Messiah: An ITS drive safety application

Subhadeep Patra, Sergio Martinez, Carlos T. Calafate,
Juan-Carlos Cano and Pietro Manzoni [1]

Abstract—
This article describes a novel safety application based on the open source navigation software OsmAnd, which runs on the Android platform. The application offers vehicles with "smart navigation", and maintains a network of the vehicles that use our application. The process of network creation and maintenance is important as our application enables vehicles to communicate with one another to exchange useful information. The main function of the application is to inform vehicles of relevant vehicles approaching, termed as "administrative vehicles" in this article, and include ambulances, police cars and fire brigades. Based on the received information, our application notifies the driver, who can now take navigation decisions based on it. While developing the application, problems were found when attempting to create an Ad-hoc network. A solution to the problem of managing the Ad-hoc network has been proposed and is under development.

Keywords—
ITS, VANET, Mobile Application, Android, Navigation, Software, Ad-Hoc Network.

I. Introduction

INTELLIGENT Transportation Systems(ITS) are advanced applications that makes use of vehicular and infrastructured networks to provide innovative services related to both traffic and, mobility management, and that interface with other models of transport [1]. Some of the examples of this type of applications include automatic emergency calling to a helpline number when an accident occurs, monitoring traffic do detect traffic rule violations [2] and defining variable speed limits for a particular road depending on the congestion [3].

Many software applications and testbeds have been developed keeping ITS and VANETs in mind. In [4], the authors developed an Android/OSGi platform which integrates the Open Gateway Service Initiative Vehicle Expert Group (OSGi/VEG) into an Android platform to generate a vehicular Android/OSGi platform. The features provided by the platform includes remote management, rich class-sharing, proprietary vehicular applications, security policies, easy management of the Application Programming Interface (APIs), and an open environment. Whipple et al. [5] developed an Android Public Safety Application which uses the GPS feature to find out the location of the vehicle, and it uses the Google Maps API to determine the location of nearby schools. It alerts the driver when surpassing the speed limit within a school zone. CarSafe [6], is an application developed for Android phones; it uses the dual camera of smartphones and other embedded sensors to detect drowsy and distracted drivers. In particular, it uses computer vision and machine learning algorithms to detect the condition of the driver using the front camera while tracking road conditions using the back camera. Cabernet [7] is a system that takes advantage of open 802.11 access points encountered opportunistically during travel for delivering data to and from vehicles. Another example is CVet [8], a VANET testbed presented by researchers from UCLA. The CVet testbed was deployed over vehicles belonging to the UCLA car fleet.

The aforementioned works highlight the many applications that are being developed for ITS and VANETs. Navigation applications are one such group of very popular, interesting and useful applications. Our goal is to build an application for Android [9] devices that, along with displaying maps and location information, could warn the user of important vehicles approaching (for example, an ambulance), so that the driver may take decisions based on this information. For this purpose, the OSM Automated Navigation Directions (OsmAnd) [10] application, a popular open source navigation solution was used and modified for our work. OsmAnd is a map and navigation application with support for both online and offline use. It relies on OpenStreetMap(OSM) [11] [12] for map rendering and displaying. It is an application rich in features, and it may be used by cars, bicycles or pedestrians. Our application works as a plugin for the OsmAnd application that enables vehicles to communicate, create a network, exchange important information among the nodes, and alert the drivers about important nearby vehicles.

The rest of the paper is organized as follows: Section II provides a general overview of the different features of the developed application. The implementation details of the developed application are provided in section III. Section IV describes the testing of the the application prototype, the problems found during application development, and its possible solution. Finally, Section V concludes the paper.

II. Features of the Messiah Application

The application developed has been named Messiah. The Messiah application, upon installation, allows the user to be a part of a "drive safety" network. If the user chooses to be a part of this network, the software allows the user to select whether it is an administrative vehicle like the police or an ambulance, or a vehicle in need of help (SOS vehicle). Based on this input, the software has three modes of operation:

- Administrative Mode.
- Civil Mode.
- SOS Mode.

[1]Grupo GRC, DISCA, Universitat Politècnica de València, e-mail: subpat@doctor.upv.es, sermarto@upv.es, {calafate, jucano, pmanzoni}@disca.upv.es

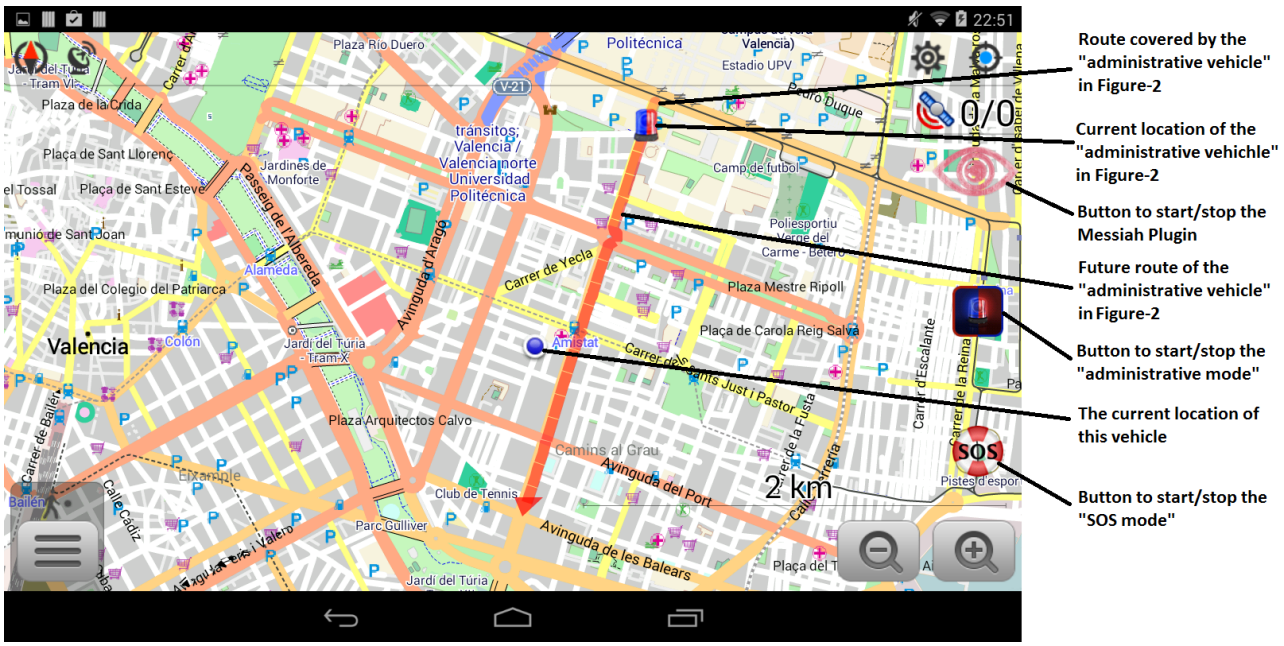

Fig. 1. The Device working in "Civil Mode"

When choosing to participate in the network without selecting the "SOS" or "Administrative" mode, the vehicle is assumed to be a civilian vehicle by default, and so the software starts working in the "Civil mode". In this mode, it only receives or forwards data to and from its neighboring nodes. Apart from this, it also displays on-screen any received information, as shown in Fig. 1.

Fig. 2. The Device working in "Administrative Mode"

Users can also choose their vehicle as an "Administrative vehicle". In the future, we intend to make this option available to only authorized personnels by implementing security checks. Upon selecting this option, the "Administrative mode" is turned on. In this mode, the nodes forward packets sent from other nodes and, in addition, also broadcasts generated packets containing mainly of their current location, their future route and the destination. The application shown is Fig. 2 is working in the "Administrative mode", which forwards its location information, future route and destination. The node in Fig. 1 receives this information, forwards it, and displays the received information on screen.

The software has another mode, through which a vehicle in distress can send an SOS beacon, which upon receipt by nearby vehicles results on the display of a different icon on screen, thus alerting other vehicles that someone in need of help is close-by.

Also, some intelligent features have been included in the software, for instance, when a source of information is more than 1 kilometer away from the receiving node, the packet is dropped as the information received may be stale. That is, by the time the information reaches the destination, the sender may have moved to some other remote location. Again, if the future route is too long, the whole of the route information is not sent, but only a part of it. In Fig. 1, we can see that this particular node receives only a part of the route of the vehicle shown in Fig. 2, and it displays a broken line with an arrowhead to indicate the future route. The arrowhead indicates the possible direction in which the route will be updated in the future.

Since the network is ad-hoc in nature, nodes may join and leave the network at any time, and network disconnections may arise. All these are automatically detected by the Messiah application. Furthermore, the application continues to work in the background even if the user is currently using some other application, or has minimized the Messiah Application, this allows the user to be a part of the network without requiring of our application to be on the foreground.

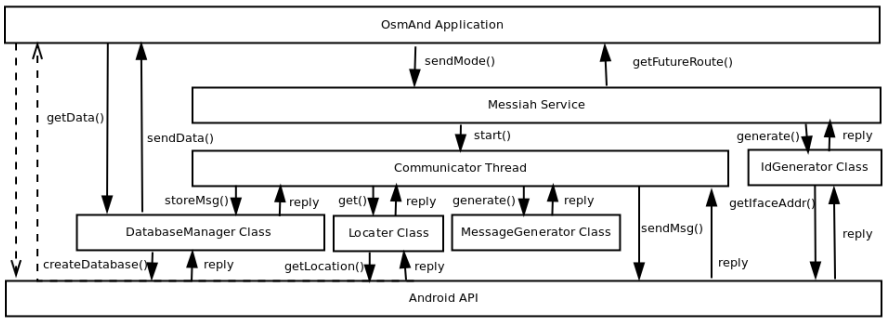

Fig. 3. Architecture of the Messiah Application

III. Implementation Details of the Messiah Application

Fig. 3 shows the different working blocks of the Messiah Application. "Messiah service" controls the functioning of our application and receives user input from the OsmAnd Application, for example, the mode of operation. The "Communicator Thread" is responsible for both sending and receiving of data. The "IdGenerator Class" is used to generate a unique id for a node. The "MessageGenerator Class" is used to generate the messages to be send. The "Locater Class" helps to track the current location of the device and supplies it when asked for. The "DatabaseManager Class" is used to store received message, acting as a cache for messages. It is also used to retrieve information later when needed.

ID	TimeStamp	Type	Location	Future Route	Destination

Fig. 4. The informations contained in the message used by Messiah Application

Now, let us look closely at the structure of messages received or sent within the Messiah network. The message contains a node-id, time-stamp, the type of message, the current location of the sender, its intended future route and the destination, as shown in Fig. 4. What is important to note here is the field "type" in a message. The icons and information displayed on screen is dependent on the "type" field of a received message. A message can be of three types: i) Administrative ii) SOS iii) Bye-Bye.

An "Administrative" and "SOS" message are basically identical, sent by an "Administrative vehicle" or a "SOS vehicle" respectively. Any important vehicle like ambulances, police cars or fire brigades are considered "Administrative vehicles", while "SOS vehicles" includes vehicles that have faced critical accidents and need assistance. A node sends these type of messages to notify other nodes in the network its updated current location. The only reason for separating the Administrative and SOS into two distinct types is so that they can be identified and different icons can be displayed for them. The "Bye-Bye" type message, on the other hand, is used by nodes that had been working as "Administrative/SOS" vehicle to notify other nodes within the network that it is either going to switch to "Civil mode" of operation or may leave the network altogether.

Let us look more closely at how the Messiah Application works. Fig. 5 demonstrates the operation of a normal vehicle working in the "Civil mode". In this mode, the node is to only receive information, display it and then forward it to its neighbors. Hence, the Communicator thread checks for incoming messages. If a message is received, it is first checked if it a valid message for the Messiah Network. On finding the received message to be valid, the distance of the source from the current node is checked. If the sender is more than 1 km away from this node, the database is checked for any entry from this particular sender and the entry if any is eliminated. After this, packet is dropped assuming that the information contained may be stale. On the other hand, if it is less than 1 Km and the message is found to be new, its type is checked. On receipt of a "Bye-bye" type message, the message is broadcasted and any entry related to the source node in the database is deleted. Otherwise, if the message is of other types, the information is stored in the database after adding a local timestamp. And then, the original message is broadcasted. The reason for adding a local timestamp before entering the information to the database is because a node may lose connectivity or network-partitioning may occur at any time. Just relying on bye-bye messages is not sufficient, it is necessary to detect if a node, especially if it working as an "Administrative/SOS" vehicle, leaves the network. And to address to these kind of situations, the database is checked for stale entries that have been there for more than 10 seconds without any updates from the sender. Such entries on being encountered, are eliminated.

A node functioning as an "Administrative/SOS" vehicle, receives and broadcasts the received packets in the same manner as a civil vehicle. They also generate and broadcast a message containing their id, a local timestamp, the type of message depicting the mode they are working in, their current location, intended future route and destination if any.

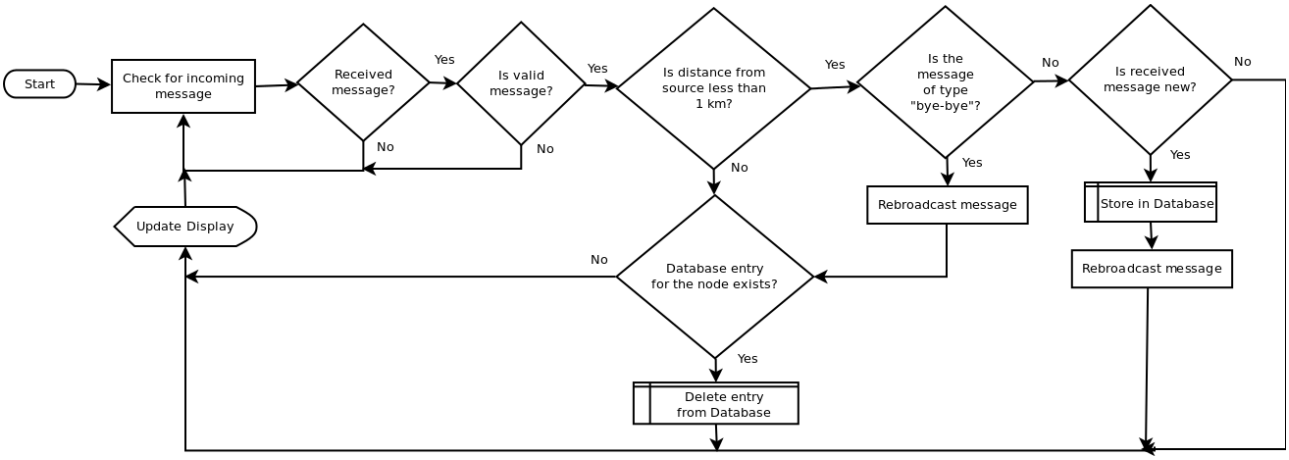

Fig. 5. Flow Chart for a vehicle working in "Civil mode"

IV. Prototype Evaluation

The first version of the application [13] has been tested within the campus of Universitat Politècnica de Valencia, with real cars forming a Ad-hoc network, using the Wifi of the mobile devices.

(a) A photograph from within the car

(b) A photograph of the device used

Fig. 6. The testing of the application within UPV Campus

For the evaluation, the devices used for testing our application had to be rooted to make them work in an environment where any sort of infrastructured network is absent, because Android does not yet support full Ad-hoc mode. Normally, the Android operating system does not provide the users with administrative rights thus the user lacks some access privileges. Thus, to gain total control of the wifi so that it may be used to form a ad-hoc network, it was necessary to root the device.

The Fig. 6 shows a successful testing of the application was performed within the UPV campus with real cars. A video demonstrating the same application being tested and its different functionalities is available at the following link: http://www.youtube.com/watch?v=Wh4cwmdvecM.

We would like to discuss an idea which we believe will address the problem of configuring the devices and rooting them. The idea is that we include cheap yet efficient devices to work as routers and handle Ad-hoc communication. Each vehicle participating in the network is supposed to contain one of these devices, and the mobile nodes present within this car would be connected to the device and forward their data to it. The device would then forward the received data to other devices in other vehicles that are present within its communication range. And in this way, the data would be forwarded until it reached the destination router or device, which would deliver the received data to the destination node in its local network.

It was decided to use a cheap computer known as Raspberry Pi [14] [15] to work as the router placed within the cars to help the network function in Ad-hoc mode. A Raspberry Pi is more or less of the same size as that of a credit card, working on Linux and supports different programming languages like Python, Java, C and Pearl.

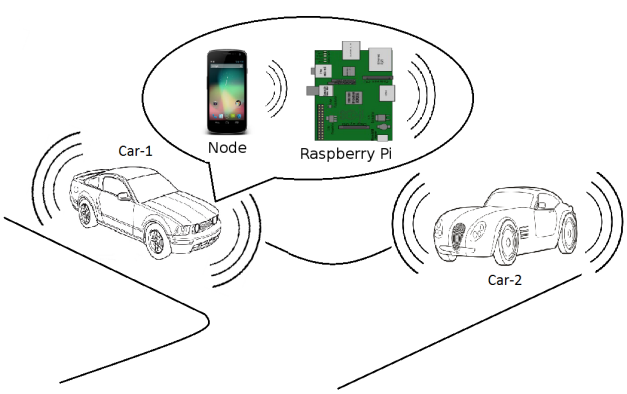

Fig. 7. The Structure of the Ad-Hoc Network

The Fig. 7 explains the idea in details, each of the cars participating in the network contains a Raspberry

Pi and the mobile nodes within the car is connected to this Raspberry Pi. Thus the smartphones within the car forms the local network of the Raspberry Pi which works as a router for them and forwards their data to the outside world.

V. Conclusions

Our application helps create a more safer traffic environment as it displays important vehicle like the police cars and ambulances on the navigation screen, thus warning drivers to drive with attention and take decisions based on the displayed information. It has also a mode when a vehicle in distress can call for help. In this mode, a special icon is visible on screen of the vehicles that are close to the "SOS vehicle", and hence they can rush to its help. The ITS network may be both infrastructureless or infrastructured. Problems were encountered related to configuration of the devices to make them them work when there is no infrastructured network available. Our application requires the devices to be rooted to support Ad-hoc network. Rooting of the device is not very encouraging from the user perspective as it may void the manufacturer warranty of the devices. An alternative and hassle-free solution to this problem has been proposed and is under implementation. On successful implementation, it would save us the trouble of configuring the android devices. This added functionality would not only help the Messiah Application with Ad-hoc networks, but also can be used as a testbed facilitating experimentation with different scenarios, application and protocols for the VANET networks.

Acknowledgments

This work was partially funded by (i) Svagata.eu, the Erasmus Mundus Programme, Action 2 (EMA2), of the European Commission. (ii) The *Ministerio de Economía y Competitividad*, Spain, under Grants TIN2011-27543-C03-01 and BES-2012-052673.

References

[1] The 2010 directives of the European Parliament and of the council on the framework for the development of ITS, *http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2010:207:0001:0013:EN:PDF*, accessed May 2014.

[2] Luo Qi, *"Research on Intelligent Transportation System Technologies and Applications,"*, Workshop on Power Electronics and Intelligent Transportation System, 2008. PEITS '08, pp.529,531, 2-3 Aug. 2008.

[3] Tackling congestion by making better use of England's motorways and trunk roads, *http://www.nao.org.uk/report/tackling-congestion-by-making-better-use-of-englands-motorways-and-trunk-roads/*, accessed April 2014.

[4] M. Chen, J. Chen, and T. Chang, *"Android/OSGi-based vehicular network management system,"*, Comput. Commun., pp. 1644-1649, 2011.

[5] J. Whipple, W. Arensman, and M.S. Boler, *"A public safety application of GPS-enabled smartphones and the android operating system,"*, Systems, Man and Cybernetics, pp. 2059-2061, 2009.

[6] C.W. You, N.D. Lane, F. Chen, R. Wang, Z. Chen, T.J. Bao, M. Montes-de-Oca, Y. Cheng, M. Lin, L. Torresani and A.T. Campbell, *"CarSafe App: Alerting Drowsy and Distracted Drivers Using Dual Cameras on Smartphones,"*, MobiSys '13, pp. 13-26, 2013.

[7] J. Eriksson, H. Balakrishnan, and S. Madden, *"Cabernet: vehicular content delivery using wifi,"*, MobiCom '08. New York, NY, USA: ACM, 2008, pp. 199-210.

[8] M. Gerla, J.-T. Weng, E. Giordano, and G. Pau, *"Vehicular testbeds- Validating models and protocols before large scale deployment,"*, Computing, Networking and Communications (ICNC), 2012 International Conference on, 30 2012-feb. 2 2012, pp. 665-669.

[9] J. Gozalvez, *"First Google's android phone launched [Mobile Radio],"*, IEEE Veh. Technol. Mag., vol. 3, no. 4, pp. 3-69, Dec. 2008.

[10] OsmAnd software hompage, *http://osmand.net/*, accessed October 2013.

[11] OSM homepage, *http://www.openstreetmap.org/*, accessed November 2013.

[12] M. Haklay and P. Weber, *"OpenStreetMap: User-Generated Street Maps,"*, Pervasive Computing, IEEE , vol.7, no.4, pp.12-18, Oct.-Dec. 2008.

[13] S. M. Tornell, C. T. Calafate, J. Cano, P. Manzoni, M. Fogue, and F. J. Martinez, *"Evaluating the feasibility of using smartphones for ITS safety applications,"*, in VTC Spring 2013 IEEE Vehicular Technology Conference, 2013.

[14] Raspberry Pi homepage, *http://www.raspberrypi.org/*, accessed April 2014.

[15] Cellan-Jones, Rory, "A £15 computer to inspire young programmers", *http://www.bbc.co.uk/blogs/legacy/thereporters/rorycellanjones/2011/05/a_15_computer_to_inspire_young.html*, accessed May 2014.