Bridging the Gap between Simulation and Experimentation in Vehicular Networks

Sofiane Khalfallah and Bertrand Ducourthial
Laboratoire Heudiasyc UMR CNRS 6599
Université de Technologie de Compiègne
Compiègne, France 60205
Email: firstname.name@hds.utc.fr

***Abstract*—The Intelligent Transportation Systems (ITS) currently attract a lot of attention. Many simulators and some testbeds have been proposed to validate new ideas and to guide new theoretical developments. Yet, to this date, simulator tools and testbeds evolve in disjoint research work. We developed Airplug-ns, an open source simulation environment, as a step towards bridging this gap. This environment allows to easily design new protocols, to test them on the road in order to obtain real measures as well as to study their behaviour. A simple simulator architecture is sketched; it is composed of a network simulator and a software suit switable for real tests. Experimentations are reported and some results are given.**

I. Introduction

A. Motivation

These last years, Intelligent Transport Systems gain attention of the research community. It is expected that ITS will reduce the road fatalities (around 40,000 deaths per year in USA or Europe), increase the productivity and the profitability of the infrastructures, avoid traffic jams and reduce the impact of road transports on the environment.

Research projects regarding ITS can be found in the USA (VII, CICAS, IVBSS...), in Europe (CVIS, SAFESPOT, COOPERS, PReVENT, GST, HIGHWAY, FLEETNET...), in Japan (SmartWay, VICS...), in India (ITSIndia), in Germany (NOW), in France (PREDIT) *etc*. Standardization of the vehicular communication is now ongoing in major international organizations (IEEE, IETF, ETSI, ISO, SAE, ASTM), industrial consortia such as the Open Mobile Alliance (OMA) and the Car-to-Car Communication Consortium (C2C-CC) and national ITS authorities. Hence, ITS is extensively studied by both theoretical and experimental researchers. The study of Inter-Vehicles Communication (IVC) networks in [3] exhibits characteristics that are dramatically different from many generic MANETs. It is our opinion that experimentation and prototyping is important for guiding theoretical studies, and validating technical solutions. But large experiments are very complex to organise and simulations remain very important to study the scalability of the protocols with many vehicles. However, there is generally a gap between road experiments and simulation. In this paper, we describe a simulation architecture that aims to reduce this gap. Starting from the implementation of a protocol, it allows both experimentation and simulation.

B. Related work

Various experiments [11], [15], [4], [12], [17], [6] concerned the communications in VANETs. They use mainly the standard IEEE 802.11. These experiments rely generally on a static routing and specific applications.

The simulation gives an easy way to prototype inter-vehicles applications and to test their limits in comparison with other solutions, with realistic traffic and large scale scenarios. Various tools for realistic simulations has been designed to evaluate the communications in VANETs.

The simulator TraNS [2] links two open-source simulators: a traffic simulator, SUMO [13], and a network simulator, ns-2 [14]. Thus, the network simulator can use realistic mobility models and influence the behavior of the traffic simulator based on the communication between vehicles. This approach employs a middleware to interconnect a traffic simulator and a network simulator. The middleware provides bidirectional links, realized by TCP connection, between SUMO and ns-2.

The simulator NCTUns [18] is a open source integrated simulation platform, for wireless vehicular communication. This tool allows a run-time control of vehicle movements through an intelligent driving behavior module. Furthermore, NCTUns integrates traffic and network simulations and provides a fast interaction between them.

There exists other simulators, *e.g.*, VISSIM/ns-2 [1], CARISMA/ns-2 [16] and CORSIM/QualNet [19], which achieve real-time interactions between a traffic and a network simulator. Note that solutions rely on commercial products, that prevent any adaptation by researchers.

All these simulations tools emphase interaction between traffic and network simulators. To the best of our knowledge, no vehicular simulator has been proposed to reduce the gap between the simulation and the road testbed, in term of development time, protocol implementation...

C. Contribution

In this work, we propose an architecture which permits to fast feedback loop between simulations and real tests of vehicular applications. To achieve this goal, we present first how to adapt a simple architecture suitable to VANETs experiments into ns-2. Afterward, we describe the extension brought to the simulator to read the real-time GPS logs. Finally, to illustrate the interest of such an architecture, we

explain how it can be used to compare the simulation and experiments of the performance of a given multi-hop protocol.

II. Architecture for road experiments

A. Process-based architecture

The Airplug software suit has been developed for experimenting in dynamic ad hoc networks [8], [10]. We summarize here its capabilities. The Airplug software suit is composed of a core program and a set of applications. It allows to easily experiment and prototype either on the road or on the lab. The core program `airplug` manages the inter-applications communications, either local-to-the-host or inter-vehicles. The applications are plugged on top of the core program. The applications reach the network through `airplug`. All these processes run in user-space for robustness and portability reasons. This does not prevent cross-layering protocols, as shown in Figure 1.

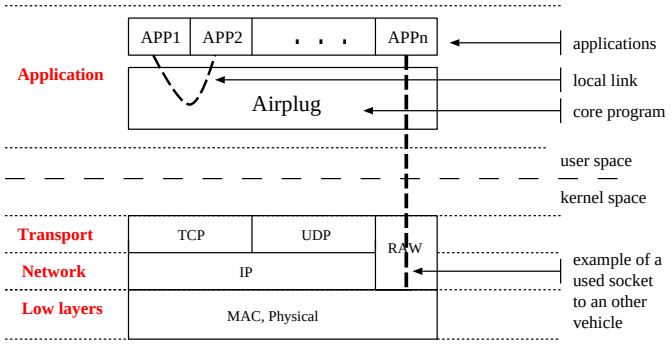


Fig. 1. Airplug architecture.

B. Airplug IPC

The Inter-Process-Communication relies on text-based ASCII messages and very few rules. This simple message-based framework uses an addressing well adapted to dynamic networks such as VANETs. An application `APP` can send a message locally (keyword `LCH` for localhost), to the nodes in the neighborhood (keyword `AIR`), to both (keyword `ALL`) or to a given host using a name or any kind of address. The destination is either a specific application (known by a three-characters mnemonic such as `APP`) or all the applications (keyword `ALL`). In this last case, only applications which have subscribed to the sender `APP` will receive the message. Such subscribing are managed by `airplug`; applications use the action keyword `BEG` to indicate to `airplug` that they want to receive the flow of messages from a given application, and the `END` keyword to end such a subscribing.

When a local application `APP` wishes to send a message to the neighbor nodes, `airplug` will broadcast a sixth field message composed of the sender application `APP`, its identity (if any), the keyword `ALL` or the name/address of the receiver, the receiver application or the keyword `ALL`. All messages have a *control* field used for piggybacking (eg. GPS data...) and a *payload* field used by the applications.

This architecture imposes no paradigm nor specific programming languages. The applications can be developed in any language, providing that they can send to `stdout` and receive to `stdin`.

III. Adaptation into ns-2

Ns-2 is implemented in C++ and OTcl (short for Object Tcl). The OTcl is a directed object extension of the script language Tcl/Tk. It is used to write simulation's scripts. The Tcl with Classes (TclCl) adds a C++ layer on the top of the OTcl layer with the aim of combining both classes. These classes are implemented in C++, as objects to be used in the environment of simulation. To analyse the results of simulation, the simulator outputs log files. In this section we describe how to add an Airplug code program as a routing agent in the network layer of ns-2. Furthermore, we describe the details of this code program. Finally, we illustrate who to use this code program by implementing a GPS application.

A. Airplug Routing Agent

When adding C++ module to ns-2, this module should be linked to Tcl. This enables Tcl to create a shadow object of the corresponding C++ object during simulation. A special static class that inherits from `TclClass` should be added in the C++ module. To achieve this goal a file called airplug.cc is created. Figure 2 illustrates the linkage in the `Class Agent_Airplug`.

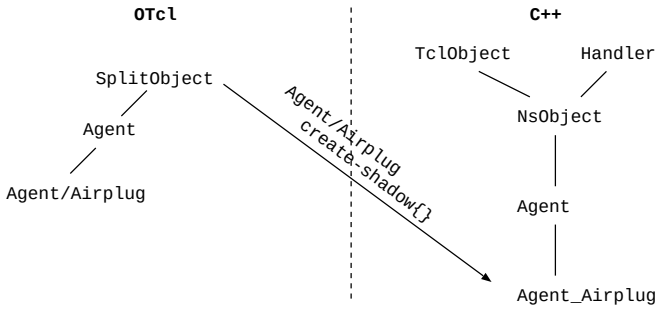


Fig. 2. Linkage between C++ and OTcl classes in airplug-ns.cc.

B. Communication primitive

The type of communication determines the contents of a message: only the part useful for the application, the address/sender application, the communication zone (localhost, AIR, hostname...). This feature is introduced with the aim of facilitating the creation of distributed applications. We distinguish three types of communication primitive:

- `what`: only payload
- `whatwho`: payload and application
- `whatwhowhere`: payload, application and communication range

Here are some examples of situation allowing to determine what is the most adapted type of communication.

To develop an application distributed in a scenario only of instances of this application, the mode `what` is quite indicated.

For example, during the study of a routing algorithm (such as AODV or OLSR), it is better to consider only the contents of messages.

To develop a distributed application in a scenario where this application should have a dialogue with another local application, then the type of communication `whatwho` is indicated. For example, to develop a routing algorithm of type geocast, the local instance will certainly use GPS positions supplied by the local application GPS.

Finally, the mode by default is `whatwhowhere`. It is then necessary to specify, besides the contents of the message and the application address, the reception zone: intern (LCH), extern (AIR), both (ALL), a specific machine (eg. "vehicle number 1",...).

The reason to use such communication primitive is to separate protocol implementation and ns-2 source code. Airplug-ns allows to make an abstraction layer which permit an easy reuse of the implementation code into a real plateform. For instance, when an application use the primitive `snd_what` the Airplug-ns source code call to the function `Target` (Respectively `Dmux`) to forward the message to up (respectively down) layer. If the application uses the primitive `rcv_what` the Airplug-ns source code call to the function `recv(packet)` and check if the message can be send to the corresponding local application.

C. Traffic generation

Airplug-ns uses a GPS application to generate microscopic traffic traces. For instance, a noted application `APP` who wants to use the module `GPS` has to call to the function `snd_whatwho`. The heart of airplug-ns (`airplug-ns.cc`) check whether the address of the destination application (value contained in `who`) is equal with that of the broadcasting address. Here, `airplug-ns.cc` redirects the primitive of communication towards the local application `GPS`. This application returns the latitude and the longitude of the mobile node which results from testbeds. Some modifications were brought in ns-2 to update the positions of the mobile node in real time. Such as real GPS module, this modification allows to update and recover the real position of a node each timer expiration. Furthermore, airplug-ns integrate into its library an implementation of a conversion of the Cartesian data towards the Polar coordinates and an implementation of the Harversein formula to compute the distance between two vehicles. Finally, Airplug-ns allows to use any microscopic traffic traces generated by a tool which permit an easy conversion into ns-2.

IV. Validation

We performed simulations of IEEE 802.11b communications between five to seven vehicles in convoy, as well as a real experiments of data transfer between many cars on the road.

A. Road experiments

The aim of these experimentations is to evaluate the performances of conditional transmissions, implemented through the

(a) (b) (c)

Fig. 3. The experimental testbed

`HOP` program. The conditional transmissions substitute conditions to addresses. By dynamically evaluating the conditions at the message reception, the protocol better fits to the dynamic than those relying on addresses. Indeed, using addresses in a dynamic network is a big challenge because many updates are necessary. Two kind of conditions are used. The forward condition (CFW) determines whether a message should be broadcasted by an intermediate node or not. The upward condition (CUP) determines whether a message should be given to an application layer or not. Conditions are fixed by the sending applications, and are included in the messages. Examples of conditions are: time or delay, geographical position, distance, trajectory matching (eg. being behind the sender). For more details on the conditional transmissions, refer to [9].

These experiments involved up to five vehicles in convoy; the first vehicle sends messages. The CFW condition has been set in order that messages progress from the first to the last vehicle in the convoy, by forcing retransmission on each intermediate vehicle.

B. Simulations

The simulation parameters are the same as the real experiment. We used version 2.33 of the simulator. We introduced into the simulator the parameters of our Alfa network awus036eh 802.11b card and our external antenna. The emission power is 200mw and the receiver threshold is equal to -85dBm. The antenna is fixed to the car's roof, yielding an approximate height of 1.5m, and having a gain of 7dBi. Figure 4 shows the Airplug-ns architecture used to make these simulations.

C. Propagation model

The propagation model has an important impact on the simulation results. We then used five different models, that we describe here.

To know if a packet is correctly received, ns-2 compares the power of the message with two thresholds RXThresh (Receive power threshold) and CSThresh (Carrier sense threshold). To calculate the receiver power, it uses the propagation function, which give this power according to the distance between two mobiles. If the received power is higher than the threshold RXThresh, the message is considered as received correctly. When the power is between CSThresh and RXThresh, the message is detected but cannot be understood. Finally if the power is lower than CSThresh, the message is not detected.

The radio distribution signal is the key to check whether the communication between two vehicles can be established. The

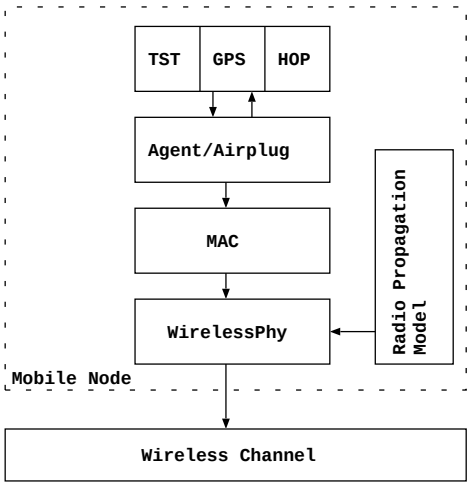


Fig. 4. Airplug-ns wireless simulation architecture

two-ray ground Reflection model considers both the direct path and a ground reflected propagation path between transmitter and receiver. Consequently, the formula for the power of reception against the distance also depends on the height of both antennas.

A more general model is the *shadowing model*. The model *shadowing* consists of two parts. The model of loss on a road (*path loss model*) predicts the power of reception for the distance d. The path loss exponent on the road β depends on the environment. To reach identical values with the model *free space* the value, $\beta = 2$ should be used. The second part of the *shadowing* model takes into consideration the power of reception. Random variation is reached by a zero-mean Gaussian distributed random variable X_{dB} with standard deviation σ_{dB}, more known under the name of *shadowing deviation*. We used two sets of parameters for the shadowing model.

An empirical model of the V2V channel path loss was given by [5]. The authors in [5] collected two sets of channel measurements at two different dates in a suburban V2V environment. Those measurements included the received signal strength (RSS) and the respective locations of the receiver and transmitter at the time each RSS measurement was taken. Then a dual slope piecewise linear model was used to approximate the path loss using linear regression on the measured date. Recently, in [7], this method has been implemented in ns-2 by using a pre-computed lookup table to generate the power envelope with minimal calculation in the simulator. We use this implementation for our tests.

V. Results

A sample of results is given in Figure 5 (more results and analysis will be presented in a further paper). The type of communications experimented on road and simulated is the broadcast mode. Let us note that the throughput of broadcast of the used card is 2Mbps. Knowing that the head vehicle sends a packet of 1024 bytes, each 100ms, the maximal receiver rate is equal to 80 Kbps. The theoretical transmission delay is equal to 5ms. Now, by forcing the relay in each vehicle of the convoy, we obtained results depending on the number of hops. Since the experiments were performed on the road, the environment varied a lot during the tests (traffic and surrounding variation).

In, the Tow ray ground model, the communication ranges are circles around the sender. So, all the vehicles which are in this circle will receive all the packets and those which are outside will not receive any packets. Let us note that all the vehicles in scenario 1 are static and are in the same communication range. Consequently, in the Two ray ground model, no packets loss is recorded (Figure 5(a)) and we observe a maximal throughput of 80 Kbps (Figure 5(c)). These results can be also explained by the fact that the inter-packets delay is big enough (the transmission delay is lower than the inter-packets delay). Indeed, the head vehicle will allow other one to retransmit again their packets before sending a new packet. With a inter-packets delay equal to 100ms, the loss rate, in the Two ray ground model, is null.

In opposite to the Two ray ground model, the Shadowing model does not use an ideal circle as communication range but a statistical model where vehicles can communicate only with a certain probability when they get closer to the limit of a communication range. When the Shadowing model is parameterised to be deployed in a noisy environment ($\beta = 3$, $\sigma_{dB} = 5$) the communication range can vary according to the congestion of the wireless channel. Consequently, we observe an increase of the loss rate (Figure 5(a)) and a decrease of the received throughput (Figure 5(c)) when the number of vehicles in the convoy increases. We observe the same tend-an with the experiment on road. However, the observed results under the noisy Shadowing model, is twice superior to those recorded on road. When the Shadowing model is parametrised with an free space environment ($\beta = 2$, $\sigma_{dB} = 3$) it shows the same results observed with the Two ray ground model.

Both Two ray ground and Shadowing model considers the path loss only with regard to the distance. Unlike these two models, the VanetProp model, take into counts the influence of the environment such as the Doppler shift and the fading. The relative motion between the sender and the receiver vehicle results in random frequency modulation due to different Doppler shifts on each of the multipath components. The VanetProp generates Nakagamu fading which has been shown to fit well to some urban multipath propagation data. When the vehicles are static and the distance between them is closer, there is few presence of reflecting objects and scatterers in the channel. So, the VanetProp model gives no packets loss and a maximal throughput when the convoy is stable (Figure 5(a) and Figure 5(c)). When, the vehicles are in suburban scenario, this model gives results which are close to those recorded on the road (Figure 5(b) and Figure 5(d)). In the same figures the other models gives results which are far from recorded on the road.

This analysis of the wireless communications under the sim-

ulator ns-2, shows that the propagation models are generally far from modelling a real environment, which vary a lot along the experimentation. Same heuristics can be add to improve the loss rate or the receiver throughput. These heuristics will allow to take into account the effects of the strong mobility and the wireless variation of VANETs.

VI. Conclusion and Perspective

This paper deals with bridging the gap between simulation and experimentation, which are of great important for designing VANET applications. A simulation platform has been presented. It relies on common network simulator and the Airplug software suite. This architecture allows easy prototyping of inter-vehicles applications and cross-layering protocols. To illustrate the interest of our solution, we compare the simulation and experiments of the performance of a given multi-hop protocol. Besides the performances evaluation, it is interesting to note the variation due to the environment along experimentations and simulation. Future road experiments have been planed to complete this study.

A large set of applications have been developed for Airplug by different contributors. All the developments (applications and protocols implemented in plugged processes) can be used with only few change, providing they have been written in Tcl/Tk. Many applications was adapted and other are being adapted to run into Airplug-ns. The Airplug-ns software suit is available for research teams [1]. We believe that the facilities provided by Airplug-ns can help in studying the scalability of the protocols. Moreover, simulations can take benefit from road measures.

Although the second distribution of NS is currently the most used, ns-3 has been released. Ns-3 is a new simulator with totally different architecture than ns-2. It is written entirely in C++ which remove some complexities of ns-2 due to its dual architecture. We plan to adapt Airplug to ns-3.

References

[1] Multiple Simulator Interlinking Environment (MSIE) for C2CC in VANETs. http://www.cn.uni-dusseldorf.de/projects/MSIE.

[2] The TraNS (Traffic and Network Simulator Environment). http://www.wiki.epfl.ch/trans.

[3] J. Blum, A. Eskandarian, and L. Hoffman. Challenges of Intervehicle Ad Hoc Networks. *IEEE Transaction on Intelligent Transportation Systems,*, 5:347–351, 2004.

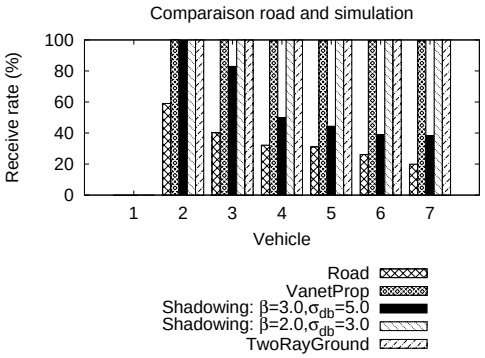
[4] P. Bucciol, E. Masala, N. Kawaguchi, K. Takeda, and J.C. De Martin. Performance evaluation of H.264 video streaming over inter-vehicular 802.11 ad hoc networks. In *Proc. of 16th Annual IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, Berlin, Germany, Sep. 2005.

[5] L. Cheng, B.E. Henty, D.D. Stancil, F. Bai, and P. Mudalige. Mobile Vehicle-to-Vehicle Narrow-Band Channel Measurement and Characterization of the 5.9 GHz Dedicated Short Range Communication (DSRC) Frequency Band Export Find Similar. *IEEE Journal on In Selected Areas in Communications*, 25(8):1501–1516, November 2007.

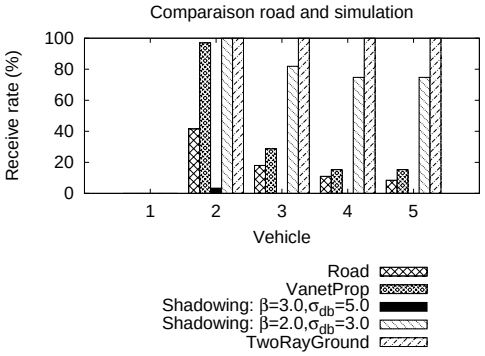
[6] N.D. Cottingham, J.I. Wassel, and K.R. Harle. Performance of IEEE 802.11a in Vehicular Contexts. In *Vehicular Technology Conference,2007. VTC 2007-Spring*, Dublin, Ireland, 2007.

[7] G. Dib. *Vehicle-to-Vehicular Channel Simulation in a Network Simulator*. Thesis Master of Science in Information Networking, Carnegie Mellon University, 2009.

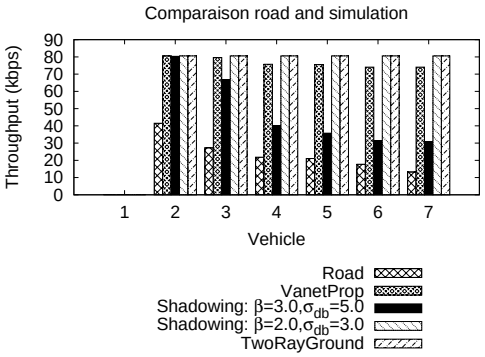
[1] http://www.hds.utc.fr/ ducourth/airplug/doku.php?id=fr:dwl:ns:accueil



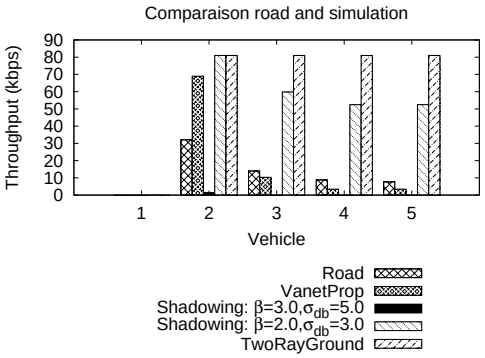
(a) (stop)



(b) (move)



(c) (stop)



(d) (move)

Fig. 5. Loss and throughput obtained with Airplug-ns and road in convoy

[8] B. Ducourthial. About efficiency in wireless communication frameworks on vehicular networks. In *ACM WIN-ITS workshop co-located with IEEE ACM QShine*, Vancouver, Canada, 2007.

[9] B. Ducourthial, Y. Khaled, and M. Shawky. Conditional transmissions: a communication strategy for highly dynamic vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology, special issue on vehicular communication networks,*, 56(6):3348–3357, November 2007.

[10] B. Ducourthial and S. Khalfallah. A platform for road experiments. In *Proc. IEEE 69th IEEE Vehicular Technology Conference (VTC 2009-Spring)*, Barcelona, Spain, April 2009.

[11] R. Gass, J. Scott, and C. Diot. Measurements of in-motion 802.11 networking. In *In Proc. WMCSA*, Apr. 2006.

[12] F. Hui. *Experimental Charaterization of Communications in Vehicular Ad Hoc Networks*. Thesis master of science in computer science, University of California Davis, 2001.

[13] D. Krajzewicz, M. Bonert, and P. Wagner. The open source traffic simulation package SUMO. In *RoboCup 2006 Infrastructure Simulation Competition*, 2006.

[14] Network simulator. http://www.isi.edu/nsnam/ns.

[15] J. Ott and D. Kutscher. Drive-thru internet: IEEE 802.11b for automobile users. In *In IEEE INFOCOM*, 2004.

[16] C. Schroth, F. Dotzer, T. Kosch, B. Ostermaier, and M. Strassberger. Simulating the traffic effects of vehicle-to-vehicle messaging systems. In *Proc. of the Fifth IEEE International Conference on ITS Telecommunications*, Brest, France, 2005.

[17] J.P. Singh, N. Bambos, B. Srinivasan, and Clawin D. Wireless lan performance under varied stress conditions in vehicular traffic scenarios. In *In Proceedings. IEEE 56th Vehicular Technology Conference*, volume 2, pages 743–747, 2002.

[18] S.Y. Wang and C.L. Chou. Nctuns tool for wireless vehicular communication network researches. *Elsevier, Simulation Modelling Practice and Theory*, 17(7):1211–1226, August 2009.

[19] H. Wu, J. Lee, M. Hunter, R. Fujimoto, R.L. Guensler, and J. Ko. Efficiency of simulated vehicle-to-vehicle message propagation in atlanta. *Journal Transportation Research Record: Journal of the Transportation Research Board*, 1910/2005:82–89, 2006.