Vehicular networks emulation

Anthony Buisset, Bertrand Ducourthial, Farah El Ali, Sofiane Khalfallah

Vehicular networks emulation

A. Buisset, B. Ducourthial, F. El Ali, S. Khalfallah
(1) Université de Technologie de Compiègne (2) CNRS Heudiasyc UMR6599,
Centre de Recherche de Royallieu, B.P. 20529, Compiègne, France (corresponding author: ducourth@utc.fr)

***Abstract*—Many applications and protocols are planned for the so-called Intelligent Transportation Systems (ITS). Most of them are supposed to work in dynamic networks, such as the vehicular ad hoc networks (VANET). However, designing and studying distributed applications and protocols in such networks is not easy. Analytical studies suffer from the lack of pertinent models. Simulations are often far from reality. Road experiments are generally limited, due to their complexity.**

In this paper, we present an environment that emulates the vehicular networks. It allows to reproduce road experiments without further developments of the studied prototypes. These protocols can be tested with more complex road traffic. The impact of the communication range and the dynamics of the network can be studied. Some comparisons with road tests and simulations show the advantage of such an emulation framework.

I. Introduction

Nowadays, Intelligent Transport Systems (ITS) attract much attention. ITS applications do indeed increase road safety and transport efficiency, limit the impact of vehicles on the environment, improve the overall productivity... A lot of inter-vehicle applications are considered; most of them rely entirely or partially on the vehicular network: emergency facilities, crash prevention, collision avoidance, traffic jam management, Internet access, infotainments (travel and tourist information, chats, distributed games...).

However the Vehicular Ad hoc Network (VANET) exhibits characteristics that are dramatically different from many generic MANET (Mobile Ad hoc networks) [1]. The topology varies very frequently due to vehicle movement or connection losses. The loss rate is often high. The speed, the density and the type of movement depends on the kind of roads. New and specific protocols are often required to reach acceptable performances, and important open issues remain to be solved.

It is worth noting that the design and the study of new protocols or distributed applications is not easy. Analytical studies require accurate models to take into account the dynamics of the network. However it is not easy to model all the variations (speed, density, topology, loss rates...). Analytical studies are generally used for highway scenarios, where the network dynamics are low or at least predictable.

Simulations can take into account complex road traffic, generated by specific traffic generators. However simulation requires some simplifications regarding the propagation model, or the studied protocols, which are generally far from those used on the road. The performances obtained by simulation are then often different from real measures. Simulations are generally used for scalability studies and comparisons.

Road experiments give accurate performance measures in realistic situations. However they generally involve few vehicles and remain quite rare. Indeed such experiments are tedious; they require many people, vehicles and equipment. Moreover, they are generally not reproducible because the scenario depends on the other vehicles. Also, the inter-vehicle distances are difficult to control, and wireless communications are subject to environmental factors, including weather and trucks... Road experiments are generally used for punctual measures and proof of concepts.

As we can see, there is a lack of tools facilitating the design and the study of protocols and applications dedicated to vehicular networks. Analytical studies, simulations and real experiments all have their limitations, which are enforced by the dynamics of vehicular networks.

In this paper, we show that emulation is a convenient way to study such networks. During an emulation, some parts are real, while some others are artificially reproduced by several means. We present a powerful framework, named *airplug-emu*, which is very close to real road experiments while still easy to use. It allows fast prototyping and accurate performance measures.

The Airplug-emu framework is based on the Airplug software suite dedicated to the study of dynamic networks and road experiments [2]. We summarize this software architecture in Section III. We show how such an architecture can be used to emulate the vehicular network in Section IV. In Section V, we explain how realistic mobility scenarios can be built, thanks to the `GPS` application. The emulation tool, called `EMU`, is detailed in Section VI. Comparisons with road tests and simulations are analyzed in Section VII. Concluding remarks end the paper. We shall now begin by summarizing related work.

II. Related work

Overview. An emulator is a tool that combines real and artificial implementation (simulated or emulated). A lot of studies show that emulation is interesting both for wireless and wired network emulation. This paper deals with wireless emulation. Wireless network emulators allow quick indoor deployment and protocol testing without requiring physically moving the nodes. According to [3], emulators can be sorted in two categories: physical layer emulators and MAC layer emulators. We will now follow up by summarizing related work in both categories, to target the differences and position our work.

Physical layer emulators. In physical layer emulators, all the network layers, except the physical one, are real. The physical layer can be emulated by attenuating the signal using programmable Radio Frequency (RF) attenuators [4]. Other emulators use RF wires to connect two pairs of nodes [5], [6], [7]. This type of connectivity does not allow to take into account the impact of outside factors, such as interferences due to multipath. Moreover, these emulators do not implement mobility simulation. The physical layer emulator EWANT [5] can attenuate the sent signal of the sender using RF wires while emulating the node mobility thanks to different antennas. The emulator MiNT [7] takes into account interferences using non programmable radio signal attenuators and handles the mobility by positioning nodes on remote controlled robots.

The emulator ORBIT [8] attenuates the sent signal by injecting white Gaussian noise in the environment. It relies on a grid of 400 nodes in a $20\,m^2$ area. The nodes mobility is simulated by a server that activates different nodes at different times. The activated node corresponds approximately to the geographical location the node would have at the same moment in time.

In [9], another technique is used for emulating the physical layer: the radio signal is digitized, modified to add radio propagation effects and re-injected into the network interfaces.

MAC layer emulators. Concerning the MAC layer emulators, all the network layers are real, except the physical and the MAC ones. MAC layer emulation permits to determine whether a node should receive a packet or not. In other terms, if in the emulator, a node is in the neighborhood of other nodes, then it should receive the packets coming from these nodes. In the other cases, all the received packets are deleted. This emulation is performed by using a filter tool, either centralized [10], [11], [12] or distributed into each node of the network, using `iptables` [13], [14] or specific filter tools [15], [16].

In addition to the unique admission control of a packet, other features can be added to the filter tool, such as acceptance or suppression, modification or delay of the received packets [17], [18]. In this type of emulator, the real system behavior is measured and used as input of the filter tool.

Hybrid emulators. Lastly, hybrids emulators are proposed in [19], [20], [21], [7]. These emulators combine emulation, real equipment and/or simulation. For instance, higher network layers can be simulated using a network simulator while low layers operate on real wireless devices.

Airplug-emu, the emulation framework presented in this paper, is mainly a MAC layer emulator, designed for vehicular ad hoc network studies. It can also be hybrid, by allowing the integration of real wireless links in the emulation.

III. Architecture for road experiments: Airplug

The airplug-emu emulation framework is part of the Airplug software suite [22], designed for experimentation in dynamic ad hoc networks [23], [2]. It relies on a core program and a set of applications. We summarize here its main characteristics.

The core program named `Airplug` manages the inter-applications communications, either local-to-the-host or inter-vehicle. The applications (`GPS` and `PRO` in Figure 1) are plugged on top of the core program; they reach the network through Airplug. The core program and the applications run in independent user-space processes for robustness and portability reasons.

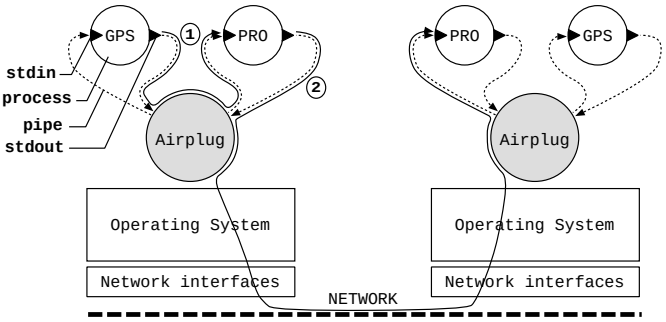

Fig. 1. Airplug architecture, intra- (1) and inter- (2) vehicle communications.

The communications are handled in the easiest and most robust way, by using standard input and output for receiving and sending messages respectively. This guarantees complete independence from the programming language used to develop the applications. For each process launched by Airplug, the standard input and output are redirected from and to Airplug by pipes (Figure 1). Thus, each time a process writes in its standard output, Airplug receives the data, and each time Airplug writes in the pipe, the process can read the data from its standard input. Since the network interfaces are handled by Airplug, the applications access the network in the same way they would to communicate with other local applications, by writing in their standard output. Airplug forwards the data to the network interface, preventing any network resources abuse by bugged applications.

Messages use a specific addressing format, well adapted to dynamic networks. The destination of a message is specified with an *area* and the *name of the destination application*. This addressing scheme is closed to the one in the WAVE Short Messages Protocol (WSMP) [24].

The area can be local (keyword `LCH` for localhost) or external (in other words composed by cars in the neighborhood; keyword `AIR`), or both (keyword `ALL`). But it can also be more specific (name or address of a nearby vehicle). A message coming from a given application can be sent to many other applications by filling in the destination application field with the keyword `ALL`. However, by default an application `A` receives only messages addressed to it and sent by a local application `B`. To receive messages sent by `B` addressed to `ALL` applications, `A` must first subscribe to the messages of `B` by contacting Airplug. Similarly, to receive messages sent by a remote application `C`, the application must first subscribe to them, even if they where sent directly to it. This registering system (relative confidence locally, and limited confidence remotely) allows an application to control its receptions. It also

increases the architecture's robustness by avoiding chained problems in case of bugged applications.

It is worth noting that the design of new protocols as well as cross-layering architectures is facilitated, by programing in user-space. Airplug can bypass the operating system protocol stack using raw sockets.

IV. From road to emulation: airplug facilities

The Airplug architecture has been designed to offer a simple, portable and robust framework for experimenting in vehicular networks (and in any other dynamic network). In this section, we explain that such an architecture is also very convenient for network emulation.

Since applications run in independent processes and communications rely on standard input/output, the network can be emulated using shell facilities. For example, communications (1) and (2) in Figure 1 can simply be reproduced in a shell by the command:

```
./gps | ./pro | ./pro
```

A bidirectional link between two applications such as PRO ↔ PRO can be emulated with named pipes:

```
mkfifo link1 link2              creates 2 named pipes
./pro < link1 > link2           launches first app. PRO
./pro < link2 > link1           launches second app. PRO
```

Mobility is emulated by changing pipe redirects, by deletion or by creation. In order to avoid packet loss, a *gateway* is inserted, using the `cat` command: by temporarily freezing this process, messages are stored in its input pipe while its output pipes are modified. The following script creates a loop of two vehicles running the protocol PRO, and then inserts a third vehicle.

To create a network connecting two vehicles in a loop:

```
mkfifo in1 in2 out1 out2 gtw1 gtw2          Pipes
./pro < in1 > out1 &                        Creates the vehicles
./pro < in2 > out2 &
cat out1 > gtw1 &                           Connectes the gateway
pid_cat1=$!                 ($! = process id. of the last process)
cat out2 > gtw2 & ; pid_cat2=$!
tee in1 < gtw2 & ; pid_tee1=$!              Creates the loop
tee in2 < gtw1 & ; pid_tee2=$!
tee in1 &                                   Starts the communications
```

To create a third vehicle to be inserted in the loop:

```
mkfifo in3 out3 gtw3
./pro < in3 > out3 &
cat out3 > gtw3 & ; pid_cat3=$!
```

To freeze the gateways and new connections:

```
old_pidtee1=$pid_tee1 ; old_pidtee2=$pid_tee2
kill -STOP $pid_gtw1 $pid_gtw2 $pid_gtw3
tee in2 in3 < gtw1 & ; pid_tee1=$!
tee in1 in3 < gtw2 & ; pid_tee2=$!
tee in1 in2 < gtw3 & ; pid_tee3=$!
```

To unfreeze the gateways and to remove old connections:

```
kill -CONT $pid_gtw1 $pid_gtw2 $pid_gtw3
kill -KILL $old_pidtee1 $old_pidtee2
```

As we can see, simple shell scripts can easily reproduce dynamic network topologies. No modification is required for the applications and protocols, providing they have been developed with the Airplug rules (independent processes and communications using standard input and output). Such scripts are very convenient, robust and powerful for prototyping. Nevertheless, their writing has to be automated when scenarios become complex and when real vehicle locations have to be used. This is the aim of the EMU program, presented below.

V. Generating realistic trajectories: the GPS app.

In order to populate the emulated vehicular network with realistic vehicle locations, we developed the GPS application.

The GPS application is above all an acquisition tool, able to retrieve geographic locations when receiving NMEA frames from a GPS device connected to the computer. As an Airplug compatible application, it allows in real time to send the current vehicle location to local applications that subscribed to it.

In order to reproduce road tests in a lab, the GPS application can save the locations in a file during the road tests. Then, when using the GPS application in a shell script that emulates the vehicular network, it can read the file of locations at a given frequency, and write them in its standard output to send them to connected applications.

However, a road test relies on fewer vehicles than the number desired during the emulation. To circumvent this problem, the GPS application can generate new realistic trajectories starting from a file of locations obtained on the road. Generated locations are calculated using a weighted barycenter from each successive pair of positions read in the file. This barycenter is modified by a random factor, truncated if necessary to prevent the new position from leaving the interval between the two original positions. This avoids sudden reverses while adding some irregularity in the node movement. The generated locations can be saved in a file and/or written in the standard output to be sent to connected applications. Thanks to this feature, a single log obtained on the road can lead to long realistic convoys of vehicles in the emulation.

VI. Emulation tool: the EMU application

In this section, we describe the EMU application. It automates the emulation and offers powerful features.

Overview. The EMU application aims to perform realistic experiments of protocols and applications designed for vehicular networks. With EMU, the applications and protocols to be studied run on independent processes as they do during road experiments, without any modification. EMU handles all the communications, either intra- or inter-vehicle, by using the shell facilities, as explained in Section IV. This application is written in Tcl/Tk and runs on a Linux PC. Several computers can be used in order to introduce real links instead of emulated ones (hybrid emulation).

Scenario. The scenario of the test is described in an XML file, that indicates the number of vehicles, the size of the geographic area, the applications and protocols running in each vehicle, the trajectory of each vehicle, and so on. The trajectories are produced with the GPS applications, by using real positions as explained in the previous section. However,

EMU accepts other input for node mobility, produced by traffic generators such as VanetMobiSim [25]. The Network Simulator format is also accepted. This allows to compare road experiments and simulations with the emulation, as we shall do in the next section.

Link. EMU reads the position of each node in the network with a user-defined frequency. The communication links are determined by the wireless communications range (user-defined) and a random factor hazard. If the distance between two vehicles is less than range × hazard, then there is a link. The hazard permits to add (or not) a variation in the antenna scope (to avoid perfect discs). It is also possible to use node-specific ranges, which is useful for some VANET security studies. The links affected by the movement of a node v are determined by checking each node position located in the range of v. Node locations are stored in lists sorted by x and y axis; the algorithm complexity is generally lower than $O(n^2)$ because neighbors of v are searched in a square centered on v with a side equal to $2 \times$ range. This avoids checking each pair of nodes at each move.

Network emulation. The emulation of the dynamic network generalizes the scheme explained in Section IV. For each vehicle, EMU launches the applications and protocols specified in the XML file with the related command line, so that they run in independent processes (TST and HOP in Figure 2). The standard input of these processes are connected to a reception process RCP and their standard output are connected to a directional process DIR. The first one receives all the inter- and intra-communications. The second one forwards the messages either for local applications or for neighbor vehicles through the gateway process GTW.

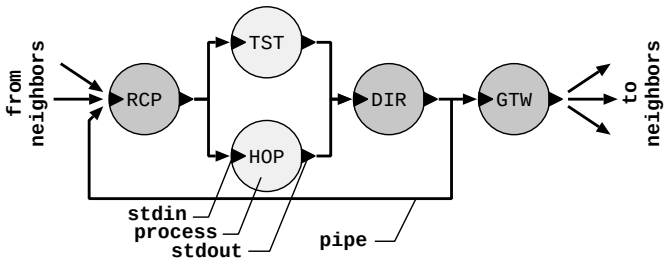

Fig. 2. Two applications (TST and HOP) in an emulated vehicle.

The RCP process is a Tcl shell script that forwards intra-vehicle messages. The GTW process is implemented with the cat command. As previously explained, it is used to change the inter-vehicle connections without packet losses (if a perfect network is desired): first GTW is frozen, next the inter-vehicle links are changed, and then GTW is unfrozen and the messages waiting in its input are sent without losses. The DIR process is a Tcl shell script that analyzes the header of the messages sent by the local applications in order to determine whether they should be sent locally (keyword LCH) or to neighbor cars (keyword AIR) or both (keyword ALL).

Realistic emulation. All inter-process links including inter-vehicle links (from a GTW process on vehicle A to a RCP process on vehicle B) rely on shell named pipes. In order to reproduce the conditions of communication observed on the road, the RCP process can delay or lose inter-vehicle messages. It then accepts two parameters (delay and lossrate); such values can be measured during road experiments.

It is also possible to run an emulated vehicle on another computer, thanks to the *remote* mode. In this mode, the messages generated by a remote application reach the others through a socket. This socket can be established on any real network (Ethernet, 802.11a/b/g/p and so one) in order to include real links in the emulation (hybrid emulation).

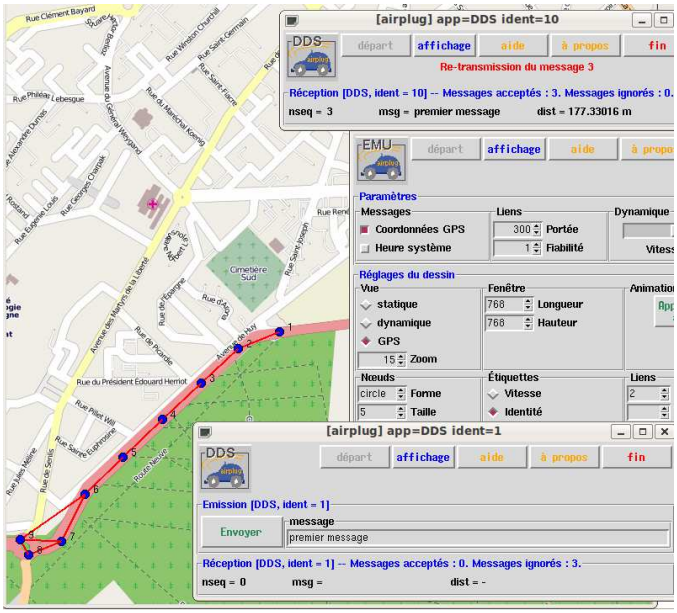

Fig. 3. Emulation of the DDS application on a convoy of 9 vehicles generated by the GPS application from a real GPS trajectory obtained on road N131, Compiègne, France. We can see the DDS applications running on vehicles 1 and 10, as well as parameters of EMU. Maps by *OpenStreetMap.org*.

Output. The EMU application offers several outputs, which can be used simultaneously. First, it provides a graphical representation of the moving vehicles as well as the links between them. When the positions are read from GPS coordinates, EMU downloads OpenStreetMap tiles [26] in order to geographically locate mobile nodes (see Figure 3). This display is useful to validate generated mobility scenarios and to study the network topology depending on the range and the reliability criteria (which can be changed on-line).

Second, EMU is able to launch the applications of all vehicles. An option allows to iconify some of them in order to focus on the most interesting for the current study. This is the main use of EMU (Figure 3). It allows to test applications in an environment similar to the road. Communication links are modified as the emulation progresses and the vehicle positions change. The messages that are exchanged between applications and protocols circulate through named pipes, either inside emulated vehicles, or between emulated vehicles. Each vehicle can be set to delay or loose messages in order to reproduce wireless communication links. The dynamics of the network

can vary on-line thanks to a `dynamic` parameter in order to study the robustness of protocols in function of different levels of network dynamics. This parameter increases or decreases the vehicle locations updates whereas messages always take the same time to reach their destination.

Third, EMU can generate a shell script similar to the one described in Section IV. It contains the commands required to reproduce the vehicular network emulation by means of shell facilities. This allows to reproduce the emulation without requiring the emulator to reuse the scenario with other applications or parameters, simply by changing the first lines of the script.

Finally, logs can be generated by the applications and by EMU for further analysis.

VII. Validation: comparisons with road and ns-2

In this section, we evaluate the accuracy of the emulation framework, by comparing results between simulations and real tests, either in lab or on the road.

Scenarios. Three real tests were used for this validation: a 7-car stopped convoy, a 5-car moving convoy and a 4-hop in-lab test. Each time, the first vehicle sends some packets and the others forward them until the end of the convoy. The in-lab test was done using 4 PC's forming a loop, in order to obtain very precise measures of the time delays (departure and arrival of messages on the same computer).

We measured the delays, the loss and the throughput using the TST application that generates packets with given sizes and inter-packet gaps (for the first vehicle), and performs many measures at reception (for other vehicles). The measures were done at the application level. At the routing level, the HOP protocol ensures that the packets progress from vehicle to vehicle in the convoy and from PC to PC for the in-lab test [27], [2]. At the low layers, packets are simply broadcast in the vicinity of each sender (802.11 broadcast at 2 Mbit/sec).

These experiments were reproduced with Airplug-emu, using the same applications (Figure 2). As explained in the previous section, EMU is able to delay or lose messages depending on the given input (measured on the road), in order to mimic the real wireless communication. We evaluated the impact of using either accurate input or average input. The first case is expected to give better results than the second even though the second remains more convenient for the user. Accurate inputs means that, for each emulation, the loss rates and delays measured during the corresponding real test are given to EMU. Average input means that a single delay and a single loss rate have been given to EMU for all the emulations; these are the average values observed during all the real tests.

The tests have also been reproduced with a simulator in the aim of evaluating the advantages of emulation to those of simulation, from the point of view of performance measures. We used Network Simulator (ns version 2.33), a well known and largely used simulator. To generate the packets, a Constant Bit Rate source was used. Packet forwarding was done with an implementation of the HOP forwarding application. The standard *two rays ground* propagation model was used. Other ns-2 parameters were chosen to reproduce real conditions (Table I).

Parameters	Values
Antenna height	1.5 m
Inter-Packet Gap (IPG)	100 ms
Packet size	1000 bytes
Data rate	1 Mb/s
Communication range	520 m
Propagation model	Two-Ray ground reflexion

TABLE I
General parameters for simulations

Reproducing in-lab tests. We compared real in-lab tests, ns-2 simulations and emulations performed with Airplug-emu. In order to study the impact of accurate versus average inputs for EMU, two sets of emulations were done. We varied the inter-packet gap (IPG) from 20 ms to 100 ms with a step of 10 ms and from 100 ms to 1000 ms with a step of 100 ms in order to check the ability of EMU to handle many simultaneous packets and to reproduce conditions close to loaded networks. The packet size is fixed to 1000 bytes and the experience duration is 200 seconds.

Figure 4-left displays the results for accurate inputs, while Figure 4-right displays the results for average inputs (in-lab and ns-2 results are the same on the left and on the right).

We observe that EMU is very close to real tests with accurate inputs. This validates the Airplug-emu framework: it is able to faithfully reproduce real experiments, providing that average inputs regarding loss and delays are given.

Nevertheless, using accurate inputs requires that we perform the related real tests to measure the loss rate and the delays on each node. This is interesting when an experiment needs to be replayed several times by emulation to fit specific parameters for instance. When the emulation framework is used for performance studies on scenarios larger than those realized on the road, this is no more practical. In this case, it is much more convenient to set EMU with average values observed during representative real tests.

We observe that, with average values, EMU is close to real tests for inter-packet gaps larger than 100 ms (Figure 4-right). Note that this value is sufficient for many protocols relying on beaconing packets as well as for many VANET applications, including safety related applications [28].

We also observe that EMU gives better results than ns-2, whose results remain far from real test results, especially for delays (close to 0) and loss-rates. This can be explained by the fact that ns-2 does not take into account the collision probability due to node proximity. The throughput given by ns-2 is close to those of real tests for large inter-packet gaps.

We conclude that the Airplug-emu framework is an interesting tool for measuring performances: with accurate inputs, it is very precise and with average inputs, it is precise for inter-packet gaps larger than 100 ms.

Reproducing road tests. We compared real road tests, ns-2 simulations and emulations performed with Airplug-emu. In order to study the impact of varying environments on EMU,

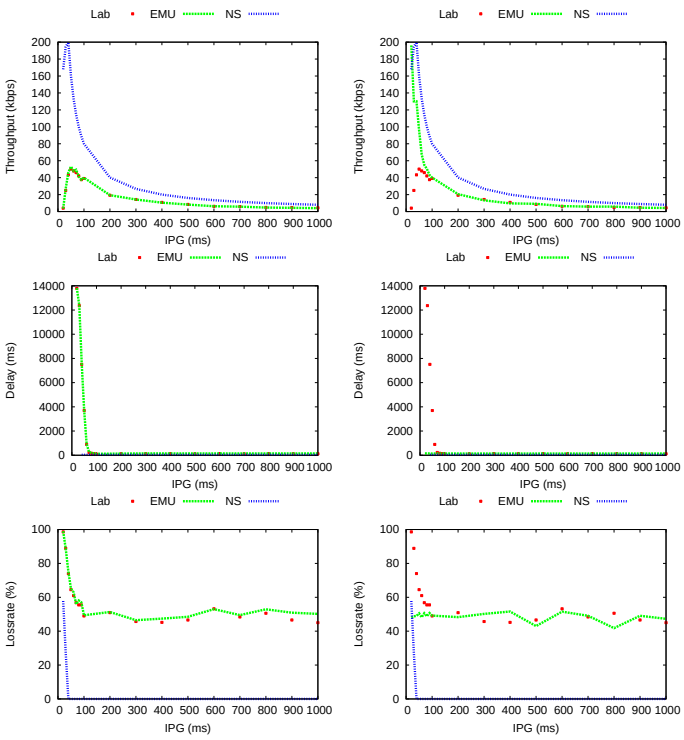

Fig. 4. Comparisons of in-lab tests, EMU and ns-2 depending on the inter-packet gap (IPG) in milliseconds. On the left, accurate inputs for EMU. On the right, average inputs. From top to bottom: throughput, delay, loss-rate.

two sets of comparisons were done. Both concern a convoy of vehicles, allowing to study the impact of the number of hops on the results. The first set consists of a stopped convoy of 7 vehicles, leading to a stable environment in regards to wireless communication. The second set consists of a moving convoy of 5 vehicles offering a varying environment for communication. The speed was approximately stable during all the tests (around 70 km/h). Only the average delay and loss-rate measured on the road has been used as input for EMU (average input). Following conclusions of the in-lab study, the inter-packet gap was fixed to 100 ms. The packet size was equal to 1000 bytes.

Figure 5 displays results for the fixed convoy (on the left) and the moving convoy (on the right). We observe that EMU gives results which are closer and closer to real tests. Even though mean values for delay and loss-rate were used, EMU is able to give the trend.

We can notice that the delays and the loss-rates measured during road tests are larger in the moving convoy than in the fixed one. The throughput is then better in stopped convoy (almost twice the throughput of the moving convoy). This is due to the varying communication environment and the many connection losses (due to trucks for instance). Nevertheless, we can see that EMU is not really sensitive to such a varying communication environment: results are not far from reality even in the moving convoy, even though mean input were used.

To the contrary, ns-2 gives results far from those measured on the road. The delay is badly estimated and the loss-rate is close to 0. As a consequence, the throughput is maximal (1000 bytes per packet with an inter-packet gap of 100 ms leads to a throughput of 80 kbits/sec). This may be explained by the fact that ns-2 does not take into account the environment.

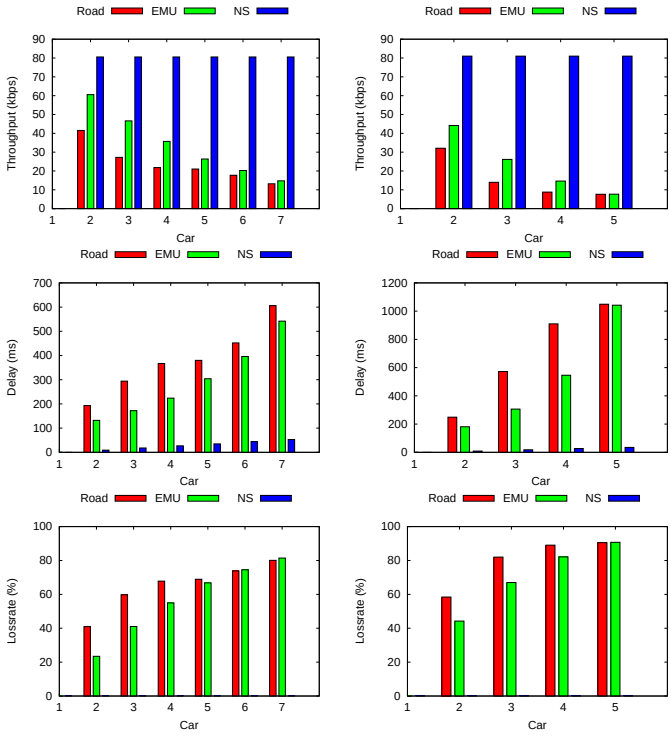

Fig. 5. Comparisons of road tests, EMU and ns-2 with average inputs for EMU. On the left, a stopped convoy of 7 vehicles. On the right, a moving convoy of 5 vehicles. From top to bottom: throughput, delay, loss-rate.

VIII. Conclusion

In this paper, we proposed an environment for vehicular networks emulation. It allows to prototype, to test and to tune protocols. It can faithfully reproduce road experiments. It can also be used to study application scaling.

This environment accepts any applications and protocols that use the very simple *Airplug software suite* conventions: run in independent processes, receive the messages by reading standard input, send the messages by writing in standard output. Realistic traffic scenarios can be generated using the GPS application, traffic generators or ns-2. The emulation relies on the shell facilities, which makes it powerful, robust and portable. The emulator EMU offers a good graphical representation of the vehicular network; it allows to study the impact of the network dynamic and the wireless communication range on the protocols. Hybrid emulation is possible by including real wireless links, thanks to the remote mode.

We believe that emulation is promising for vehicular networks studies. The Airplug-emu framework is available on demand for research and teaching. This tool has been used successfully by several research teams (academic and industrial) as well as for teaching (distributed algorithms, vehicular networks).

REFERENCES

[1] J. Blum, A. Eskandarian, and L. Hoffman, "Challenges of intervehicle ad hoc networks," *IEEE Transaction on Intelligent Transportation Systems*, vol. 5, pp. 347–351, 2004.

[2] B. Ducourthial and S. Khalfallah, "A platform for road experiments," *Proc. of the 69th IEEE Vehicular Technology Conference (VTC2009-Spring)*, April 2009.

[3] W. Kiess and M. Mauve, "A survey on real-world implementations of mobile ad-hoc networks," *Ad Hoc Netw.*, vol. 5, no. 3, pp. 324–339, 2007.

[4] D. Beyer, "Accomplishments of the DARPA SURAN program," in *Proceedings of the IEEE MILCOM 90 Conference*, California, October 1990.

[5] S. Sanghani, T. Brown, S. Bhandare, and S. Doshi, "Ewant: The emulated wireless ad hoc network testbed," in *Proceedings of the IEEE WCNC*, 2003, pp. 1844–1849.

[6] C. Kwan-Wu, J. John, W. Aidan, and K. Roger, "Implementation experience with manet routing protocols," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 5, pp. 49–59, 2002.

[7] P. De, A. Raniwala, S. Sharma, and T. Chiueh, "Mint: A miniaturized network testbed for mobile wireless research," in *Proceedings of the IEEE INFOCOM*, 2005, pp. 2731–2742.

[8] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh, "Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols," in *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC)*, March 2005, pp. 2731–2742.

[9] J. Glenn and S. Peter, "Repeatable and realistic wireless experimentation through physical emulation," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 63–68, 2004.

[10] Q. Ke, I. David, D. Maltz, and D. B. Johnson, "Emulation of multi-hop wireless ad hoc networks," in *Proc. of the 7th International Workshop on Mobile Multimedia Communications (MoMuC)*, 2000.

[11] T. Lin, S. F. Midkiff, and J. S. Park, "A dynamic topology switch for the emulation of wireless mobile ad hoc networks," in *Proc. of the 27th Annual IEEE Conference on Local Computer Networks (LCN)*. Washington, USA: IEEE Computer Society, 2002, p. 0791.

[12] J. Flynn, H. Tewari, and D. O'Mahony, "Jemu: A real time emulation system for mobile ad hoc networks," in *Proc. of the First Joint IEI/IEE Symposium on Telecommunications Systems Research*, November 2001.

[13] Z. Yongguang and L. Wei, "An integrated environment for testing mobile ad-hoc networks," in *Proc. of the 3rd ACM international symposium on Mobile ad hoc networking & computing (MobiHoc)*. New York, USA: ACM, 2002, pp. 104–111.

[14] M. Matthes, H. Biehl, M. Lauer, and O. Drobnik, "Massive: An emulation environment for mobile ad-hoc networks," *Proc. of the Second Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, vol. 0, pp. 54–59, 2005.

[15] D. A. Maltz, J. Broch, and D. B. Johnson, "Experiences designing and building a multi-hop wireless ad hoc network testbed," School of Computer Science, rapport de recherche CMU-CS-99-116, Carnegie Mellon University, 1999.

[16] M. Heissenbuttel, T. Braun, T. Roth, and T. Bernoulli, "Gnu/linux implementation of a position-based routing protocol," in *Proc. of the IEEE ICPS Workshop on Multi-hop Ad hoc Networks: from theory to reality (REALMAN)*, July, 2005.

[17] B. Noble, M. Satyanarayanan, G. Nguyen, and R. Katz, "Trace-based mobile network emulation," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 4, pp. 51–61, 1997.

[18] J. Liu, Y. Yuan, D. Nicol, R. Gray, C. Newport, D. Kotz, and L. Perrone, "Simulation validation using direct execution of wireless ad-hoc routing protocols," in *Proc. of the eighteenth workshop on Parallel and distributed simulation (PADS)*. New York, USA: ACM, 2004, pp. 7–16.

[19] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: accurate and scalable simulation of entire tinyos applications," in *Proc. of the 1st international conference on Embedded networked sensor systems (SenSys)*. New York, USA: ACM, 2003, pp. 126–137.

[20] L. Girod, N. Ramanathan, J. Elson, T. Stathopoulos, M. Lukac, and D. Estrin, "Emstar: A software environment for developing and deploying heterogeneous sensor-actuator networks," *ACM Trans. Sen. Netw.*, vol. 3, no. 3, p. 13, 2007.

[21] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, and T. Schoellhammer, "A system for simulation, emulation, and deployment of heterogeneous sensor networks," in *Proc. of the 2nd international conference on Embedded networked sensor systems (SenSys)*. New York, USA: ACM, 2004, pp. 201–213.

[22] "Airplug software suite," http://www.hds.utc.fr/∼ducourth/airplug.

[23] B. Ducourthial, "About efficiency in wireless communication frameworks on vehicular networks," in *Proceeding of the ACM WIN-ITS workshop (with IEEE ACM QShine'07)*, 2007.

[24] "Trial use standard for wireless access in vehicular environments (WAVE) – architecture," IEEE, 2007.

[25] "Vanetmobisim," http://vanet.eurecom.fr/.

[26] "Open street map," http://openstreetmap.org.

[27] B. Ducourthial, Y. Khaled, and M. Shawky, "Conditional transmissions: performances study of a new communication strategy in VANET," *IEEE Transactions on Vehicular Technology, special issue on vehicular communication networks*, vol. 56, no. 6, pp. 3348 – 3357, November 2007.

[28] S. Olariu and M.-C. Weigle, Eds., *Vehicular Networks: From Theory to Practice*, ser. CRC Computer & Information Science Series. Chapman & Hall, 2009.