Validation of a Vehicle Emulation Platform Supporting OBD-II Communications

Oscar Alvear, Carlos T. Calafate, Juan-Carlos Cano, Pietro Manzoni
Department of Computer Engineering (DISCA)
Universitat Politècnica de València, Spain
e-mail: oalvear@gmail.com, {calafate, jucano, pmanzoni}@disca.upv.es

***Abstract*—In the next few years, important developments are expected in the Intelligent Transportation Systems (ITS) area. One of the key issues enabling future solutions is achieving an effective integration between mobile apps and vehicles. Such integration can be efficiently achieved on all existing vehicles by relying on the On Board Diagnostic (OBD-II) interface. This allows obtaining critical information such as speed, fuel consumption, gas emissions and system failures. In this paper we propose a vehicle emulation platform, called VEWE, that allows developing and testing OBD-II aware applications. The advantages of this approach include: avoiding the need for a real vehicle, allowing to easily generate realistic vehicle parameter patterns, and supporting emulated GPS functionality. We evaluate our platform by conducting a performance analysis in terms of OBD-II response times and channel capacity when relying on a Bluetooth adapter. We compare our results with respect to those obtained in real vehicles, and demonstrate that our VEWE platform behaves similarly to realistic on board devices, thereby providing a complete and reliable platform for smartphone application development.**

***Index Terms*—OBD-II; emulation; Bluetooth; application development; performance evaluation.**

I. Introduction

The Intelligent Transportation Services (ITS) envisioned by the research community rely on efficient vehicular communications as the keystone upon which to build a plethora of services, including accident alerts, traffic congestion updates, infotainment, etc. Nevertheless, the slow process of standardization and the adoption of new technologies by the vehicle manufacturers requires taking a pragmatic approach, seeking alternative solutions that allow to, at least, partially meet the goals of these future systems and services. In this scope, the widespread adoption of high-performance smartphones emerges as the candidate solution to make vehicles "smarter" if an efficient integration between both is achieved with low costs.

The On Board Diagnostics (OBD-II) [1] standard, introduced in 1994, has recently become an enabling technology for in-vehicle applications due to the appearance of Bluetooth OBD-II connectors [2]. These connectors enable a transparent connectivity between smartphones and the different Electronic Control Units (ECUs) available in the vehicle.

By combining smartphones with OBD-II devices, endless possibilities in terms of novel services and applications arise, including the detection of vehicle faults and interacting with the manufacturer to solve them, developing applications to automatically analyze driver behavior and suggest corrective actions when necessary, or providing georeferenced data about speed, CO2 emissions, or other relevant parameters.

Currently, it is already possible to find several smartphone-based applications that rely on OBD-II communications [3]. However, the development of these applications has a high cost since the developer has to deal with real ECUs from different manufacturers in order to test and debug the applications, usually requiring taking the vehicle for short test trips. To avoid this requirement, and to speed-up the development process, in this paper we propose VEWE[1], a Vehicle ECU Wireless Emulator that includes map-based mobility modeling, and a simulated engine ECU accessible through a wireless interface. The functionality provided by VEWE allows the developers to test their OBD-II based smartphone applications as if driving a real vehicle. To make sure that VEWE is able to adequately emulate an OBD-II connection, we perform a set of experiments to assess the performance of Bluetooth-based OBD-II connections on real vehicles, and then use them to validate the behavior of our platform. Experimental results in terms of end-to-end request capacity and per-request delay show that VEWE is able to accurately emulate an OBD-II connection, as intended.

This remainder of this paper is organized as follows: in section II we present some related works, evidencing how our work differs from previous ones. The VEWE solution is introduced in section III, and technical details are provided in section **??**. Experimental results are then presented in section IV. Finally, section V presents the main conclusions of this paper.

II. Related works

Currently we can find a broad range of smartphone applications able to communicate with a vehicle's ECU to provide enhanced services to drivers in the scope of the Intelligent Transportation Systems (ITS) area.

In a first group of solutions we can find simpler approaches where the goal is basically to provide the user with a graphical display of the different parameters obtained through the OBD-II interface. Examples of such applications include Torque [4] and the one proposed by Teng et al. [5].

[1] Available for download at: http://www.grc.upv.es/Software/vewe.html

A second group of solutions address road security and inter-vehicle communications. In this context, Laskowski et al. [6] proposed a rapid prototyping environment targeting emerging data-intensive telematic and control applications. Zaldivar et al. [7] proposed an Android-based application that monitors the vehicle through the On Board Diagnostics (OBD-II) interface, being able to detect accidents and send predefined notifications. DriveAssist [8] is a solution for Android smartphones that allows to visualize traffic information originating from both Vehicle-to-X (V2X) communications and central traffic services (CTSs).

A third group of solutions focuses on monitoring relevant data in vehicles. Tahat et al. [9] proposed such a solution, which monitors the vehicle's fuel consumption and other vital electromechanical parameters; SMaRTCaR [10] goes a step further by interacting with sensors on board and in the vehicle surroundings.

A fourth group of solutions uses OBD-II retrieved parameters to indirectly monitor the driver behavior. Proposals like Artemisa [11] and Driving coach [12] assess the driver's driving style from the standpoint of fuel. More recently, Meseguer et al. proposed DrivingStyles [13], a solution able to determine the type of road where the driver is circulating, as well as his driving habits.

The aforementioned research works highlight the different areas where there is a clear interest in developing smartphone applications that interact with vehicles through the OBD-II interface. Nevertheless, developers should perform frequent test driving to evaluate their proposals, which is expensive and time consuming. The platform presented in this paper attempts to simplify and accelerate development by allowing to jointly emulate OBD-II communications and geopositioning. To the best of our knowledge, this is the first emulation tool of vehicular mobility over real maps offering (i) access to the engine ECU through OBD-II along with (ii) access to geolocation information, all data being provided in a consistent and coherent manner to resemble real-world behavior.

III. VEWE: Vehicle ECU Wireless Emulator

VEWE is a solution developed to simulate vehicular behavior. It follows a client-server paradigm where an Android-based client application queries a Java-based server to retrieve a vehicle's ECU parameters through a Bluetooth OBD-II interface (see figure 1). By using VEWE, the developers of OBD-II based smartphone applications are able to test them as if they were moving in a real vehicle, thereby speeding up the development time and lowering costs.

The VEWE platform is composed of the VEWE server, running on a standard PC, and the GPSEmulator, an optional application running on an Android client.

The VEWE Server is the main Java application, being responsible for simulating the behavior of a moving vehicle, its engine Electronic Control Unit (ECU), and its location. For this endeavor it relies on different components: (1) Vehicle Mobility Simulator, (2) Map Manager, (3) Engine ECU, and (4) an OBD-II Emulator.

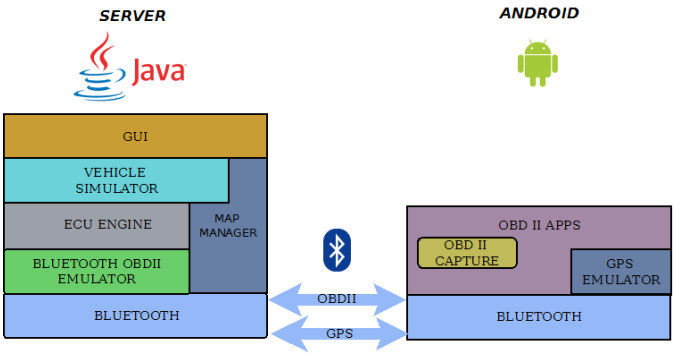


Figure 1: VEWE architecture.

Concerning the Vehicle Mobility Simulator (1), it is responsible for simulating a real vehicle, and dynamically determining the value of all relevant parameters. The Map Manager (2) retrieves the actual street map from the OpenStreetMap platform [14], and dynamically updates the vehicle's position on screen. The Engine ECU (3), it is a component used to store the value of all relevant PIDs being simulated, which can then be retrieved through the OBD-II interface. Finally, the Bluetooth OBD-II interface emulator (4) creates and maintains the OBD-II Bluetooth connection.

Besides the VEWE server application described above, our solution also includes an Android component (GPSEmulator) used to emulate a (fake) location on Android systems. Together, the *VEWE server* and the *GPSEmulator* provide a complete test system for Android-based mobile vehicular applications.

Below we provide details about the vehicle simulator, the emulation of GPS coordinates, and the OBD-II parameter database.

A. The vehicle simulator

The vehicle simulator is a key element in the VEWE Server, being responsible for modeling vehicle mobility according to the user input, the vehicle characteristics, and the terrain profile. To that purpose it relies on the acceleration calculator module which, by taking into account the different forces affecting the vehicle's mobility - traction, friction, aerodynamic and gravity - determines the acceleration value (see figure 2). Based on the acceleration value, it then updates the vehicle position on the map, as well as the desired parameters on the engine's ECU (e.g. speed, RPM, gear).

Real-time updating of the vehicle position on the map provides the user with a feeling of control over the simulated vehicle using the joystick available on screen. Notice that, by modifying the vehicle characteristics element, different types of vehicles can be modeled.

B. Emulating GPS coordinates

The Map Manager component obtains updated vehicle coordinates from the vehicle simulator described above. In particular, vehicle positions are updated by taking as reference: (i) the current vehicle position, (ii) the orientation, and (iii)

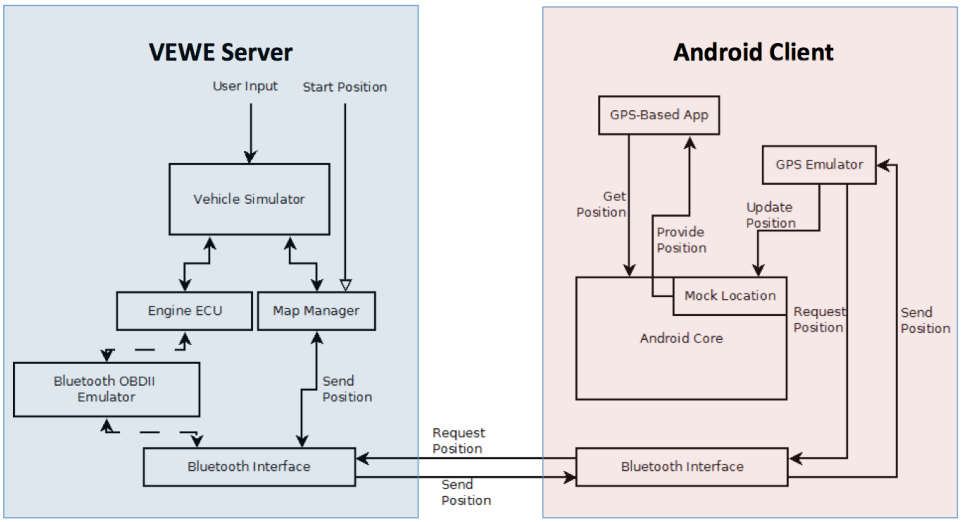


Figure 3: Emulated position update procedure.

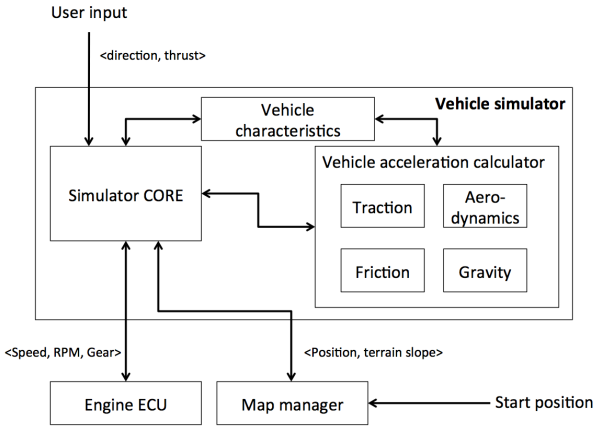


Figure 2: Overview of the vehicle simulator.

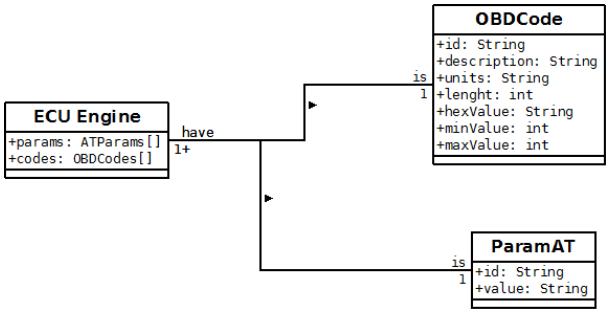


Figure 4: Structure of the OBD-II parameter database.

the distance traversed; the latter is calculated by combining the vehicle's speed with the inter-sample times.

The Map Manager component is able to provide geopositioning information to external applications through a Bluetooth channel. In our framework, we developed an Android application whose only purpose is to generate fake positions based on the information retrieved from the Map Manager via Bluetooth (see Figure 3). This application, called *GPS Emulator*, uses the *mock location* functionality provided in the Android API to introduce the fake locations into the system; to avoid interferences, it also cancels all other sources of localization information, including GPS, WiFi, or cellular-based positioning. This way, all running applications that register for localization services will receive the mock locations generated by the VEWE Server Map Manager component.

C. Engine ECU and emulated OBD-II interface

Providing an emulated OBD-II interface accessible via Bluetooth is one of the main goals of the VEWE platform. Similarly to the system deployed in a real vehicle, our OBD-II interface allows retrieving parameters from the different ECUs made available by the vehicle. In our platform, we model the engine ECU, since it is the one managing all the OBD-II PIDs that can be retrieved by users.

In order to retrieve the actual PID values, the OBD-II layer communicates with the Engine ECU module. Figure 4 shows the structure of the OBD-II database used, where the Engine ECU handles a data structure capable of storing all the vehicle parameters (OBD-II Codes) as well as the AT parameters.

As shown in Figure 5, the OBD-II emulator is able to maintain a serial connection with a client, and adequately process AT commands and PID requests, as would occur when using real on board OBD-II devices. The requested PIDs are then retrieved from the engine ECU. The OBD-II interface emulator converts the retrieved values to the appropriate format for delivery through the serial port.

To avoid that application developers experience a performance not comparable to real-life OBD-II devices, it is important for VEWE to resemble the behavior of OBD-II communications in vehicles. Notice than, when using a real OBD-II adapter, two different delays are involved: (i) the delay associated to serial mode communications using the ELM327

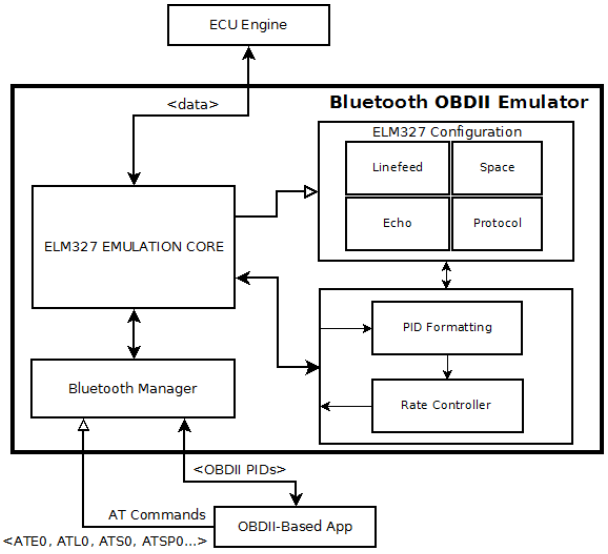


Figure 5: Structure of Bluetooth OBD-II emulator module.

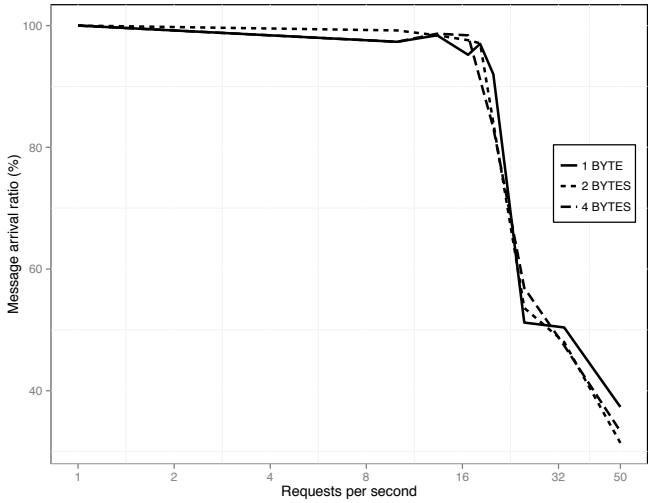


Figure 6: Analysis of the capacity of the OBD-II channel when requesting PIDs of different lengths.

interface [15], and (ii) the delay associated to Bluetooth communications using RFCOMM, which provides serial port emulation over the Bluetooth channel. Since the Bluetooth RFCOMM connection is common to both VEWE and real OBD-II devices, only the serial communication between the OBD-II connector and the engine ECU remains, and should be modeled independently. To this purpose, VEWE uses a rate controller to constrain the message reply rate (see Figure 5). This rate controller connects to the ELM327 emulation core, and it is able to model serial port transmission at 115.2 Kbit/s by adapting the Bluetooth output data rate to this transmission speed.

Equation 1 describes the total delay (D) introduced by the serial port communications:

$$D_{total} = D_{tx} + D_{prop} \qquad (s) \tag{1}$$

where D_{tx} refers to the transmission delay, and D_{prop} refers to the propagation delay. In a vehicular environment the propagation delay will be insignificant compared to the transmission delay ($D_{prop} \ll D_{tx}$). Thus, we focused the modeling efforts on the latter.

Equation 2 shows how to calculate the delay associated to the transmission process:

$$D_{tx} = \frac{L}{R_{SP}} \qquad (s) \tag{2}$$

where L is the total message size (in bits) according to the protocol used, including headers and trailers, and R_{SP} is the transmission rate over the serial port.

In the scope of VEWE, we adopted a byte-counting leaky bucket to shape traffic in order to have a regular flow of messages. In particular, we adopt the *Leaky Bucket Algorithm as a Queue* approach described by Tanenbaum [16], where the clock tick rate is adjusted to match the serial port speed, and the bucket size is set to 64 KB, large enough to prevent message dropping in most cases. As shown in section IV, this strategy allows to accurately shape traffic, providing a message rate quite similar to that of a real system.

IV. Performance Assessment and Platform Validation

In this section we evaluate the performance achieved when retrieving information through a Bluetooth-enabled OBD-II interface. We characterize the system behavior when attempting to make ECU requests from the smartphone, comparing the results achieved with a real vehicle to those provided by the VEWE platform. The vehicle used for the real tests was a Kia Sportage 1.6 Gdi (2011), which has a CAN bus type 2.0A [17].

We start our analysis by determining the behavior of the system in terms of channel capacity when varying the size of the message carried in the data field of CAN frames.

Figure 6 shows the message arrival ratio when increasing the number of PID requests per second. We find that, when more than 18 PID requests per second are issued, the system collapses, and so the number of replies is drastically reduced. This is expected as the limited channel capacity prevents delivering more replies, meaning that in real systems it makes no sense to increase the number of requests beyond this threshold. We also show that the size of the different PIDs requested does not significantly affect performance, being the behavior similar for all sizes. Notice that the message header and trailer used in CAN frames reduces the overall impact of the message carried in the data field.

Figure 7 shows the saturation behavior when comparing the results obtained using a real OBD-II device against those provided by VEWE. It is noticeable how the OBD-II communications performance causes the reply rate to drop beyond the upper limit supported by the serial channel (18 requests per second). Beyond that upper limit, the number of reply messages per second becomes slightly erratic, actually

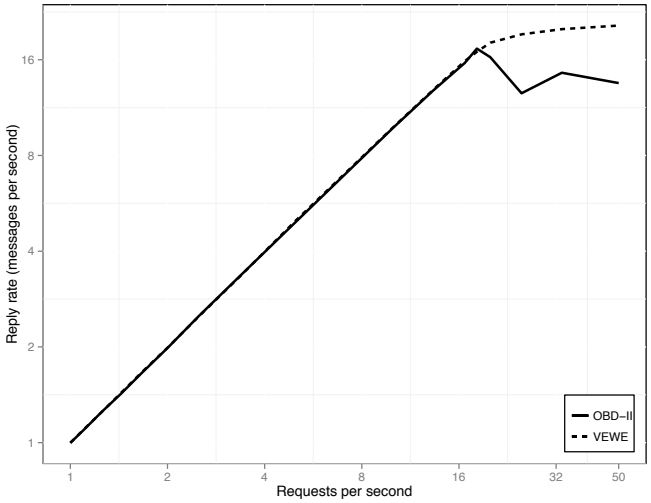


Figure 7: Determining the request saturation behavior when comparing real OBD-II communications against the VEWE platform.

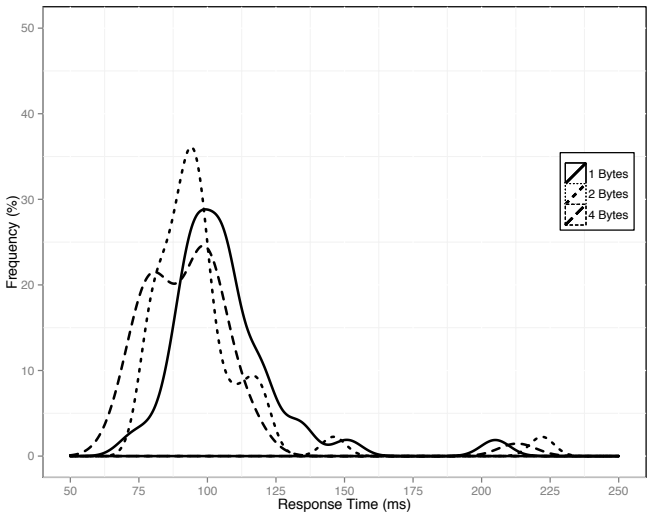
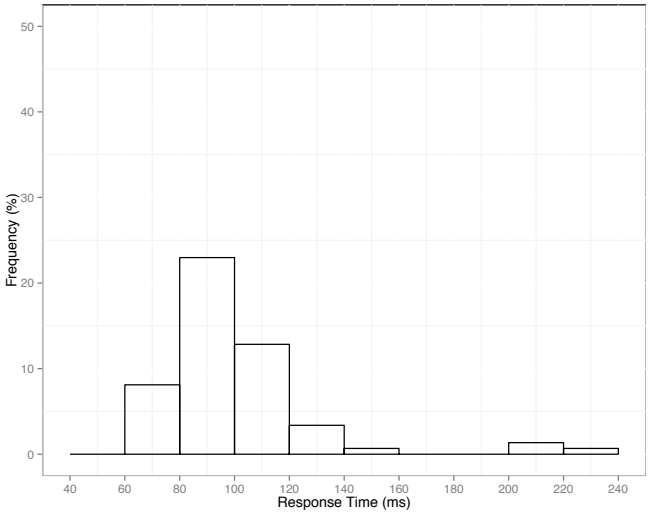
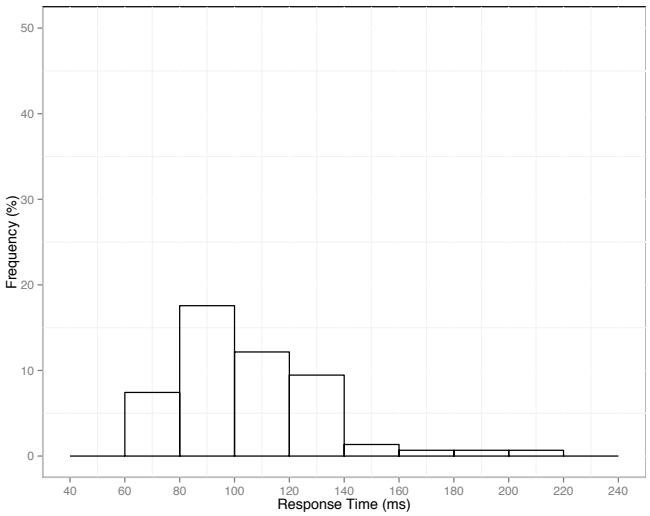


Figure 8: Probability distribution of OBD-II response times when testing with different message sizes.

going below the maximum value supported. In the case of VEWE, even though the station emulating communications has enough CPU power to reply to the requests, and despite the Bluetooth channel offers more than enough bandwidth, we found that the leaky bucket traffic shaping approach described in section III-C works effectively, avoiding that the number of replies provided by the server increases above 18/20 messages per second, thus resembling a real system. Notice that the performance of VEWE does not collapse when the number of incoming requests per second becomes high since the computing power of the desktop PC used to run VEWE is more than enough to process such requests, contrarily to the ELM327 chip located inside the OBD-II connector, which has quite limited resources.



(a) OBD-II



(b) VEWE

Figure 9: Frequency histogram for the response time.

Focusing on the response times, Figure 8 shows the behavior when requesting PIDs with different sizes. In particular, we retrieve PIDs corresponding to sizes 1, 2 and 4 bytes, thus covering all typical possibilities for standard PIDs. Results show that, in general, the range of values is usually the same. Typical values are usually greater than 50ms, and lower than 165ms. Only in rare occasions do delay values grow beyond 200ms, never surpassing 240ms. Overall, Figure 8 highlights that, although some differences are detected, the user can expect a same delay independently of the size of the PID requested. This occurs because most of the end-to-end delay jitter is introduced by the Bluetooth channel, which is prone to introduce more time variability when processing messages [18].

Finally, Figure 9 shows the average response time histogram. Notice that one of the main goals of the OBD-II

emulation system introduced by VEWE was making the delay experienced by smartphone applications to be similar to that achieved using real OBD-II devices.

We can see that the VEWE platform provides a similar behavior, having a similar range of values, the same mode and a very similar skew compared to those values achieved using a real vehicle. Such results also confirm the adequacy of using a leaky bucket of the byte-counting type to delay messages, thus accounting for the transmission delay associated to the ELM327 serial connection.

From the application development perspective, the OBD-II communication characteristics of VEWE resemble that of a real vehicle in terms of both (i) maximum number of PID requests served per second and (ii) delay distribution associated to response times, thereby being validated for its target purposes.

V. Conclusions and future work

Future ITS systems are expected to enable novel services and solutions in vehicular environments, providing drivers with updated information about weather, road conditions, traffic jams and accidents, among others. Similarly, vehicles also contribute to global solutions by sensing the driver, the road, and the different vehicle parameters, which can be delivered to a central server for information fusion/data mining purposes.

Nowadays, a growing number of researchers is making novel contributions to this field by integrating smartphones with vehicles, which typically relies on wireless OBD-II interfaces to act as a bridge between both. However, developing such solutions is costly and time consuming, typically requiring test drives to properly debug and tune the functionality of the applications being developed. In this paper we provide an efficient solution to this problem by introducing VEWE, a novel platform able to emulate a Bluetooth-based OBD-II connection, along with the GPS coordinates of the vehicle. VEWE allows the developer to control the mobility of the vehicle, as well as all OBD-II parameters required during the course of application development.

One of our key requirements was for VEWE to resemble the behavior on board OBD-II devices. To achieve this, we have modeled the serial port communications provided by the ELM327 integrated circuit using a byte-counting leaky bucket to shape traffic. Experimental results have validated our approach, showing that VEWE behaves quite similarly to real OBD-II devices in terms of channel capacity and delay introduced.

Since currently there is no similar tool available for the research community, the proposed solution is expected to boost application development in the ITS area, reducing the development effort and costs.

As future work we will make a wider analysis in order to determine the performance differences associated to different vehicle manufacturers and different OBD-II interfaces, translating those differences to our VEWE platform.

Acknowledgments

This work was partially supported by the *Ministerio de Ciencia e Innovación*, Spain, under Grant TIN2011-27543-C03-01.

References

[1] International Organization for Standardization, "ISO 14230-1:1999: Road vehicles, Diagnostic systems, Keyword Protocol 2000," 1999.

[2] Elm Electronics - Circuits for the Hobbyist, "Obd to rs232 interpreter," 2013.

[3] S. Martinez, J. Meseguer, J. Zaldivar, C. Calafate, J.-C. Cano, P. Manzoni, M. Fogue, and F. Martinez, "Smartphones as the keystone for leveraging the diffusion of ITS applications," in *9th ITS European Congress*, 2013.

[4] Ian Hawkins, "Torque: OBD2 Performance and Diagnostics for your Vehicle. Available: http://torque-bhp.com/," 2014.

[5] H.-F. Teng, M.-J. Wang, and C.-M. Lin, "An implementation of android-based mobile virtual instrument for telematics applications," in *Innovations in Bio-inspired Computing and Applications (IBICA), 2nd International Conference on*, pp. 306–308, 2011.

[6] M. Laskowski, J. Allen, M. R. Friesen, R. McLeod, and K. Ferens, "Rapid prototyping vehicle-to-infrastructure applications using the androidtm development platform," in *Intelligent Transport Systems Telecommunications,(ITST),2009 9th International Conference on*, pp. 273–278, Oct 2009.

[7] J. Zaldivar, C. Calafate, J.-C. Cano, and P. Manzoni, "Providing accident detection in vehicular networks through OBD-II devices and Android-based smartphones," in *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, pp. 813–819, 2011.

[8] S. Diewald, A. Möller, L. Roalter, and M. Kranz, "Driveassist - a v2x-based driver assistance system for android," in *Mensch & Computer Workshopband*, pp. 373–380, 2012.

[9] A. Tahat, A. Said, F. Jaouni, and W. Qadamani, "Android-based universal vehicle diagnostic and tracking system," in *Consumer Electronics (ISCE), IEEE 16th International Symposium on*, pp. 137–143, 2012.

[10] C. Campolo, A. Iera, A. Molinaro, S. Paratore, and G. Ruggeri, "Smartcar: An integrated smartphone-based platform to support traffic management applications," in *Vehicular Traffic Management for Smart Cities (VTM), 2012 First International Workshop on*, pp. 1–6, Nov 2012.

[11] V. Corcoba Magana and M. Munoz-Organero, "Artemisa: An eco-driving assistant for android os," in *Consumer Electronics - Berlin (ICCE-Berlin), 2011 IEEE International Conference on*, pp. 211–215, Sept 2011.

[12] R. Araujo, A. Igreja, R. de Castro, and R. Araujo, "Driving coach: A smartphone application to evaluate driving efficient patterns," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pp. 1005–1010, June 2012.

[13] J. E. Meseguer, C. T. Calafate, J. C. Cano, and P. Manzoni, "Drivingstyles: A smartphone application to assess driver behavior," in *Computers and Communications (ISCC), 2013 IEEE Symposium on*, pp. 535–540, July 2013.

[14] "The OpenStreetMap Project. Available at: http://www.openstreetmap.org/," 2014.

[15] ELM Electronics, "ELM327 Product Information. Available at: http://www.elmelectronics.com/obdic.html#ELM327," 2013.

[16] Andrew S. Tanenbau, *Computer Networks, Fourth Edition*. Prentice Hall PTR, 2003.

[17] ISO/TC 22/SC 3: Electrical and electronic equipment, "ISO 11898-1:2003: Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling," 2003.

[18] J.-C. Cano, J.-M. Cano, C. Calafate, E. Gonzalez, and P. Manzoni, "Evaluation of the trade-off between power consumption and performance in bluetooth based systems," in *Sensor Technologies and Applications, 2007. SensorComm 2007. International Conference on*, pp. 313–318, Oct 2007.