Smartphones as the keystone for leveraging the diffusion of ITS applications

Sergio Martínez Tornell, Javier E. Meseguer, Jorge Zaldivar,
Carlos T. Calafate, Juan-Carlos Cano and Pietro Manzoni
Universitat Politècnica de València
Camino de Vera, s/n, 46022 Valencia, Spain
sermarto@upv.es, jmesegue@upvnet.upv.es, koke.zaldivar@gmail.com, {calafate, jucano, pmanzoni}@disca.upv.es
Manuel Fogue, Francisco J. Martinez
University of Zaragoza, Spain
{m.fogue, f.martinez}@unizar.es

***Abstract*—The increasing activity in the Intelligent Transportation Systems (ITS) area faces a strong limitation: the slow pace at which the automotive industry is making cars "smarter".**

On the contrary, the smartphone industry is advancing quickly. Existing smartphones are equipped with multiple wireless interfaces and high computational power, being able to perform a wide variety of tasks. By combining smartphones with existing vehicles through an appropriate interface we are able to move closer to the smart vehicle paradigm, offering the user new functionalities and services when driving.

In this paper we review three different Android based applications, developed to evaluate the possibilities provided by the use of smartphones for ITS. Moreover we analyzed the behavior of the wireless channel and the GPS location service under different conditions to assess the feasibility of our proposal under the hardware constraints of the used devices.

I. Introduction

The integration of users, vehicles and cities requires a common agent that could act as the main conduit for information, services, and connectivity. The way the market is developing, smartphones could become this agent allowing the creation of an integrated environment. According to many analysts there are various reasons why the smartphone is key to telematics and therefore more than an interim connectivity play. The main reasons why are: (a) it's cheaper for OEMs to leverage smartphone development, (b) the car will not be the unique "vehicle", (c) smartphones contain your digital identity, and finally (d) we "love" our gadgets.

According to the European Automobile Manufacturers Association [3], the average age of the car fleet in Europe is 8.7 years, and 34.5% of the automobile fleet in the EU are older than 10 years. Taking these statistics into account, if manufacturers started to install specialized On Board Units (OBUs) in every car right now, in the best case, it would take more than 10 years to achieve a penetration rate of about 80%. In addition, experience demostrates that only luxury cars tend to incorporate these hi-tech devices as standard equipment. In our opinion, traffic management and safety-related applications cannot wait until then. Simultaneously, smartphones recently reached a 50% of penetration in developed countries, and this value is still growing. Smartphones are typically equipped with several network interfaces: WiFi, cellular network, and Bluetooth. From our point of view, smartphones offer an opportunity for developers and Vehicular Ad-Hoc Networks (VANETs), which can evolve from a pure ad-hoc network, with its known limitations, to a heterogeneous and more versatile network taking advantage of the possibilities offered by other wireless network interfaces. In addition, the On Board Diagnostics (OBD-II) [20] standard, available since 1994, has recently become an enabling technology for in-vehicle applications due to the appearance of Bluetooth OBD-II connectors [21]. These connectors enable a transparent connectivity between the mobile device and the vehicle's Electronic Control Unit (ECU).

The range of possibilities that arise when combining the car and the smartphone is endless, allowing, for example, diagnosing the car via mobile devices which assume the tasks that are typically performed by the On Board Unit (OBU) of the vehicle, or sending the collected data to a platform where diagnosis and vehicle maintenance can be done, detecting possible failures automatically.

In this article we describe three solutions that exploits the direct communication between smartphones through an ad-hoc network to provide a better driving experience when integrated together with a navigation software.

II. Related Work

In [11] its authors demonstrated how sensors available in smartphones can be used to prevent accidents in a proactive manner, through the analysis of the data collected by cameras, accelerometers and the Global Positioning System (GPS) system. Furthermore, there are several Android based applications [12] that try to make driving a better experience. The best example is Waze [13], an application that exploits the user collaboration through the cellular network. Another example of the possibilities that smartphones and their multi-interface schemes can offer to VANETs is Torque [14], an application that can connect to the OBD-II [15] interface of the vehicle to obtain real-time information about the current state of the engine. Regarding navigation devices, we have found different solutions that aim at improving the user experience

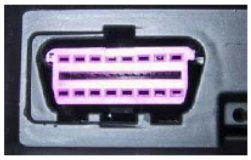
while driving and moving efficiently. Recently, navigation devices manufacturers, like TomTom [9] or Garmin [10], have presented their own *real time assisted routing* based on online updates provided via a cellular network. According to information made available at their websites, they combine data obtained from the local authorities with data obtained from each individual device. Their devices do not have any wireless interface for ad-hoc communication, so they miss the opportunity of real collaboration with neighboring vehicles. Moreover, the updates via cellular network may present a high delay.

III. The OBD-II standard

The On-board Diagnostic (OBD) standards were developed in the USA to detect car engine problems that can provoke an increase of the gas emission levels beyond acceptable limits. To achieve this purpose, the system is constantly monitoring the different elements related to gas emissions, including engine management functions, being a powerful tool to diagnose problems on vehicles electrical systems. When a failure is detected, the system must store it in memory so that technicians may analyze it later on. The first OBD standard, known as OBD-I, defined only a few parameters to monitor, and did not establish a specific emission level for vehicles. Thus, failures resulted in just a visual warning to the driver and the storage of the error. The second generation of OBD, known as OBD-II (Figure 1), standardizes different elements such as the connector used for diagnostic, the electrical signaling protocols, and the message format. Additionally, it defines a list of parameters that can be monitored, assigning a code to each parameter. A detailed list of DTCs (Diagnostic Trouble Codes) is also defined in the standard. Several operating modes are defined by the OBD-II standard to allow for an easier interaction with the system, and defining the desired functionality. Most automobile manufacturers have introduced additional operation modes that are specific to their vehicles, thus offering a full control of the available functionality. The European version of the OBD-II standard, known as EOBD, is mandatory for all gasoline and diesel vehicles since 2001 and 2003, respectively. Despite it introduces small improvements, EOBD strongly resembles OBD-II, sharing the same connectors and interfaces.

Although the physical interface is well defined, the communications protocol varies depending on the manufacturer. The available protocols are: (i) SAE J1850 PWM (Pulse- Width Modulation), (ii) SAE J1850 VPW (Variable Pulse Width), (iii) ISO 9141-2, (iv) ISO 14230 KWP2000 (Keyword Protocol 2000), and (v) ISO 15765 CAN; these protocols present significant differences between them in terms of the electrical pin assignments. Notice that most vehicles implement only one of these protocols. For instance, Chrysler uses the ISO 9141-2 protocol, General Motors uses SAE J1850 VPW, and Ford uses SAE J1850 PWM.

Diagnostic Trouble Codes were standardized in document ISO 15031-6, and allows engine technicians to easily determine why a vehicle is malfunctioning using generic scanners.



(a) Female connector

(b) Male connector.

Fig. 1: Example of a) an in-vehicle OBD-II female connector, and b) a Bluetooth-enabled OBD-II device with male connector..

The proposed format assigns alphanumeric codes to the different causes of failure, although extensions to the standard are allowed to support manufacturer-specific failures.

The OBD system was designed to offer a flexible communications system. Message delivery among different devices requires defining the type of message to be delivered, along with the transmitter and the receiver devices. The adoption of different message priorities is also supported in order to make sure that critical information is processed first. However, depending on the protocol used, the format of this message may vary slightly. Notice that both frame formats allow up to 7 data bytes, and they include a checksum field in order to detect any transmission errors.

IV. The eMDR Traffic Alert application

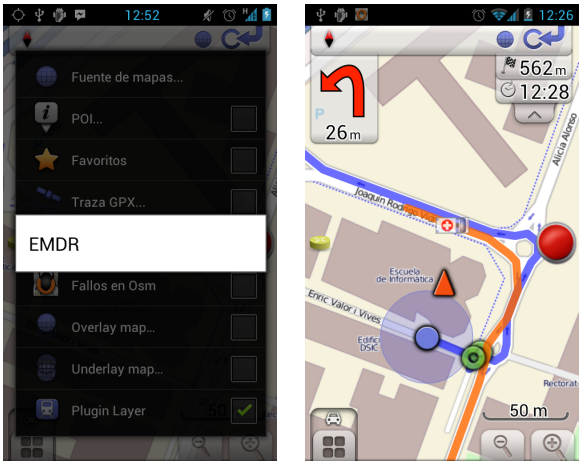
The eMDR Traffic Alert application exploits the direct communication between smartphones through an ad-hoc network to provide a better driving experience when integrated together with a navigation software. The application implements our protocol “enhanced Message Dissemination based on Roadmaps” (eMDR) [4] to inform its users about the presence of an ambulance close to their route; allowing the drivers to act in consequence, anticipating its actions.

We integrated our VANET application architecture with an existing open source navigation software. Our main requisites were: (i) to have a free map data source to avoid royalties issues, (ii) to have an offline route calculation system, and (iii) to present an easily expandable structure. With these premises in mind, after scouting the Android market, we chose OsmAnd [17], a navigation software that uses maps and route layout information from OpenStreetMaps [18]. The map rendering process in OsmAnd is composed by different layers that are rendered sequentially. Therefore the different applications can be programmed as a special layer that not only draws new data on the map, but also communicates with the dedicated protocol threads which use the socket API to communicate with other vehicles. We have developed a class, called ‘GeoHelper”, to simplify the use of geographical data

provided by the integrated GPS, and to deal with map issues related with the navigation service. Summarizing, the OsmAnd map layer calls the *draw()* of our new VANET App Layer, which obtains the required info from the different protocol threads, eMDR in this case. On the other side, the protocol threads communicates with our GeoHelper class, which runs under its own thread, to obtain geographical info.

eMDR uses location and street map information to facilitate an efficient dissemination of warning messages in VANETs, avoiding the well known broadcast storm problem and taking into account the effects of buildings to avoid wasting transmissions. In this section we include a short description of the behavior of eMDR, a more detailed description of eMDR can be found in [4].

1) User Interface: As explained above, our application is implemented as a new layer that adds information obtained from the VANET to the map view in OsmAnd. Before implementing our application we have also created a new interface called *GeoPluginLayer* that offers a common structure to child classes. This allows us to quickly implement different protocols reusing common code. Figure 2a shows the interface through which users will select the desired protocol, in our case eMDR. Figure 2b shows our "Warning Ambulance Application" running. We can see two idle neighbors represented by green circles, and a neighbor in alarm mode, represented by an ambulance icon; the orange line is the ambulance's route, and the blue line represents the vehicle programmed route. Both the red button on the right and the route-shaped button on the left, are used for testing purposes. The red button activates the alarm mode and, if present, broadcasts the programed route; it is supposed to be available only to authorized devices, such as ambulances, police-cars, etc. The other one is used to select between three forwarding modes: (i) normal forwarding, *i.e.* following eMDR rules, (ii) unconditional forwarding, *i.e.* every alarm message is rebroadcasted, and (iii) forwarding disabled, *i.e.* no alarm message is rebroadcasted.



(a) Layer selection screen. (b) Application screenshot

Fig. 2: Different application screenshots.

2) eMDR Implementation: Following a divide and conquer paradigm we have structured our implementation of enhanced Message Dissemination based on Roadmaps (eMDR) in three different classes: an upper class, called EmdrPluginLayer, that handles *onDraw()* calls from the map and all user interface related events (buttons). It makes use of two privates threads that implement the eMDR protocol, namely: EmdrPacketGenerator and EmdrProtocol. The former is in charge of packet generation, while the latter is in charge of receiving, processing, and forwarding the messages received from other nodes following eMDR rules. Our eMDR protocol will send a beacon every second, or warning messages instead when in alarm mode. In addition, if a route is programmed, it will be added to the warnings as an array of points.

3) GeoHelper Class: To simplify the management of geographical data we have developed a class called "GeoHelper". This class collects and processes data from files and from the services provided by both OsmAnd and the Android operating system (*i.e.* GPS, routing service, etc), offering different methods to our application.

The most important method provided by our GeoHelper class is the method *getCurrentLocation()*, which returns, an estimated current location based on the last two updates provided by the GPS interface and the direction of the current road. The difference between the last two updates is used to estimate the speed vector of the vehicle, if a *getLocation()* call occurs between two consecutive updates; the current position is estimated using the estimated speed vector and the last known location. In the case of *getLocationOnStreet()* calls, the estimated location is restricted to be on the closest road, and the speed vector also lays on the current road.

Concerning the *findRoute(location)*, we found that the time required by OsmAnd to calculate a route between two locations was in the order of tens of seconds, and that this value is strongly dependent on the number of possible routes. In our opinion, these issues impede the usage of some routing protocols that calculate the shortest route for every sent packet, as in [19].

V. The DrivingStyles application

This application applies data mining techniques to generate a classification of the driving styles of users based on the analysis of their mobility traces. Such classification is generated taking into consideration the characteristics of each route, such as whether it is urban, suburban or highway.

To achieve the overall objective, the system is structured around the following four elements:

1) An application for Android based smartphones. Using an OBD-II Bluetooth interface, the application collects information such as speed, acceleration, engine revolutions per minute, throttle position, and the vehicle's geographic position, being the latter obtained by the GPS interface. After gathering the information, the user uploads the route data to the remote data center for analysis.
2) A data center with a web interface able to collect large data sets sent by different users concurrently, and to

Fig. 3: Snapshots of the home screen and the application running on a vehicle.

graphically display a summary of the most relevant results. Our solution is based on open source software tools such as Apache, PHP and Joomla.

3) A neural network, which must be trained using the most representative route traces in order to correctly identify, for each path segment, the driving style of the user, as well as identify the segment profile: urban, suburban or highway. We adopted the backpropagation algorithm [31], which has proven to provide good results in classification problems such as the one associated to this project.
4) Integration of the tuned neural networks in the data center platform. The goal is to use neural networks to dynamically and automatically analyze user data, allowing users to find out their profiles as a driver, and thus promoting a less aggressive and more ecological driving.

A. The Android application

The Android application (see figure 3) is a key element of the system, proving connectivity to the vehicle and to the DrivingStyles web platform. Currently, it can be downloaded for free from the DrivingStyles website http://www.drivingstyles.info, or from Google Play.

In order to adjust the functionality of our Android application to the user requirements, several configuration options must be defined related to user creation, connection options**,** GPS activation, and sensor sampling**.** The available functionalities are: User creation, Connection options, GPS Activation, and Sensor sampling.

The main module of our application launches the background processes responsible for capturing data sent by the OBD-II and the GPS interfaces, as well as the phone's accelerometer.

Besides showing the sensors we are monitoring, we can perform several parallel actions without affecting the data capture. Possible options are:

- Start and stop data capture of the Electronic Control Unit (ECU), the mobile's accelerometer, and the GPS.
- List the routes captured by the app.
- Show the current position of the vehicle on the map, as well as the detailed route followed whenever an Internet connection is available.
- Real time visualization of the speed, rpm, and acceleration in a time window of 10 seconds.

1) Route Upload Module: The route upload module is in charge of sending the users' traces to the website data center for further analysis. This module can be accessed either from the historic stored routes, or immediately after stopping the data capture. The information screen displays the header information of the selected route such as: date of the captured data, start time, finish time, and maximum speed.

This module includes a graphical interface for showing the routes on a map, as well as the collected statistics. Additionally, it also includes communication facilities for uploading the collected routes to the data center.

Finally, the file received in the DrivingStyles server is stored in the corresponding user directory, and also generates a record in the database for every sample submitted for analysis.

2) Map Module and Graphical Information: These modules are in charge of displaying information relevant to the user in the most convenient manner. The graphs can be displayed in real-time or by selecting data from previously stored paths. Depending on the device model, the user can also zoom in and out to display all or part of the graph using the device's touch screen.

The charts that appear on screen are the acceleration, the speed, and the revolutions per minute (rpm). We have chosen these three parameters because they are the most relevant ones, and are also the ones we have selected for training our neural network (see figure 4).

The map module allows displaying the GPS position on the map. GPS coordinates are drawn using the Google Maps APIs. A green car icon indicates the beginning of the route, and a red icon shows the current position of the vehicle. The path is shown by using different colors depending on the vehicle's speed (see figure 4).

B. The DrivingStyles Web interface

The second main component of our architecture corresponds to the data center and its web interface. For this endeavor we have selected open source software such as Apache HTTP, and Joomla as the content management system (CMS). We have used a CMS, combined with the use of a resource wrapper, which allows detach our system from the presentation layer, thus focusing on the problem of driving styles characterization. The URL of this module is http://www.drivingstyles.info.

Basically the data center provides functionality to work with User, Routes and Statistics.

1) User: Once the user is logged in, he is asked to record a number of important data, especially for future data mining studies. The most relevant items are sex, age, and other details concerning the vehicle used: car manufacturer, model, fuel type, and the theoretical 0-100 acceleration (important to normalize the user behavior in our study). Finally, a third block allows drivers to indicate what they feel about their own behavior behind the wheel, i.e., whether they perceive themselves as aggressive, moderate, or quiet drivers.

Fig. 4: Snapshots of the acceleration, speed, rpm parameters and map module.

2) Routes: In the Routes' section, the users can access all the routes they have uploaded. The first grid shows routes that are in the database including the name, date, starting time, ending time, samples sent, total time, average speed, and kilometers travelled. Below the grid, the selected route is shown in a map. The path varies its color depending on the speed of the car (see figure 5).

3) Statistics: In addition, other graphs, not represented here, will show the results that the neural network sends back, including the driving styles and the route characteristics.

In the next section we provide more information about the neural network we proposed for characterizing driver styles.

C. Neural Networks based data analysis

In our project, we face a classification problem: starting from some input data, which in our case are the speed, the acceleration, and the revolutions per minute of the engine (rpm), we intend to obtain as output the type of road and the driving style. After studying the different types of algorithms available, we decide to choose backpropagation [31] since this

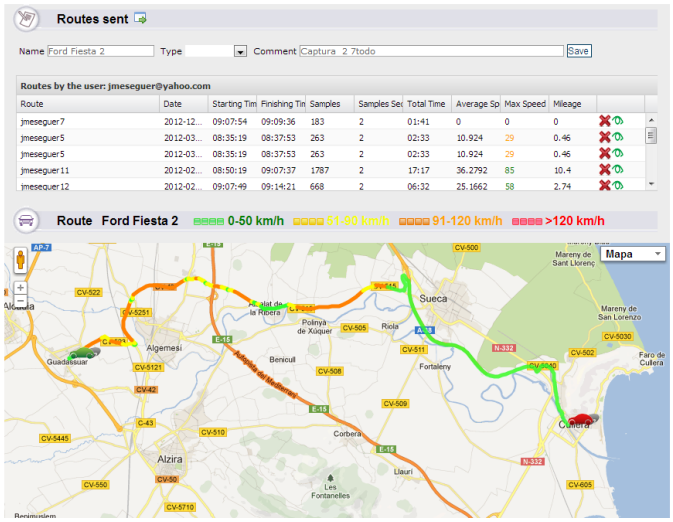


Fig. 5: Snapshot of a route map.

kind of algorithm provides very good results in classification problems.

A data preprocessing stage is selected from all the possible input variables of the neural network that we have considered initially. From all the possible data, we keep a subset of these variables. In practice, this subset is not the minimum one; instead, it is a compromise between a manageable number (not too large) of variables and an acceptable network performance. In this project, after considering the many variables that can be obtained from the Electronic Control Unit (ECU), we have chosen to train the neural network using: a) the mean and standard deviation of speed, b) the vehicle acceleration, and c) the rpm. In all vehicles used for testing, these variables were easily obtained, while other variables, such as the position of the throttle, which would provide important information for the neural network training, finally had to be rejected because not all ECU manufacturers provide such information. The data input of each parameter is normalized between 0 and 1; this normalization should take into consideration the whole range of possible values.

With the neural network implemented, every time a route or route segment is selected, the system automatically returns the type of road, and the associated driving style. The overall behavior of each user is also obtained by evaluating all the routes sent by the user.

VI. The BBDROID application

The final application is called BBDROID and is an application that monitors the vehicle through an On Board Diagnostics (OBD-II) interface, being able to detect accidents. Our proposed application estimates the G force experienced by the passengers in case of a frontal collision, which is used together with airbag triggers to detect accidents. The application reacts to positive detection by sending details about the accident through either e-mail or SMS to pre-defined destinations, immediately followed by an automatic phone call

to the emergency services. Experimental results using a real vehicle show that the application is able to react to accident events in less than 3 seconds, a very low time, validating the feasibility of smartphone based solutions for improving safety on the road.

Initially the smartphone connects to an OBD-II device via Bluetooth to retrieve data from the vehicle's bus. The information gained, together with data from other sources (e.g. GPS system) is packed and sent to an emergency services database or to other third parties defined by the user if an accident is detected. This procedure is followed by an automatic call to an operator, which will send an ambulance or other rescue services to the accident location. The application also offers general purpose information to the driver, including gas levels, detection of failures in mechanical elements, extensive engine feedback data, etc.

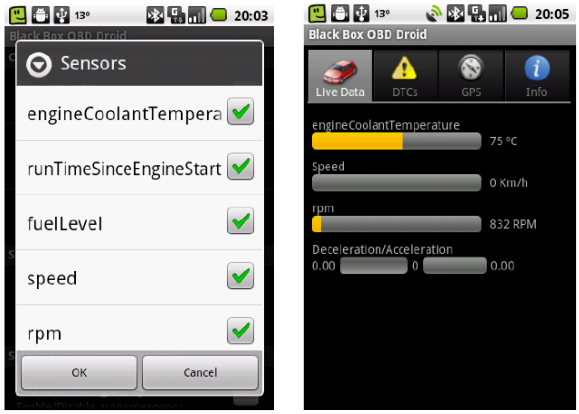
Android-based smartphones typically include different wireless interfaces, such as Bluetooth, Wifi, GPS and 3G, making them ideal for our purposes. In particular, our solution will rely on the Bluetooth technology to establish a data link between the smartphone and a Bluetooth-enabled OBDII interface. This approach removes the need for any sort of cable, thus making it more robust against car crashes.

Since a data communications channel between the smartphone and an online server is required, it can be established using either the Wifi or the 3G interface. Typically, mobile telephony services, such as voice calls and SMS generation, can also be used. For instance, the system can be configured to send an SMS to our family, establish a voice channel with the emergency services, and send detailed accident information, including impact speed and current GPS position, to a special purpose server. This way, all the entities involved in the process may obtain all the information considered relevant.

The basic startup process works as follows: first the application checks whether Bluetooth is enabled, returning an error otherwise; in a second step it attempts to contact the OBD-II device defined. In case it is found, the different protocols supported are checked to determine which one is valid for the current vehicle. Finally, if bidirectional communication is established successfully, the application will start the system monitoring process. Since the most novel component of the proposed application is automatic detection and reaction upon accidents, we now address how accidents are handled.

The accident detection process, basically works as follows: if either the airbag is triggered or the deceleration detected is greater than 5 Gs, we consider that an accident occurred. Notice that severe accidents are usually associated with forces above 20 Gs, and death conditions take place in impacts above 50 Gs. To avoid false alarm situations, the user is given 1 minute to abort the actions that follow. In case the user does not abort, the accident warning procedure acts by retrieving GPS and accident details, which are automatically sent through email and/or SMS to a pre-specified address. Afterward, an automatic emergency call to the emergency services is made. In case no data channel is available, the call takes place immediately.

In addition to the accident detection system, the application offers additional information to users. The first step is to establish communication with the vehicle using a Bluetooth connection to the OBD-II device. Once communication is established, the user may select the sensors considered of interest for monitoring, as shown in Figure 6 (left). The monitored information is then shown in the application main window, see Figure 6 (right), and refreshed periodically to provide the user with a feedback about the vehicle in terms of performance, problems detected, and other information of interest.



(a) Route type. (b) Route behavior

Fig. 6: Snapshots of the proposed application: parameter selection menu (left) and real-time parameter visualization (right)..

VII. Smartphone hardware validation

We have designed a set of experiments to evaluate the following smartphone hardware performance parameters: (i) message reception probability when in Line Of Sight (LOS), (ii) message reception probability when nodes are in different streets, and (iii) GPS updates inter-arrival time. All the experiments were performed in a real environment with the eMDR Traffic Alert application; vehicles were parked in streets with a typical traffic flow. Since all mobiles were inside vehicles, the transmissions were also affected by different issues related to adverse signal effects caused by the structure of the vehicle.

1) Message reception probability when in LOS: In this experiment we placed two of the handsets in different cars; then, using our application, sent a burst of 200 warning messages and counted the number of messages successfully received. We have executed two different experiments to evaluate this metric. The first one was executed in a high traffic density environment, and the second one was in a low traffic density environment. To achieve comparable results each experiment was repeated four times, therefore we have represented the mean and the confidence interval at 95%. Results, represented in figure 7, shows that, as expected, the reception probability decreases when the distance increases. Comparing both graphs, we can appreciate that the presence or the absence of interferences due to traffic density can highly influence the performance of VANET's application in

smartphones, reducing the communication rate from 80 m to merely 40 m, and increasing the variability of the results. Our experiments have shown that the eMDR threshold distance (*i.e. the minimum distance at which a retransmission is worth*), is optimal when the reception probability at such distance is of 40%. Therefore, we have chosen 60 m as the threshold distance for our eMDR protocol.

2) Message reception probability when in N-LOS: In this experiment the cars where located in perpendicular streets. One of them was located 25 meters away from the intersection, and the second vehicle was moving away from the intersection. As expected, the moving car stopped receiving messages as soon as it moved a few meters away from the intersection. With these results in mind, we decided that the threshold distance under eMDR to consider that a vehicle is "near to an intersection" would be configured to 10 meters or less. Also, experiments have shown that the best parameter to detect if a vehicle is close to an intersection is the detection of neighbors from different streets. This method avoids problems related to the street layout representation format of OsmAnd.

3) GPS updates inter-arrival time: Another important issue when checking the feasibility of our solution is the freshness of the GPS data in smartphones. In the rest of our experiments we collected around 12000 measurements for this metric. By analyzing them, we found that the mean inter-arrival time for GPS updates was 1.07 s, while the maximum value was of 15 s, and only in 1 % of the total test its value was different from 1.0s. Although we have configured the GPS interface to notify our application about location changes as soon as possible, the minimum time between updates that the system was able to provide was of 1.0 s. If we consider that, a vehicle with a speed of 25 m/s, a maximum acceleration of 0.8 m/s^2, and a maximum deceleration of 4.5 m/s^2 can typically travel between 21.40 m and 25.40 m per second, our position estimation system, which assumes a constant direction and speed during the inter-update time, will introduce a maximum error of 3.6 m due to mobility. We believe that this value is small enough to be used in VANETs.

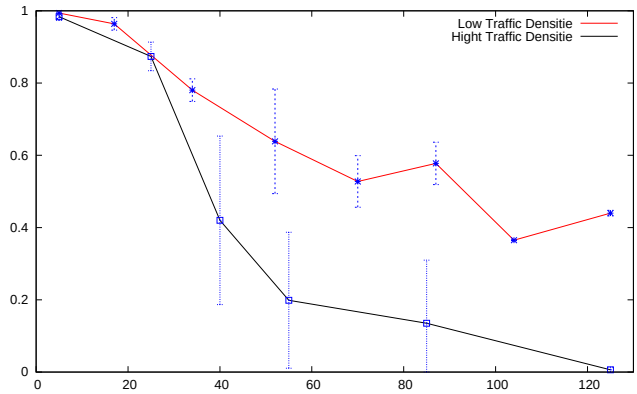


Fig. 7: Rx Probability vs Distance

VIII. Conclusions and Future Work

In this paper we presented an implementation of three ITS services based on the use of smartphones. Our objective was to validate the idea that the integration of users, vehicles and cities requires a common agent that could act as the main conduit for information, services, and connectivity. The way the market is developing, smartphones could become this agent allowing the creation of an integrated environment.

In the near future we will explore the possibilities that emerge when evolving from pure 3G or WiFi connections to ad-hoc (VANETs) network or even to heterogeneous and more versatile network, like the Delay Tolerant Networks (DTN), by taking particle retention to the combined use of the various available network interfaces.

Although security issues are outside of the scope of this paper, spoofing attacks, where a non-authorized entity sends warning packets, could be avoided using an asymmetric keys scheme to sign every warning packet sent by authorized entities. We will carefully study this problem and its different solutions in future work.

Acknowledgments

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

References

[1] "European commission's website for smartcities," http://eu-smartcities.eu/mobility_transport, last visit Jul 2012.
[2] "Esafety website," http://ec.europa.eu/information_society/activities/esafety/index_en.htm, last visit Jul 2012.
[3] "European automobile manufacturers association," http://www.acea.be/news/news_detail/vehicles_in_use/, last visit Jul 2012.
[4] F. J. Martinez, M. Fogue, M. Coll, J.-C. Cano, C. T. Calafate, and P. Manzoni, "Evaluating the impact of a novel warning message dissemination scheme for vanets using real city maps," in *Proceedings of the 9th IFIP TC 6 international conference on Networking*, ser. NETWORKING'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 265–276.
[5] L.-C. Tung and M. Gerla, "An efficient road-based directional broadcast protocol for urban vanets," in *Vehicular Networking Conference (VNC), 2010 IEEE*, dec. 2010, pp. 9 –16.
[6] R. Frank, E. Giordano, P. Cataldi, and M. Gerla, "TrafRoute: A different approach to routing in vehicular networks," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2010 IEEE 6th International Conference on*. IEEE, Oct. 2010, pp. 521–528.
[7] J. Eriksson, H. Balakrishnan, and S. Madden, "Cabernet: vehicular content delivery using wifi," in *Proceedings of the 14th ACM international conference on Mobile computing and networking*, ser. MobiCom '08. New York, NY, USA: ACM, 2008, pp. 199–210. [Online]. Available: http://doi.acm.org/10.1145/1409944.1409968
[8] M. Gerla, J.-T. Weng, E. Giordano, and G. Pau, "Vehicular testbeds; validating models and protocols before large scale deployment," in *Computing, Networking and Communications (ICNC), 2012 International Conference on*, 30 2012-feb. 2 2012, pp. 665 –669.
[9] "Tomtom," http://www.tomtom.com/, last visit Jul 2012.
[10] "Garmin," http://www.garmin.com/, last visit Jul 2012.
[11] A. Campbell and T. Choudhury, "From smart to cognitive phones," *Pervasive Computing, IEEE*, vol. 11, no. 3, pp. 7 –11, march 2012.
[12] "Google play," https://play.google.com/store, last visit Jul 2012.
[13] "Waze," http://www.waze.com/, last visit Jul 2012.
[14] "Torque," http://torque-bhp.com/, last visit Jul 2012.

[15] International Organization for Standardization, “ISO 15765: Road vehicles, Diagnostics on Controller Area Networks (CAN),” 2004.

[16] J. Gozalvez, “First google’s android phone launched [mobile radio],” *Vehicular Technology Magazine, IEEE*, vol. 3, no. 4, pp. 3 –69, december 2008.

[17] “OsmAnd website,” http://www.osmand.net, last visit Jul 2012.

[18] “OpenStreetMap website,” http://www.openstreetmap.org, October 2012.

[19] X. Wang and C. Song, “Distributed Real-Time Data Traffic Statistics Assisted Routing Protocol for Vehicular Networks,” in *Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on*. IEEE, Dec. 2010, pp. 863–867.

[20] International Organization for Standardization, ”:1999: Road vehicles, Diagnostic systems, Keyword Protocol 2000”, 1999.

[21] ELM327DS. OBD to RS232 Interpreter. Elm Electronics – Circuits for the Hobbyist.

[22] U. Hernandez, A. Perallos, N. Sainz, and I. Angulo, ”Vehicle on board platform: Communications test and prototyping,” in Intelligent Vehicles Symposium (IV), 2010 IEEE, pp. 967 –972, 2010.

[23] Co Eco-Driving: Pilot Evaluation of Driving Behavior Changes among U.S. Drivers.

[24] E.Erikcsson. ”Independent driving pattern factors and their influence on fuel-use and exhaust emission factors” Transportation Research Part D: Transport, 2001 – Elsevier, 325-345J.

[25] Johansson, H., Gustafsson, P., Henke, M., Rosengren, M., 2003. Impact of EcoDriving on emissions. International Scientific Symposium on Transport and Air Pollution, Avignon, France.

[26] M.-C. Chen, J.-L. Chen, and T.-W. Chang, ”Android/OSGi-based vehicular network management system,” Elsevier Computer Communications, vol. 34, no. 2, pp. 169 – 183, 2011

[27] Fogue, M. Garrido, P., Martinez, F.J., Cano, J., Calafate, C.T., Manzoni, P. Automatic Accident Detection: Assistance Through Communication Technologies and Vehicles.Vehicular Technology Magazine, IEEE, Sept. 2012.

[28] Zaldivar, J., Calafate, C.T., Cano, J.C., Manzoni, P., Providing accident detection in vehicular networks through OBD-II devices and Android-based smartphones. Local Computer Networks (LCN), 2011 IEEE 36th Conference 4-7 Oct. 2011.

[29] JavaNNS, Java Neural Network Simulator, User Manual, Version 1.1 Igor Fischer, Fabian Hennecke, Christian Bannes, Andreas Zell.

[30] Haykin, S. (1994), Neural Networks: A Comprehen-sive Foundation, NY: Macmillan, p. 2.

[31] Hecht-Nielsen, R., Theory of the backpropagation neural network - Neural Networks, 1989. IJCNN., International Joint Conference.