Hadiwardoyo SA, Patra S, Calafate CT *et al.* An intelligent transportation system application for smartphones based on vehicle position advertising and route sharing in vehicular ad-hoc networks. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 33(2): 249–262 Mar. 2018. DOI 10.1007/s11390-018-1817-4

An Intelligent Transportation System Application for Smartphones Based on Vehicle Position Advertising and Route Sharing in Vehicular Ad-Hoc Networks

Seilendria A. Hadiwardoyo, Subhadeep Patra, Carlos T. Calafate, Juan-Carlos Cano and Pietro Manzoni, *Senior Member*, *IEEE*

Deptartment of Computer Engineering, Universitat Politècnica de València, València 46022, Spain

E-mail: {seiha, subpat, calafate, jucano, pmanzoni}@disca.upv.es

Received July 4, 2017; revised January 21, 2018.

Abstract Alerting drivers about incoming emergency vehicles and their routes can greatly improve their travel time in congested cities, while reducing the risk of accidents due to distractions. This paper contributes to this goal by proposing Messiah, an Android application capable of informing regular vehicles about incoming emergency vehicles like ambulances, police cars and fire brigades. This is made possible by creating a network of vehicles capable of directly communicating between them. The user can, therefore, take driving decisions in a timely manner by considering incoming alerts. Using the support of our GRCBox hardware, the application can rely on vehicular ad-hoc network communications in the 5 GHz band, being V2V (vehicle-to-vehicle) communication provided through a combination of Android-based smartphone and our GRCBox device. The application was tested in three different scenarios with different levels of obstruction, showing that it is capable of providing alerts up to 300 meters, and notifying vehicles within less than one second.

Keywords intelligent transportation systems (ITS), vehicular ad-hoc network (VANET), mobile application, Android, navigation, ad-hoc network

1 Introduction

The intelligent transportation systems (ITS) paradigm includes vehicular networks that cover services like traffic, mobility management, and safe driving. Vehicular ad-hoc networks (VANETs) create a network of vehicles that communicate with one another. These networks allow transportation-related applications to be deployed[1]. When the surrounding infrastructures are included, not only Vehicle-to-Infrastructure (V2I) communications take place, but also Vehicle-to-Everything communication is enabled, commonly known as V2X. This combination of communications is an emerging technology while Internet access and smartphone-based applications are already present in the field of transportation[2].

Different applications have been proposed for ITS that enable VANET communications, including applications providing navigation in order to improve traffic efficiency. One example is the vehicular social networks that promote information sharing[3].

One of the main concerns in transportation is safety issues, which attract the attention of both research communities and the society. VANETs can be an efficient solution in the ITS field to solve the problem of safety issues and inefficiency in traffic motorways since they allow vehicles to communicate with one another, thus allowing the exchange of traffic-related information. With the goal of reducing the probability of traffic accidents, recent studies include the development of safety-related applications[4]. One example is the application that delivers emergency alerts. This application

Regular Paper

Special Section on Computer Networks and Distributed Computing

A preliminary version of the paper was published in the Proceedings of ICCCN 2017.

This work was partially supported by the "Ministerio de Economía y Competividad, Programa Estatal de Investigación, Desarollo e Innovación Orientada a los Retos de la Sociedad, Proyectos I+D+I 2014", Spain, under Grant Nos. TEC2014-52690-R and BES-2015-075988.

can be deployed when an accident occurs, providing an emergency call[5].

The goal of our work is reducing the probability of traffic accidents leveraging the advantage of inter-vehicular communication. This research work aims at designing, implementing, and deploying an Android-based application[6] providing the geographical information of relevant nearby vehicles displayed in maps.

Our proposed application works as a plugin for OsmAnd①. OsmAnd is an open source navigation application, which relies on OpenStreetMap (OSM)[7]. It can be run online or offline depending on the specific purpose, and can be easily modified. Maps are rendered and displayed to provide geographical information. Therefore, it can be used by cars, bicycles, and pedestrians. Hence, not only do users enjoy the features of a navigation application when driving, but also they can make use of a vehicular network.

The goal of our application is to notify vehicles about the presence of priority vehicles in their neighborhood, including ambulances, police cars, and fire brigades. The user is not only made aware of the position of such critical emergency vehicles but also about their tentative future routes so that drivers can take actions like allowing them to pass. This decision can be made beforehand, when the application notifies the users, instead of having to wait until the vehicle appears in the rear view mirror. The main idea behind the decision to implement this application is to minimize deaths due to a delayed emergency response upon the occurrence of accidents, or at crime scenes. Cities with dense traffic would largely benefit from our application due to two clear reasons. Firstly, the information dissemination in such scenarios would be much easier due to the large number of vehicles participating in the network. Secondly, the emergency vehicles would be able to take advantage of the streets that would normally be busier but this, of course, depends on how responsive and responsible the users of the application are.

Apart from the more critical applicability of our solution explained above, there is yet another very important feature that our proposed application presents here. This feature is the ability of users to ask for help if they are in situations where their life is at risk. Such situations may arise if a vehicle faces an accident, in which case users may notify vehicles around them to ask for assistance merely by clicking a button. In this way, others can rush to their aid even if the person in need of help is not directly within line of sight. Thus, by making use of wireless communications, our application is able to generate alerts that might result in saving lives.

This paper extends the results of our preliminary work presented in [8], where we have introduced the Messiah application. In this paper, Messiah's performance is assessed with more parameters and the traffic overhead is also assessed. The tests were done in three different types of street environments based on the level of obstructions. Moreover, detailed explanations of the application are presented in the following sections.

The organization of the paper is as follows. Section 2 refers to relevant related work. Section 3 includes the features provided by our application, followed by an overview of its architecture and mechanisms in Section 4. Section 5 explains the deployment of GRCBox to provide VANET communications. The application's performance is assessed in Section 6, where functionality testing of the application is covered. Security issues are presented in Section 7. Conclusion and future lines of research are then discussed in Section 8.

2 Related Work

Enhancing ITS and VANET technologies is one of the recent efforts made by the research community in order to promote safety, efficiency, and infotainment in the transportation area. One of the solutions proposed in this field is Cabernet[9], which uses open 802.11 access points to make data circulate between vehicles during travel. In UCLA (University of California, Los Angeles), researchers presented a VANET testbed[10]. Another effort includes developing an Android platform that integrates the Open Gateway Service Initiative Vehicle Expert Group (OSGi/VEG). This allows the platform to generate a vehicular Android/OSGi platform that provides an open environment, rich class-sharing, proprietary vehicular applications, and remote management of the application programming interface (APIs). With the ceaseless research efforts on involving smartphones with vehicle telematics, the driving experience is nowadays enriched with multiple features related to sensors, efficiency, and interfaces for the sake of safety and convenience[11].

Specifically, when integrating safety-related applications with Android-based smartphones, authors like Whipple *et al.*[12] proposed using an Android application in a school area to notify passing-by drivers about

①http://osmand.net/, Jan. 2018.

the speed limit. The GPS feature is used by this Android Public Safety application to discover the location of the vehicle, and it then uses the Google Maps API to define the location of nearby schools. When the driver surpasses the speed limit in a school zone, this application will send an alert. The development of other safety-related applications includes analyzing the behavior of the driver and notifying it. Drivingstyles[13] is one of the examples that allows warning the driver based on the behavior analyzed.

Other contributions combine visual information with sensor data to provide safe driving. One example is Carsafe[14], where the researchers developed an application that can detect the condition of the driver. Using computer vision technology, by capturing the visual information from the front camera and also the sensors that are embedded, the application analyzes the driver's state, determining whether he/she is drowsy or distracted. The use of visual information for the safety-related application is also proposed in EYES[15]. The authors of [15] used the smartphone cameras to capture the video that is streamed between drivers. This video sharing application is developed by pairing two driver smartphones so that one can be used for safe overtaking scenarios.

In addition, safety applications consider navigation data as well. There have been a lot of applications involving navigation services on smartphones for vehicles. In particular, these applications aim at improving user convenience. One example is a navigation system for smartphones that is not infrastructure-dependent[16], using maps that can be rendered offline. Another example involves the use of a projector for cyclists[17], to provide navigation services and minimize distractions while riding. In a car context, a few studies even proposed a user interface for displaying audio as an in-vehicle device that provides car navigation[18-19], where the interface is based on a smartphone. The authors of [20] built a Java API for these kinds of in-vehicle devices. In terms of providing efficient driving, a smartphone can also be used for navigation assistant considering an efficient route to avoid traffic congestion[21-22]. Navigation services can be combined with safety or emergency alerts. The application called iOnRoad② uses a smartphone for assistance when the users drive. This application provides information like advanced collision warnings, off-road alerts, headway distance alerts, and speeding alerts deployed in the smartphone. Another application[23] involves OBD devices as sensors to trace events and broadcast alerts using the Wi-Fi AP/client mode.

Navigation applications can make use of social networks to circulate the information between users. In [24] authors used voice tweets as part of vehicle social network groups to share driving experiences between drivers. These tweets are based on location and destination and can be used to calculate an optimum route based on collected tweets. This proposed application called NaviTweets is later expanded to benefit from a vehicular cloud, allowing to report and share traffic information[25]. Waze③ is the most popular driving assistant application. Not only does the application provide navigation features, but also it is based on social networks. Hence, the information is shared among users, allowing to offer traffic reports, traffic jam alerts, under construction road alerts, or notify the presence of police. However, these social network driving applications are heavily dependent on infrastructure support or the Internet.

Several efforts in the research community are focused on making special-purpose smartphone applications dedicated to safety and emergency alerts or notifications. Researchers in [26] have proposed an Android safety application based on ad-hoc networking. This application relies on Wi-Fi direct and incorporates Android devices to send automatic emergency data when the car is involved in an accident by relying on accelerometer data. Another effort in [27-28] includes analyzing the personal health of a driver and analyzing abnormal movements of the vehicle, which in turn will send an alert message via Facebook wall, email, or SMS. This emergency tracking system based on the Android platform will record not only the location of the user, but also the heart rate using a monitoring device. In an urban environment, sharp turns might result in accidents. The system proposed in [29] can send a warning alert to the driver to indicate the presence of sharp turn ahead (within 700 meters) so that the driver can slow down and take the turn carefully. In addition, this system can give a recommendation regarding places to request for help in the case of an emergency, as for instance a police station or a hospital within 9 000 meters.

Our research involves real vehicular environment settings that involve driving aids like navigation and emergency alerts. This work has been initially developed in [30], where the application that can notify

②iOnRoad. http://www.ionroad.com/, Jan. 2018.

③Waze. https://www.waze.com/, Jan. 2018.

other drivers in the network about its geographical location, as well as its future route, was proposed. Our proposed application now suits for a vehicular environment using the combination of Android smartphone and our GRCBox, featuring packet dissemination through ad-hoc network created by vehicles, and is infrastructure-independent. Furthermore, our application was assessed based on real experiments to evaluate its network performance.

3 Application Features

Our proposed application is called Messiah. With the use of this application, users can participate in a "drive safety" network. By being part of this network, users can choose vehicle modes provided by the application. The application distinguishes users by classifying them into three kinds of categories. The first is the administrative entity, acting as police car or ambulance. The second is the normal civilian car. The third is a car that is in need of help. With the input from the user, three modes of operations are provided in the application based on the mentioned categories:

- administrative mode;
- civil mode;
- SOS mode.

By default, the application starts with the "civil mode". This means that, whenever the user launches the application in the default mode (without choosing the "administrative" or the "SOS" mode), the application will assume that the user is in a standard vehicle. In "civil mode", the application will only receive and forward data using V2V (vehicle-to-vehicle) communications to and from neighboring nodes. Any received information is displayed on screen, as shown in Fig.1. In this figure, we can see that the on-display screen shows not only the location of the user (in this case, the civil mode) but also the current location of the administrative vehicle through a siren light icon. The future trajectory of the administrative vehicle is also drawn by a line with an arrow showing its direction on the street. As the screen shows two sirens and two routes, this indicates the presence of two administrative vehicles. Another icon displayed in the screen is the SOS icon. This indicates that within the area, a vehicle in need of help is present.

Whenever the user is in an emergency vehicle, no matter whether it is a police car or an ambulance, the administrative mode is selected. By working in this mode, the participating nodes will forward packets received from neighboring nodes, and in addition, will announce their presence in the network by using broadcast messages that contain their current geographical location. Additional information is included in the message such as the current route and destination. In the administrative mode's side, the on-display screen shows the route and the destination, as well as the nodes other than civilian nodes. In this case we can see one administrative node (including its route), and one SOS node (see Fig.2). The interface of the application display is

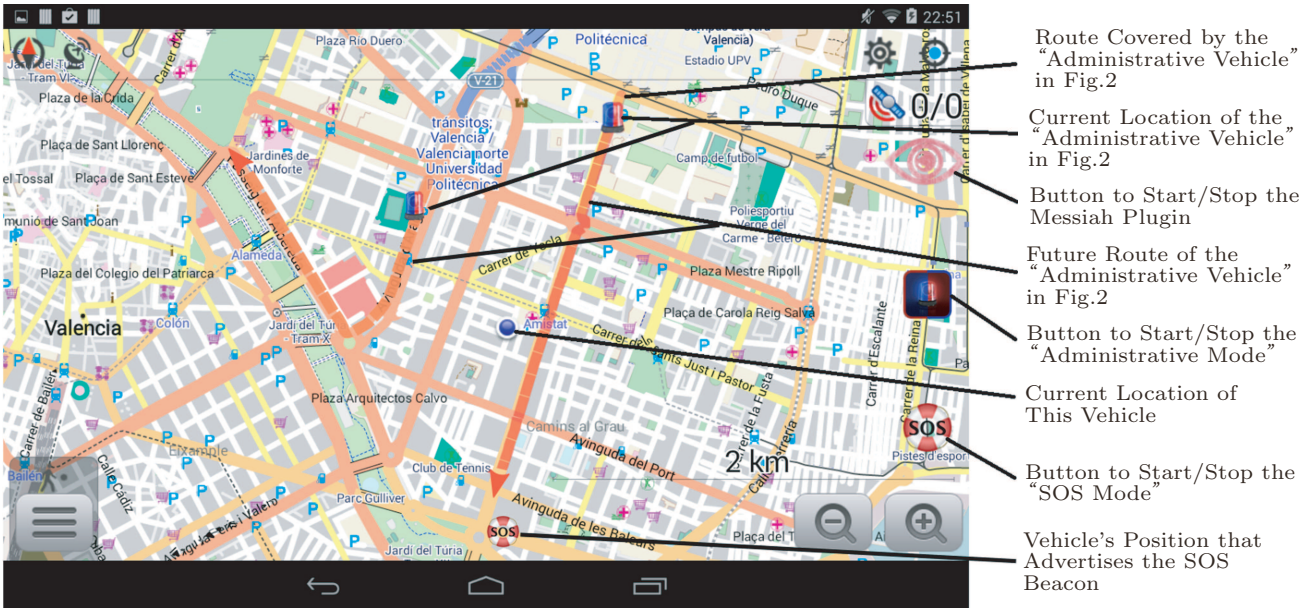


Fig.1. Device working in "civil mode".

similar in the "civil mode", with the difference that, in the administrative node, the display will not show other vehicles. The exception is the case when there is a vehicle in the SOS mode. In the background, the vehicle in the administrative mode will also receive and forward this information.

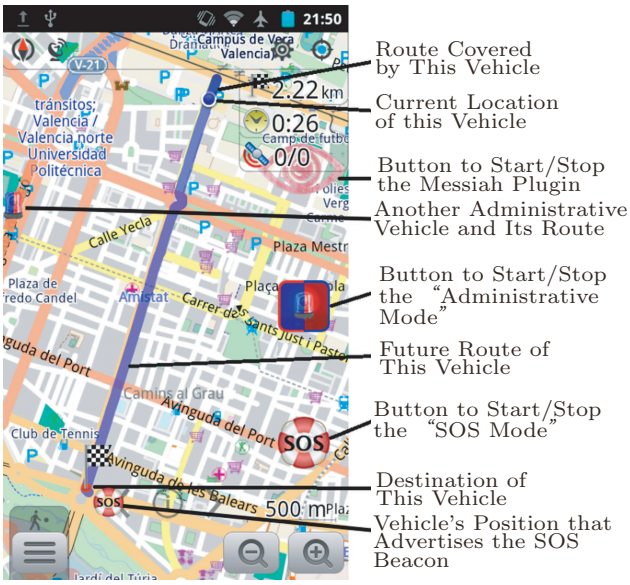


Fig.2. Device working in the "administrative mode".

Another feature of this application is the option of the SOS mode. This mode enables the users to ask for rescue. In the SOS mode, the user will generate SOS beacons to notify the neighboring vehicles. These beacons will allow the display of a special alert icon on the received vehicle's display. This method is used so that nearby vehicles can be warned that a problem is present, or that someone in need of help is close by.

A closer look at both figures (Fig.1 and Fig.2) shows that basic features of the default OsmAnd application are available. When the application is launched, the user can start the Messiah plugin by pressing the "eye" button that causes two other buttons to appear. In this way the user can choose whether to switch to the administrative or SOS mode. If the user does not press these additional buttons, the application will assume to be running in the civil mode.

The dissemination of messages from administrative and SOS vehicles is limited to a 1 km range. This means that the information is broadcasted to neighboring vehicles within 1 km. If the source of information is more than 1 kilometer away from the receiving node, the forwarding process stops. This is due to the reliability of the message. The packet is dropped as the information received may be stale: by the time the information reaches the receiving node, the source of the information may have moved to another remote location. The reason behind this limitation is to detect unwanted packets circulating in the network. We discard the use of time-based deadlines as a limitation since time is generated by each individual system, and thus we cannot have a uniform and synchronized time base for all devices. Hence, the distance parameter is considered as the most adequate option.

In the case of broadcasting future routes, the information is also purposely limited. The whole route information is not sent entirely to the receiver, but only a part of it. Hence, as we can see in Fig.1, this node's display only shows a part of the sender route, with the other part being symbolized as an arrow's head indicating the possible route.

On the other hand, at the source's side (Fig.2), it initially shows the destination location and a straight line without arrowhead. The line indicating the source's route will be updated when the source's node approaches the receiver's node, or if it is still receiving notifications.

With the eye button present in the display, the user can join and leave the network at any time by pushing the button. The button will enable the application to activate or deactivate the Messiah plugin. Additionally, the application can also be run in the background. The user can still participate in the network of vehicles by running Messiah application, even if the application is minimized or the user is running another application.

4 Implementation Details

Fig.3 shows different working blocks of our proposed application. The "messiah service" controls the functioning of the Messiah application. This service also receives the mode of operation from the user input. For sending and receiving data, the responsible thread is the "communicator" thread.

In order to generate a unique ID for a node, the working block responsible is the "IdGenerator" class. To track the current location of the device, the location service is provided by the "locator" class. The messages that are received from the network are stored by the "DatabaseManager" class. This working block also works as a cache for messages, and it is used to retrieve information later when needed.

The messages circulating in the network follow a structure composed of the following fields: node-ID, time stamp, type of message, current location of the

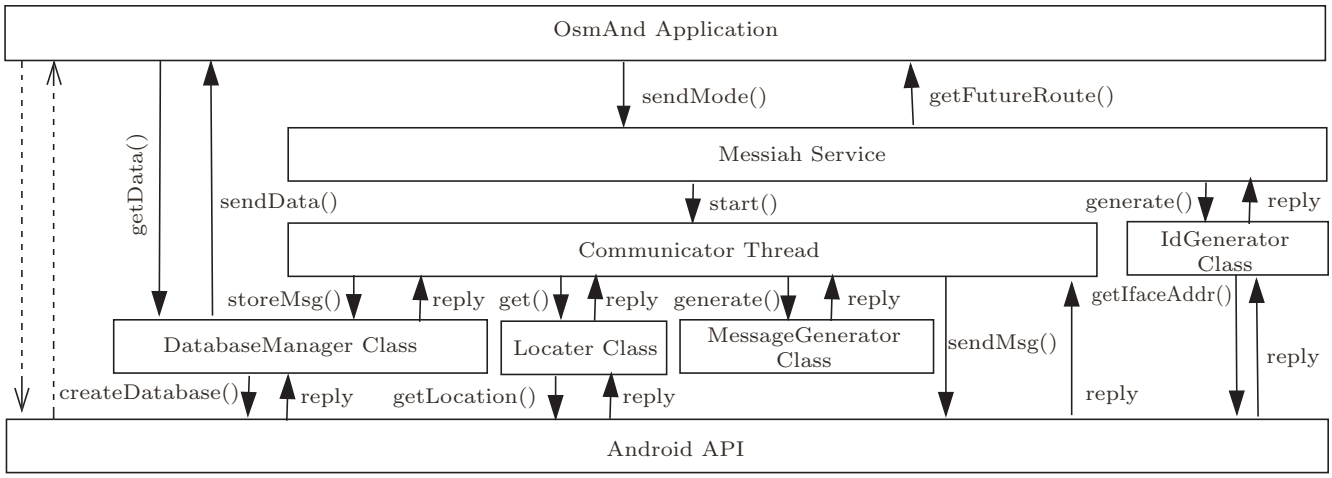


Fig.3. Architecture of the Messiah application.

sender, intended future route, and destination. As shown in Fig.4, the fields are as follows.

Fig.4. Information contained in the message used by the Messiah application.

• *ID*. It contains a unique ID generated by the IdGenerator class.

• *TimeStamp*. It indicates the time when the message is sent on the sender side, and the time when the message is received on the receiver side.

• *Type*. It contains the type of messages based on the sender, showing whether it is an administrative mode, SOS, or "bye-bye" message.

• *Location*. It contains the geographical information of the sender (in latitude and latitude).

• *Future Route*. It contains the GPS points that define the sender's path. By default, we have defined 50 GPS points that will draw the path or trajectory of the source sender.

• *Destination*. It contains the geographical information of the destination point.

Messages coming from administrative or SOS nodes are basically identical. The administrative and SOS message types have the same function, requiring a distinction between them in order to have different displays on the screen symbolized by different icons. The other type of messages, known as "bye-bye" messages, is used to signal when a node that had been previously working in the administrative or SOS mode is now leaving that mode. This notifies the other nodes in the network where these nodes are going to either switch to the civil mode or leave the network.

The sending procedure sequence is presented in Fig.5 in more details. The sending process will have to go through packet generation, and it will check the user

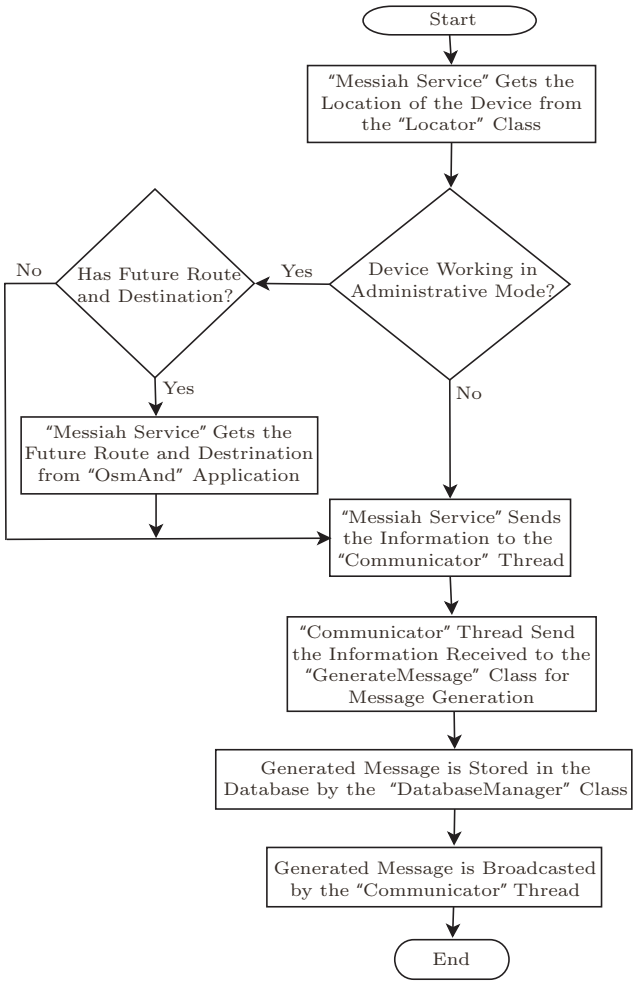


Fig.5. Flow chart for sending packets.

input mode, the future route, and the destination. Initially, the application will get the current location of the device with the help of the "locater" class. The application will then prompt the user to indicate whether the application will use the administrative mode. If it remains in civil mode, the application will directly generate messages. However, if the device switches to the administrative or SOS mode, the application will then check if a user has a future route and a destination. If it has no future route and destination, the process continues to the "communicator" thread. On the other hand, if the user has a future route and destination, the application will extract them, and then continue to send the information to the "communicator" thread. The "communicator" thread, which has the role of handling the communication functions, will send the information to the "GenerateMessage" class for message generation. The message generated is stored in the "DatabaseManager" class. Finally, these messages that have gone through the whole process are broadcasted onto the network with the aid of the "communicator" thread.

All three modes work similarly when receiving messages, which means that the vehicle under the administrative or SOS profiles will receive and broadcast received packets in the same manner with vehicles working in the civil mode. This node will generate and broadcast messages containing their ID, local timestamp, type of message (indicating the mode they are working in), current location, future route, and destination.

Fig.6 provides a more general overview of the process for all modes. In this process, nodes only receive information and forward it to their neighbors. Initially, the communicator will check for the incoming messages. When a message is received, it will first check its validity. The validity of the message is then checked based on the distance towards the source of the information. If the distance to the source is more than 1 kilometer, the application will check the database to determine if the node already exists. If this is the case, the information entry is deleted from the database, and the message is dropped. Otherwise, if the distance to the source is less than 1 kilometer, the application will then check the message type. If it is a "bye-bye" message, the message is rebroadcasted and then dropped. At the same time, the entry associated with the node whose message is received is deleted from the database. Otherwise, if the message is of the administrative or SOS type, it is checked to determine whether it is new. Notice that a packet is defined as new if the ID of that incoming packet does not match any of the identifiers of existing packets in the database. In case the message is new, the local timestamp is added to the message and it is stored in the database. In this way, when a node connectivity loss or network partition event occurs, the message is still stored with the adequate time stamp. The message without the local timestamp will then be rebroadcasted in the network. As a side note, nodes working in the administrative or SOS mode may leave the network at any time, without notifying its neighbors using "bye-bye" messages. In order to prevent having a message that is stale in the database, the application will check whether database entries have been there for 10 seconds without any updates. If the mentioned entries are still available, they are eliminated from the database.

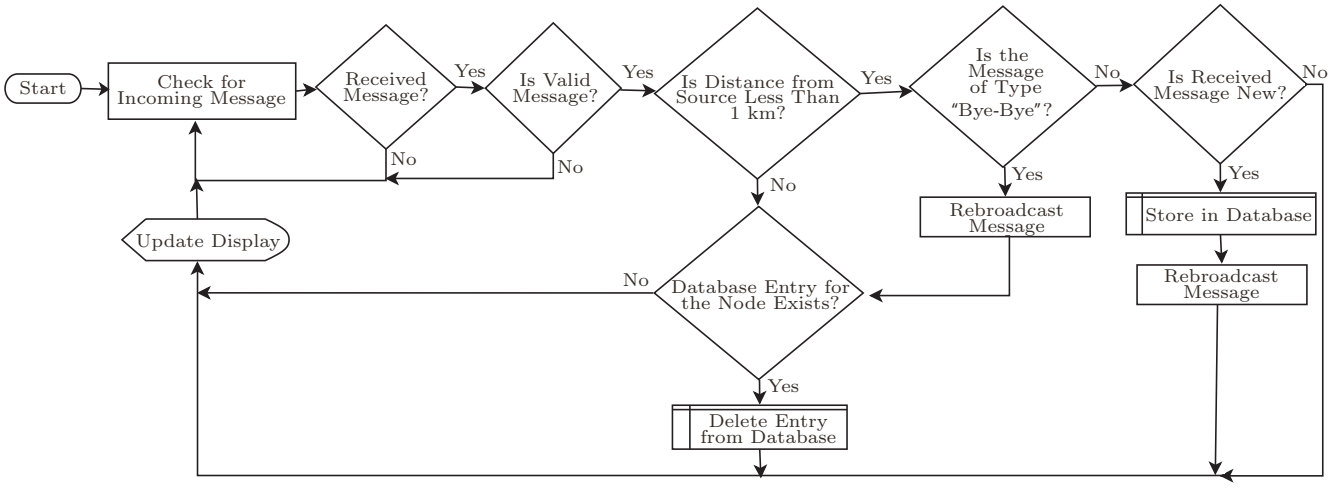


Fig.6. Flow chart for receiving packets.

5 Deployment with GRCBox

Ad-hoc connectivity is required for our application since it is deployed for V2V (vehicle-to-vehicle) communications. Android devices do not include options to enable ad-hoc network connectivity by default. Thus, a support device that provides functionality to create an ad-hoc network is required. We have selected the GRCBox[31] as the device that delivers V2V functionality, as it can create ad-hoc communications between smartphones without having to root them. GRCBox is a low-cost device that has multiple interfaces based on the Raspberry Pi platform. This connectivity device integrates Android smartphones, and it supports V2X communications.

GRCBox is based on a single-board computer (Raspberry Pi 2 model B). This low-cost device costs only 35 USD and has the size of a credit card. This computer can perform low-scale routing. Therefore, with the Debian-based Raspbian distribution installed in the computer, current networking hardware is supported along with several interfaces. This device has two modes of connectivity. One mode provides access point functionality, and it is associated with the inner interface, allowing the connection to the GRCBox using smartphones that support WiFi communications in the 2.4 GHz band. The goal is to provide connectivity to smartphone users by creating a local network via WiFi links. The other mode includes ad-hoc connectivity, being provided by the outer interface. This outer interface offers ad-hoc communications for vehicular networks, and it can be configured for usage in the 5.8 GHz band. Another interface can be added to the outer interface in order to connect to the Internet, as well as to offer 4G connectivity, as shown in Fig.7.

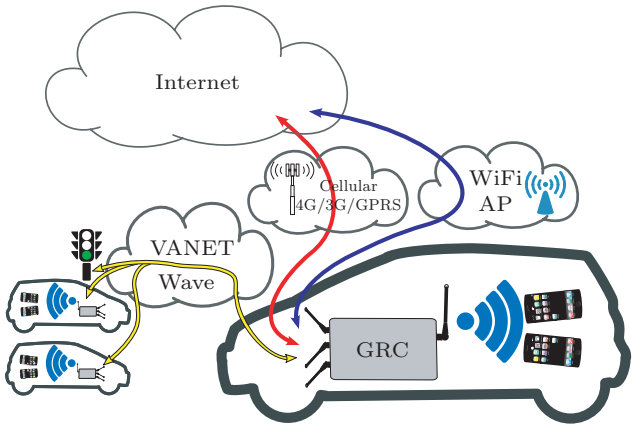


Fig.7. GRCBox hardware module connected to a VANET with three different nodes[31].

Our GRCBox module has a range of services that enable detecting and connecting to the GRCBox, forwarding packets based on connection rules. Other services include 1) listening to multicast and broadcast packets, 2) providing the status of the outer interface, and 3) managing the whole communication. The interaction with the GRCBox basically consists of creating a set of rules addressing different network protocol parameters. This includes the ports, the destination address, the communication type (multicast or unicast), and the protocol ID (UDP or TCP).

In detail, these services are: 1) the discovery service, to transparently detect and connect to the GRCBox service without any previous knowledge; 2) the routing and headers modification service, that forwards packets based on connection rules defined by the application; 3) the multicast and broadcast proxy service, which listens for multicast and broadcast packets on all interfaces, forwarding them from the inner interface to the outer interface as defined by rules; 4) the interface monitoring service, which is the service that provides information about the status of the outer interface; 5) the core service, which is in charge of the communication between the application and the connection rules validation module, updating the rules' database, routing configuration, and managing multicast proxy services accordingly.

The GRCBox's inner interface acts as a soft-AP (access point) for smartphones. Once these smartphones are connected to the GRCBox, they can access all the services running on the external networks. Since every connection is forwarded by the GRCBox, any application requiring the use of an available interface that differs from the default one (providing Internet connectivity) must notify its requirements to the GRCBox. These steps require rules that are defined by: rule type, interface name, protocol, source port, source address, destination port, and destination address. The steps taken by GRCBox-aware applications include: 1) checking the GRCBox availability, 2) registering the application, 3) checking the status of the interface, 4) registering the desired rule, 5) transmitting data, 6) closing the connection, and 7) disconnecting the application.

The GRCBox works as a router for our Messiah application. In particular, the GRCBoxes are located on vehicles to create a vehicular ad-hoc network between them. The GRCBox also allows creating a connection with an Android device that is used by a user inside the vehicle by offering a local network. With the Messiah application running at the user's side using an Android

device, packets are generated and sent locally to the GRCBox. These packets will then be forwarded from the local network to the external ad-hoc network used for VANET communications. In order to realize this type of communications, rules have to be defined. We have extended our Messiah application with an additional class so that the required rules are set automatically. This allows the GRCBox to distinguish between regular Internet traffic and VANET traffic in a transparent manner.

In terms of hardware, a configuration of the interfaces is required so that it discriminates between the internal and the external one. The minimum of two interfaces is required to provide VANET communications. One inner interface would be an access point (AP) for the Android devices. On the other hand, the outer interface would provide a VANET of GRCBoxes that are located in nearby vehicles. The network devices used as outer interfaces have the ability to transmit in the 5 GHz band. V2V communications can be realized as all the configuration is correctly set up. The integration of our Messiah application with the GRCBox platform is explained in detail in Section 6, where we perform the validation of our prototype.

6 Prototype Validation and Results

In this section, we present the assessment of our application's performance. The test was realized in terms of functionality and network capabilities. We tested the features of Messiah, and we also measured the applicability of our application in vehicular network environment, as part of the network performance test. Reliability, timeliness and packet load are assessed and regarded as critical in this experiment.

6.1 Functionality Test

The functionality of the application was assessed as part of the initial test. The test was completed on the campus of the Universitat Politècnica de València (UPV) by deploying real cars, each with a GRCBox unit inside along with an Android device.

The application is integrated with the GRCBox module and relies solely on the ad-hoc network that is formed by GRCBoxes located inside vehicles, as shown in Fig.8.

These GRCBoxes are used by participating vehicles to forward received data to another GRCBox within the communication range. The forwarding of the data, making the GRCBox act as a relay node, will occur until the data reaches the destination node.

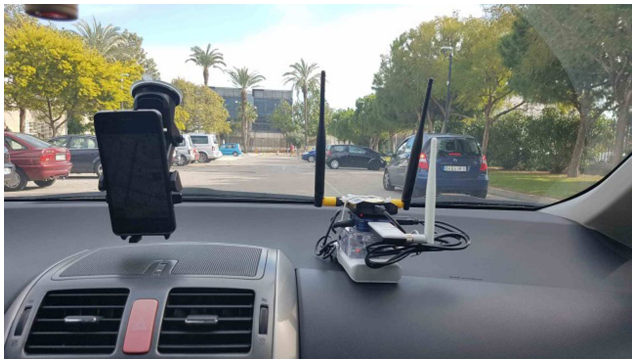


Fig.8. A photograph from within the car.

The ad-hoc mechanism is depicted in Fig.9. The Android device connected to the GRCBox will establish a path for data so that it can be forwarded to the outside ad-hoc network formed by the connected GRCBox. Each car has a GRCBox and an Android device inside, hence establishing a VANET. The communication in this network relies on IEEE 802.11-based broadcast traffic.

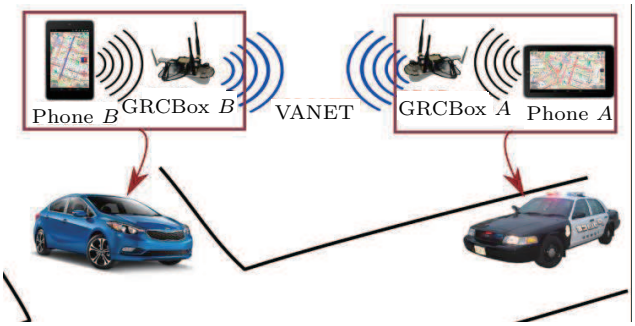


Fig.9. Structure of the ad-hoc network.

6.2 Network Performance Test

We assessed the network performance of our application in three different environments with different street characteristics (see Fig.10 and Fig.11). The objective is to test the application running in the environments having different Line of Sight (LOS) conditions. The experiment involved two cars equipped with a GRBox each. This real testbed uses broadcast traffic in its communication system, meaning that the communication is not point-to-point as the information that is spread in the ad-hoc network uses broadcast.

The first experiment was performed in a dense residential area in Valencia, where buildings create an urban canyon, as it is a populated neighborhood, creating

non-line of sight (NLOS) conditions with the highest level of obstruction for V2V communications.

(a)

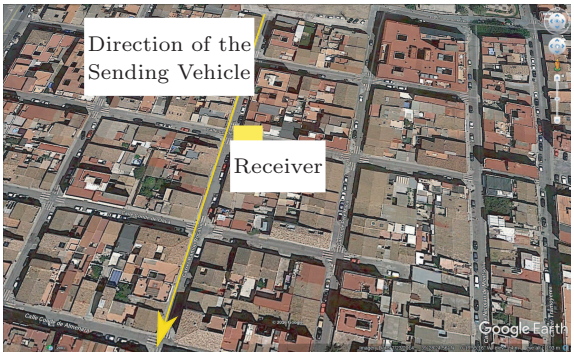
(b)

(c)

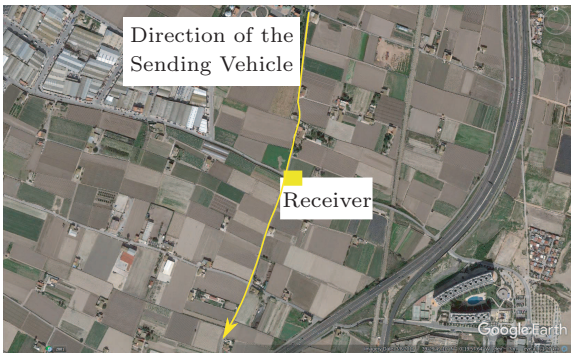
Fig.10. Location of the network performance test. (a) Scenario 1. (b) Scenario 2. (c) Scenario 3.

The second scenario was performed in a rural area, where every angle in the street will mostly support LOS conditions. No significant obstructions are present in this environment, except small vegetation and few buildings. This scenario can be classified as an environment with the lowest level of obstruction.

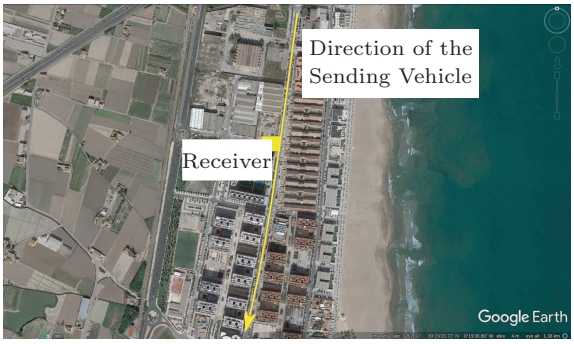
The third scenario has a moderate level of obstruction. It was performed in a less populated urban area, where buildings and vegetation, meaning that obstructions are moderately sparse in the environment. Thus, depending on the specific angle, LOS and NLOS conditions may take place.



(a)



(b)



(c)

Fig.11. Aerial view of the location of the network performance test. (a) Scenario 1. (b) Scenario 2. (c) Scenario 3.

Our first measurement targets packet delivery in these three environments. To determine the maximum distance covered by communications using this application, packet losses are calculated by determining how many packets are lost by the receiver when communicating with the sender for a specific distance interval, being that regular spatial intervals are defined.

As a result, in Fig.12 we can see that few packets are lost in all three scenarios up to 10 meters. In the dense residential area, loss levels increase when reaching 40 meters, in which only 50 percent of the packets are received. In a less-populated urban area, this similar case takes place when the communication range is of

60 meters, being that the 50th percentile corresponds to a distance of 80 meters.

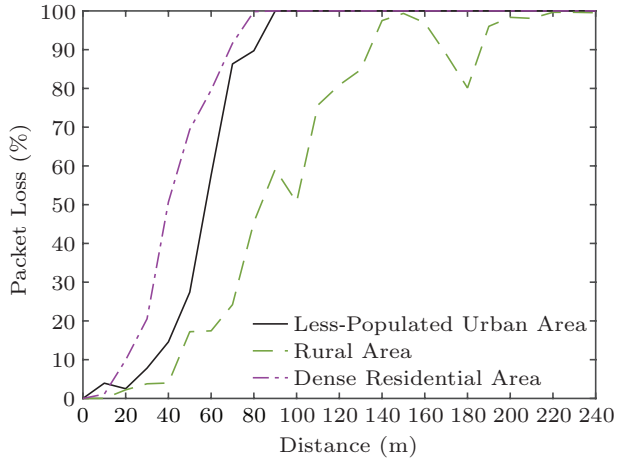


Fig.12. Distance vs packet loss.

The application can handle communications for distances up to 240 meters in rural areas, but in a less-populated urban area, nearly 100% of packet loss is detected when the communication distance reaches 100 meters. In fact, under NLOS conditions, and with the highest level of obstruction (urban canyon), the communications are successful for distances up to 80 meters.

The next assessment consists of a time-probability analysis by focusing on consecutive packet arrivals. We tested a 10 Hz packet sending rate in this experiment. We chose this rate so that the inter-packet arrival time is about 100 ms under ideal conditions. The probabilities of having inter-packet arrival time below the indicated time are derived using a cumulative distribution function. We calculated the probabilities for the three types of scenarios with different street characteristics. Fig.13 shows that the inter-packet arrival time is never lower than 100 ms. This is due to the fact that the transmission rate is 10 Hz (10 packets per second). Until 130 ms, the probability grows in all three scenarios. However, in both rural and regular urban areas, 97% of the inter-packet arrival time is below 200 ms. On the other hand, in an urban canyon (dense residential area) scenario, values of 500 ms are reached for nearly the same probability. The difference between the less-populated urban area and the rural area is that, at 150 ms, the cumulative probability for the rural area reaches 0.8, whereas in the less-populated urban area it is only 0.75. The cumulative probability reaches its maximum for all scenarios when the inter-packet arrival time is of 900 ms. These inter-packet arrival time results are related to the packet losses occurred in the tests. Greater waiting time between consecutive packet arrivals means that packet loss bursts are greater. The ideal condition is that the inter-packet arrival time has the same value with the inter-packet sending time. This condition occurs if there are no packet losses, meaning that no additional delay occurred. The results obtained from our assessments indicate that, when using the application, the time of notifying nearby vehicles about emergency situations is under 1 second in the worst case, thus being adequate for safety applications.

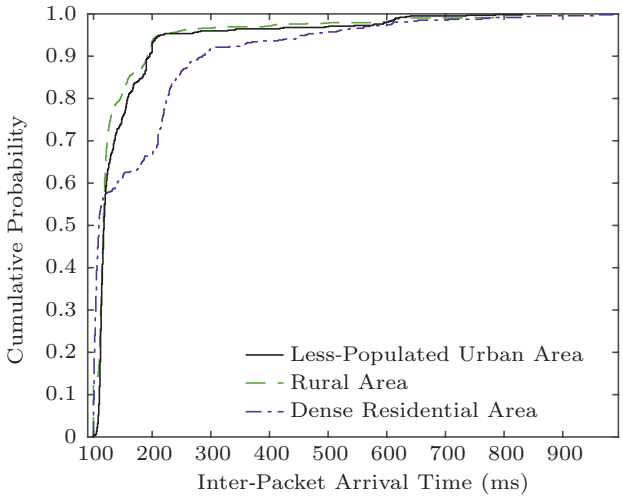


Fig.13. Cumulative distribution function of inter-packet arrival time.

In our last tests, we assessed the traffic overhead based on the route size advertised by the sender. Traffic overhead is calculated based on the GPS points advertised with three different refresh rates. Since we use UDP packets for sending the information, the packets advertised cannot be more than 1.5 KB. The GPS points advertised in this context contain the future route information provided by the sender. We varied the number of GPS points embedded into the packet to detail the future route to a smaller or greater extent. As we can see in Fig.14, the refresh rate and the route size affect the traffic overhead. The more GPS points we have, the higher the value of the traffic overhead; in particular, if packets containing 70 GPS points are advertised every second, this will generate 5 MB of traffic overhead per hour. If the inter-packet period is reduced, the traffic overhead will increase accordingly. These values show the need for adequating the route size that should be included in the packet so that it does not saturate the wireless medium.

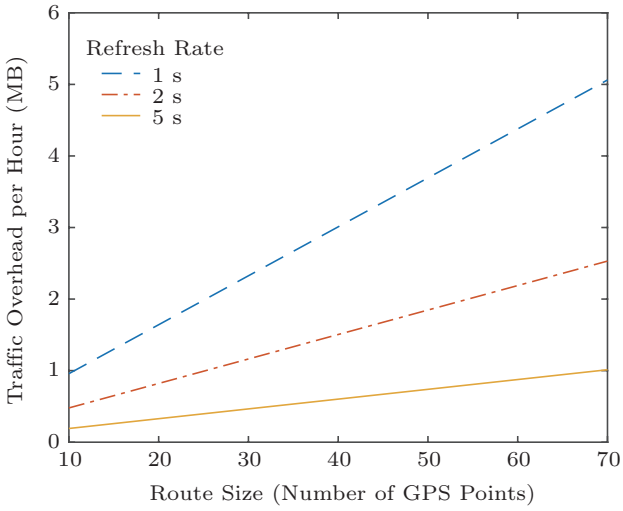


Fig.14. Traffic overhead based on route size.

7 Security Issues

Security issues might arise in these types of applications. Malicious users might attempt to join the network as administrative devices and mislead other users. That being said, one issue is associated with the authenticity of the roles or modes of the nodes that use this application. To access the administrative features, or to use the application in administrative node, the user in the vehicle should have a level of authority and proper certification. For instance, the user should be a policeman, a fire brigade member, or a paramedic staff.

Other issues might include the possible application-layer types of attacks. In Messiah, we have not used any extra security measures to protect our application. This will be included in our future work as an improvement. The performance of the application might be influenced by some possible attacks such as denial of service[32], non-control-data corruption[33], malwares, and hacks.

Denial of service can occur if the server is occupied in the communication with the attacker at the time that a legitimate client needs a response from the server. Approaches to be considered to get rid of this attack can be the use of public key cryptographic protocols, such as the solutions in [34].

Attacks by accessing and modifying the user identity, configuration or decision-making data are called non-control-data corruption attacks[33]. One way to resolve this is by storing the data that is private to the application. Storing private data is possible for developers, as the application is designed based on the Linux environment. Hence, non-control-data corruption attacks are difficult to execute④.

Another type of attacks might include inserting malware in the original code. This can modify the actual behavior of the application. This issue can be resolved by using control-flow integrity[35]. Using this solution, the control-flow policy would be enforced during runtime, and its mechanism would be embedded in the software executables.

Finally, another security threat might include hacking the software through reverse engineering. By modifying the behavior of the application, the attacker can also supply the false information of the vehicle's location instead of its real location provided by its GPS. To address this problem, the whole system should be embedded within the car, along with the application, and the system should have a manufacturer certification.

8 Conclusions

In this paper, we introduced Messiah, a novel application providing navigation information and emergency notifications with the goal of creating a safe traffic environment. Messiah's features include an on-screen display that merges maps and the presence of emergency vehicles (ambulances, police cars, fire brigades), along with their routes. This allows civil vehicles to be warned about their presence so that the drivers of those vehicles can take alternative navigation decisions to reduce the congestion along the routes followed by emergency vehicles. Also, civil vehicles can benefit from this application by being able to notify nearby vehicles in case a help is required, or may even contact nearby emergency vehicles to request immediate assistance by using the SOS mode. The application is also suitable for VANET environments if combined with GRCBox. In fact, the latter is used to enable V2V communications on vehicles, allowing them to create an ad-hoc network. In this way, smartphones running our application can seamlessly distribute messages in the ad-hoc network. Our application can also be immediately deployed and join a VANET without requiring user intervention by automatically detecting and configuring the GRCBox.

Our application was assessed in terms of network performance under different conditions, showing that it can properly meet the requirements of typical emergency situations. Regarding packet delivery effectiveness, it was assessed under different conditions. In the worst-case scenario, where the level of obstruction is

④Data storage. https://developer.android.com/guide/topics/data/data-storage.html, Jan. 2018.

high and causes NLOS conditions, packets can still be successfully delivered for distances up to 80 meters. Additionally, in the same scenario, if we set the message sending rate to 10 Hz, the inter-packet arrival time was below 900 ms in all cases. This means that despite channel losses, warnings can still be delivered in less than 1 second. In order to make the application ubiquitous and fitted to larger-scale ITS environments, future work will include the integration of the application with the infrastructure and the Internet, thereby being more adapted to V2X communications.

References

[1] Papadimitratos P, De La Fortelle A, Evenssen K, Brignolo R, Cosenza S. Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation. *IEEE Communications Magazine*, 2009, 47(11): 84-95.

[2] Gerla M, Kleinrock L. Vehicular networks and the future of the mobile Internet. *Computer Networks*, 2011, 55(2): 457-469.

[3] Vegni A M, Loscrí V. A survey on vehicular social networks. *IEEE Communications Surveys & Tutorials*, 2015, 17(4): 2397-2419.

[4] Eze E C, Zhang S J, Liu E J. Vehicular ad hoc networks (VANETs): Current state, challenges, potentials and way forward. In *Proc. the 20th Int. Conf. Automation and Computing*, September 2014, pp.176-181.

[5] Qi L. Research on intelligent transportation system technologies and applications. In *Proc. Workshop on Power Electronics and Intelligent Transportation System*, August 2008, pp.529-531.

[6] Gozalvez J. First Google's Android phone launched [Mobile Radio]. *IEEE Vehicular Technology Magazine*, 2008, 3(4): 3-69.

[7] Haklay M, Weber P. OpenStreetMap: User-generated street maps. *IEEE Pervasive Computing*, 2008, 7(4): 12-18.

[8] Hadiwardoyo S A, Patra S, Calafate C T, Cano J C, Manzoni P. An Android ITS driving safety application based on vehicle-to-vehicle (V2V) communications. In *Proc. the 26th Int. Conf. Computer Communication and Networks*, July 31-August 3, 2017.

[9] Eriksson J, Balakrishnan H, Madden S. Cabernet: Vehicular content delivery using WiFi. In *Proc. the 14th ACM Int. Conf. Mobile Computing and Networking*, September 2008, pp.199-210.

[10] Gerla M, Weng J T, Giordano E, Pau G. Vehicular testbeds-validating models and protocols before large scale deployment. In *Proc. Int. Conf. Computing Networking and Communications*, January 30-February 2, 2012, pp.665-669.

[11] Wahlström J, Skog I, Händel P. Smartphone-based vehicle telematics: A ten-year anniversary. *IEEE Trans. Intelligent Transportation Systems*, 2017, 18(10): 2802-2825.

[12] Whipple J, Arensman W, Boler M S. A public safety application of GPS-enabled smartphones and the Android operating system. In *Proc. IEEE Int. Conf. Systems Man and Cybernetics*, October 2009, pp.2059-2061.

[13] Meseguer J E, Calafate C T, Cano J C, Manzoni P. DrivingStyles: A smartphone application to assess driver behavior. In *Proc. IEEE Symp. Computers and Communications*, July 2013, pp.000535-000540.

[14] You C W, Lane N D, Chen F L, Wang R, Chen Z Y, Bao T J, Montes-De-Oca M, Cheng Y T, Lin M, Torresani L, Campbell A T. CarSafe app: Alerting drowsy and distracted drivers using dual cameras on smartphones. In *Proc. the 11th Annual Int. Conf. Mobile Systems Applications and Services*, June 2013, pp.461-462.

[15] Patra S, Arnanz J H, Calafate C T, Cano J C, Manzoni P. EYES: A novel overtaking assistance system for vehicular networks. In *Proc. the 14th Int. Conf. Ad-Hoc Networks and Wireless*, June 2015, pp.375-389.

[16] Togneri M C, Deriaz M. On-board navigation system for smartphones. In *Proc. Int. Conf. Indoor Positioning and Indoor Navigation*, October 2013.

[17] Dancu A, Franjcic Z, Fjeld M. Smart flashlight: Map navigation using a bike-mounted projector. In *Proc. the SIGCHI Conf. Human Factors in Computing Systems*, April 2014, pp.3627-3630.

[18] Yamabe T, Ikegami S, Ishizaki A, Kitagami S, Kiyohara R. Car navigation user interface based on a smartphone. In *Proc. the 7th Int. Conf. Mobile Computing and Ubiquitous Networking*, January 2014, pp.85-86.

[19] Matsuyama S, Yamabe T, Takahashi N, Kiyohara R. Intelligent user interface of smartphones for on-vehicle information devices. *Procedia Computer Science*, 2014, 35: 1635-1643.

[20] Kovacevic B, Kovacevic M, Maruna T, Papp I. A java application programming interface for in-vehicle infotainment devices. *IEEE Trans. Consumer Electronics*, 2017, 63(1): 68-76.

[21] Liu R L, Liu H Z, Kwak D, Xiang Y, Borcea C, Nath B, Iftode L. Themis: A participatory navigation system for balanced traffic routing. In *Proc. IEEE Vehicular Networking Conf.*, December 2014, pp.159-166.

[22] Liu R L, Liu H Z, Kwak D, Xiang Y, Borcea C, Nath B, Iftode L. Balanced traffic routing: Design, implementation, and evaluation. *Ad Hoc Networks*, 2016, 37: 14-28.

[23] Vochin M, Zoican S, Borcoci E. Intelligent system for vehicle navigation assistance. In *Proc. World Conf. Information Systems and Technologies*, April 2017, pp.142-148.

[24] Sha W J, Kwak D, Nath B, Iftode L. Social vehicle navigation: Integrating shared driving experience into vehicle navigation. In *Proc. the 14th Workshop on Mobile Computing Systems and Applications*, February 2013, Article No. 16.

[25] Kwak D, Liu R L, Kim D, Nath B, Iftode L. Seeing is believing: Sharing real-time visual traffic information via vehicular clouds. *IEEE Access*, 2016, 4: 3617-3631.

[26] Djajadi A, Putra R J. Inter-cars safety communication system based on Android smartphone. In *Proc. IEEE Conf. Open Systems*, October 2014, pp.12-17.

[27] Dorairaj P, Mohammed A, Grbic N. Location-Aware services using Android mobile operating platform for safety, emergency and health applications. In *Mobile Health*, Adibi S (ed.), Springer, 2015, pp.283-298.

[28] Dorairaj P, Ramamoorthy S, Ramalingam A K. Emergency based remote collateral tracking system using Google's Android mobile platform. In *Advances in Computer Science Engineering & Applications*, Wyld D C, Zizka J, Nagamalai D (eds.), Springer, 2012, pp.391-403.

[29] Zulkafi A Z, Basri S, Jung L T, Ahmad R. Android based car alert system. In *Proc. the 3rd Int. Conf. Computer and Information Sciences*, August 2016, pp.501-506.
[30] Tornell S M, Calafate C T, Cano J C, Manzoni P, Fogue M, Martinez F J. Evaluating the feasibility of using smartphones for ITS safety applications. In *Proc. the 77th IEEE Vehicular Technology Conf.*, June 2013.
[31] Tornell S M, Patra S, Calafate C T, Cano J C, Manzoni P. GRCBox: Extending smartphone connectivity in vehicular networks. *International Journal of Distributed Sensor Networks*, 2015, 2015: Article No. 5.
[32] Mirkovic J, Dietrich S, Dittrich D, Reiher P. Internet Denial of Service: Attack and Defense Mechanisms (Radia Perlman Computer Networking and Security). Prentice Hall PTR, 2004.
[33] Chen S, Xu J, Sezer E C, Gauriar P, Iyer R K. Non-control-data attacks are realistic threats. In *Proc. the 14th Conf. USENIX Security Symp.*, July 31-August 5, 2005.
[34] Fung C K, Lee M C. A denial-of-service resistant public-key authentication and key establishment protocol. In *Proc. the 21st IEEE Int. Performance Computing and Communications Conf.*, April 2002, pp.171-178.
[35] Abadi M, Budiu M, Erlingsson Ú, Ligatti J. Control-flow integrity. In *Proc. the 12th ACM Conf. Computer and Communications Security*, November 2005, pp.340-353.

Seilendria A. Hadiwardoyo is a Ph.D. candidate at the Department of Computer Engineering, Universitat Politècnica de València, València, Spain. He received his B.Eng. degree in computer engineering from Fakultas Teknik, Universitas Indonesia, Depok, Indonesia, and the Pre-Master's Diploma (Maitrise, equivalent to Pg.Dip.) in computer science from Universite de Lille 1, Villeneuve-d'Ascq, France, both in 2012. Hadiwardoyo obtained his M.S. degree in web, multimedia, and networks from Universite de Bretagne-Sud, Vannes, France, and his Predoctoral Diploma (equivalent to M.Phil.) in informatics from Universidade do Minho, Braga, Portugal, in 2013 and 2015 respectively. His research interests include ad hoc and vehicular networks, mobile applications, and implementation of intelligent transportation systems (ITS).

Subhadeep Patra is a Ph.D. student of the joint doctoral programme offered by the Universitat Politècnica de València, Spain, and Ghent University, Belgium. His research interest includes intelligent transportation systems (ITS) and vehicular adhoc network (VANET). In 2013, Subhadeep received his Master's degree (M.Tech) in distributed and mobile computing from Jadavpur University, India, and in 2011, his Bachelor's degree (B.Tech) in information technology from University of Kalyani, India.

Carlos T. Calafate is an associate professor in the Department of Computer Engineering at the Universitat Politècnica de València (UPV) in Spain. He graduated with honours in electrical and computer engineering at the University of Oporto (Portugal) in 2001. He received his Ph.D. degree in informatics from the Technical University of Valencia in 2006, where he has worked since 2002. His research interests include ad-hoc and vehicular networks, UAVs, Smart Cities & IoT, QoS, network protocols, video streaming, and network security. To date he has published more than 300 articles, several of which in journals including IEEE Transactions on Vehicular Technology, IEEE Transactions on Mobile Computing, IEEE/ACM Transactions on Networking, Elsevier Ad hoc Networks and IEEE Communications Magazine. He has participated in the TPC of more than 150 international conferences. He is a founding member of the IEEE SIG on Big Data with Computational Intelligence.

Juan-Carlos Cano is a full professor in the Department of Computer Engineering at the Universitat Politècnica de València (UPV) in Spain. He earned his M.Sc. and his Ph.D. degrees in computer science from the UPV in 1994 and 2002 respectively. From 1995∼1997 he worked as a programming analyst at IBM's manufacturing division in Valencia. His current research interests include vehicular networks, mobile ad hoc networks, and pervasive computing.

Pietro Manzoni is a professor of computer networks at the Universitat Politècnica de València, Spain. He holds his Ph.D. degree in computer science from the Politecnico di Milano, Italy. His research activity is related to networking and mobile systems applied to intelligent transport systems (ITS), opportunistic networking for smart cities and the Internet of Things. He is a senior member of IEEE.