed to be set at the current instant k, according to the speed at the previous instant $(k-1)$ $v(t_{k-1})$, the pre-calculated braking distance D_p, the idle time interval of the application (for advancing the simulation time) S, the time at which the emergency braking begins t_{stop}, and the scaling factor of the simulation time ρ (t_k and t_{stop} are expressed in ticks, i.e. simulator time units, with a particular equivalence in temporal units: in our program we configure it to be 1 tick = 100 ns).

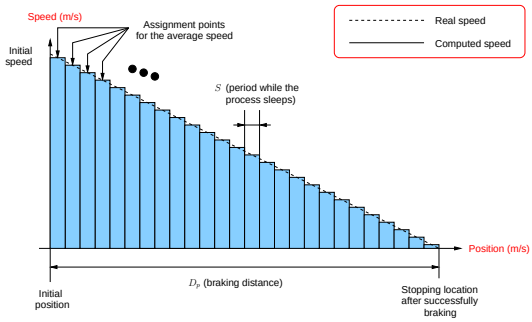

Figure 5: Idle-time period graph for the calculation of speed during braking.

The derivation of the above expression is explained below. Due to the linear decrease of speed, it will become a value at a future instant which is governed by the expression:

$$v(t_k) = v(t_{k-1}) - a \cdot t_{lapse} \tag{3}$$

Since the vehicle will need D_p meters (according to our mobility model) to fully stop, the following system of equations holds:

$$\begin{cases} v_I - a \cdot T = 0 \\ v_I \cdot T - a \cdot \frac{T^2}{2} = D_p \end{cases} \tag{4}$$

This system can be understood by realizing that when braking for a time interval of T s (time required to fulfill a complete stop), the car will reach zero speed. Moreover, the second equation expresses the distance D_p that is covered during the braking process. If we solve the system of equations we obtain a value for the acceleration that takes the expression:

$$a = \frac{v_I^2}{2D_p} \tag{5}$$

Substituting in the first term we have:

$$v(t_k) = v(t_{k-1}) - \frac{v_I^2}{2D_p} \cdot t_{lapse} \tag{6}$$

To give value to t_{lapse} let us carefully observe Fig. 5, which represents the evolution of the speed from the moment the vehicle begins to slow down until it completely stops.

As we can see, the simulation engine will dispatch the mobility manager every S time units (ticks), in order to advance the simulation time. Each time the mobility agent is awaken the simulation time will have advanced a time lapse of S ticks, thus needing to conveniently update the speed. However, due to the linear decrease of speed until the vehicle stops, the mobility manager must establish a constant moving speed during the interval when the process sleeps that allows to cover the same distance as if the speed was updated at infinitesimal time periods. Because of that, for obvious reasons we take the speed at the midpoint of each interval of S ticks. This makes t_{lapse} take the value given by the following expression:

$$t_{lapse} = \left(t_k + \frac{S}{2} - t_{stop} \right) \cdot S \tag{7}$$

Substituting Eq. (7) in the second equation of the System (4) we will get the value of the speed for successive instants.

All vehicles will start moving at an average speed that will be one of the representative configuration parameters in simulations. The intervehicular distance will also play an important role in the simulation, since the greater the distance the lower the probability of incurring in chain accidents. Moreover, the transmission power will be decisive on the behavior of vehicles reacting to the reception of warning messages, since it will influence on the end to end delay as well as other performance metrics like throughput.

2.2 WSMP Communications' Module

The communications module WSMP (WAVE Short Message Protocol) implements the network layer protocol of the architecture WAVE, using a simplified packet format that reduces overhead for a more efficient packet transmission and processing. This module is already implemented in the NCTUns 6.0 version we used for simulations, although in the context of our particular application for CcCA, it was necessary to carry out some modifications of the code to support bidirectional communications between this protocol and the mobility manager. Particularly, we have defined logical parallel threads to implement bidirectional communication between both modules.

As regards functionality, the WSMP module constantly awaits (when control of the simulation engine is given to the machine processes) the reception of:

- Messages sent by other vehicle reporting a particular issue about traffic safety (specifically, a collision ahead).
- Messages from the mobility management module within the same vehicle, reporting a particular feature of the nearby traffic, because it might have found an obstacle ahead. Therefore, it must inform the other nodes that come behind to conveniently react to this situation.

By implementing these changes in the WSMP module, the system can support bidirectional communication between both the communication and the mobility management modules, allowing us to build our own mobility model supporting active behavior in CcCA according to the reception of warning messages.

2.3 Nakagami-*m* Radio Propagation Model for the Communication's Channel

The propagation model that is used to recreate the characteristics of the physical layer in a simulation platform plays a decisive influence on the simulation speed and the reliability of results. Probabilistic models usually reproduce more faithfully the features of the associated signal propagation process in vehicular environments, even though they require more computational processing resources compared to deterministic models like the Two-Ray Ground or Free Space (Sizun and de Fornel, 2004). Particularly, the *Nakagami* model is frequently used for modeling such propagation effects in simulation platforms. Due to the wide variety of simulation approaches using Nakagami as channel model, and thanks to the trade-off between a great reliability when reproducing real-world propagation phenomena, and a reasonable computer processing cost of results, the use of this model in such scenarios is justified. Actually, this model can be used to model signal propagation in different scenarios, by controlling the value of its main design parameter m, as we will see later. In fact, for the implementation of our design we use the values of m extracted from the reference (Torrent-Moreno et al., 2004).

The Nakagami model is highly generic. The reception power follows a gamma distribution, as seen in Eq. (8):

$$P_r(d;m) \sim \gamma\left(m, \frac{P_{r\_\det}(d)}{m} \right) \tag{8}$$

The m parameter specifies the intensity of the attenuation effects on signal transmission process. Its main purpose is to configure the type of scenario of operation (low values of m, environments with higher losses, like urban environments; high values of m, environments with lower attenuation losses like motorways). P_{r_det} corresponds to the calculation of

the received power value for a deterministic propagation model like Free Space or Two-Ray Ground (see (Sizun and de Fornel, 2004)).

3 CONCLUSIONS AND FUTURE WORK

This work presents the implementation of an application-layer module for the NCTUns network platform that supports bidirectional communication between a mobility and a communications process working on the same protocol stack (of each vehicle). This characteristic allows us to implement a scheme where mobility can benefit from communications and viceversa for improved driving safety, in particular, for Cooperative chain Collision Avoidance (CcCA) support in two or more vehicles. In this regard, the mobility module manages the car's mobility and also collects information about the vehicle's dynamics (speed, position...). This collected information is given to the communication's module, which according to the specific processing scheme, will deliver safety-related notification packets alerting about a possible risk in the neighboring car traffic. On the other hand, the communication's module implements the functionality as related to the exchange of information packets between vehicles. When received at destination (based on a broadcast approach), packets are processed to extract interesting information that could be useful to avoid possible crashes by helping vehicles execute timely braking maneuvers.

We additionally solve some implementation aspects concerning the introduction of the bidirectional communications scheme between the mobility and the communications' modules by fixing the asynchronous distribution of simulation time for the different processes (mobility and communications) acting on top of the communications' stack of each participating vehicle. This paves the way for more complex applications to be implemented in this platform that may use the bidirectional mobility-communications' scheme to support road safety enhancements.

As future work, we plan to extend the functionality of the supported application in the bidirectional communications' scheme by also implementing evasive maneuvering for collision avoidance policies. Actually, a recently published algorithm for this type of collision avoidance scenarios has been developed by the Authors (Tomas-Gabarron et al., 2013), giving promising results, which now can be structured in an application in NCTUns in order to assess the performance of the communication process in this particular implementation.

ACKNOWLEDGEMENTS

This work was supported by project grant MINECO/FEDER TEC2010-21405-C02-02/TCM (CALM). It was also developed in the framework of '*Programa de Ayudas a Grupos de Excelencia de la Región de Murcia, Fundación Séneca*'. J. B. Tomas-Gabarron also thanks the Spanish MICINN for a FPU (REF AP2008-02244) pre-doctoral fellowship.

REFERENCES

Board, I.-S. S. (1999). Ieee standard for information technology-portable operating system interface (posix)-part 1: System application program interface (api)- amendment d: Additional real time extensions [c language].

de Fomento., M. (2011). Trazado. instrucción de carreteras. norma 3.1-ic.

Garcia-Costa, C., Egea-Lopez, E., Tomas-Gabarron, J., Garcia-Haro, J., and Haas, Z. (2012). A stochastic model for chain collisions of vehicles equipped with vehicular communications. *Intelligent Transportation Systems, IEEE Transactions on*, 13(2):503 –518.

Hernandez-Jayo, U., Sainz, N., Iglesias, I., Elejoste, M., Jimenez, D., and Zumalde, I. (2012). Ieee802.11p field operational test implementation and driver assistance services for enhanced driver safety. In *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, pages 305 –310.

Shie-Yuan, W. and Chih-Che, L. (2008). Nctuns 5.0: A network simulator for ieee 802.11(p) and 1609 wireless vehicular network researches. In *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, pages 1 –2.

Sizun, H. and de Fornel, P. (2004). *Radio Wave Propagation for Telecommunication Applications*. Foundations of Engineering Mechanics Series. Springer.

Tomas-Gabarron, J., Egea-Lopez, E., and Garcia-Haro, J. (2013). Optimization of vehicular trajectories under gaussian noise disturbances. *Future Internet, MDPI*, (5):1–20.

Torrent-Moreno, M., Jiang, D., and Hartenstein, H. (2004). Broadcast reception rates and effects of priority access in 802.11-based vehicular ad-hoc networks. In *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, VANET '04, pages 10–18, New York, NY, USA. ACM.