A novel On-Board Unit to Accelerate the Penetration of ITS Services

Sergio M. Tornell, Subhadeep Patra, Carlos T. Calafate, Juan-Carlos Cano and Pietro Manzoni
Universitat Politècnica de València
Camino de Vera, s/n, 46022 Valencia, Spain
sermarto@upv.es,subpat@doctor.upv.es, {calafate, jucano, pmanzoni}@disca.upv.es

***Abstract*—In-vehicle connectivity has experienced a big expansion in recent years. Car manufacturers have mainly proposed OBU-based solutions, but these solutions do not take full advantage of the opportunities of inter-vehicle peer-to-peer communications. In this paper we introduce GRCBox, a novel architecture that allows OEM user-devices to directly communicate when located in neighboring vehicles. In this paper we also describe EYES, an application we developed to illustrate the type of novel applications that can be implemented on top of the GRCBox. EYES is an ITS overtaking-assistance system that provides the driver with real-time video fed from the vehicle located in front. Finally, we evaluated the GRCbox and the EYES application and showed that, for device-to-device communication, the performance of the GRCBox architecture is comparable to an infrastructure network, introducing a negligible impact.**

***Index Terms*—Vehicular Networks, V2V, Smartphone, GRCBox, VANET, ITS, Implementation, Video**

I. Introduction

When applying Intelligent Transportation Systems (ITS) to roads, several technologies are combined to provide Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communications [1]. Opportunistic V2V networks [2] allow implementing applications such as road-status notifications [3], vehicle platoon coordination, or collaborative content downloading [4]. Although the technology is ready for deployment, it is expected that car manufacturers will introduce it gradually, starting with high-cost models, which, coupled with the low renewal rate of the vehicle fleet, will slow down the penetration of Vehicular Networks (VNs). In addition, dashboard-integrated On Board Units (OBUs) typically become technologically obsolete after a few years, and they are usually not designed to be updated or replaced during the whole vehicle lifetime, which leads to unsatisfied users.

Meanwhile, the popularisation of portable devices, such as smartphones and tablets, has brought devices with multiple network interfaces to almost everyone's pocket. Smartphones are continuously carried by users and have multiple network interfaces, which makes them a suitable platform for implementing applications based on opportunistic contacts. However, not only is the smartphone's connectivity restricted to infrastructure-based networks, such as WiFi or 3G/4G networks, but also the number of simultaneous active network interfaces is limited to one. These restrictions limit the adoption of smartphones for applications based on peer-to-peer communications in vehicular scenarios.

To connect portable devices to external networks such as Vehicular Ad-Hoc Networks (VANETs), we have designed the GRCBox Architecture. The GRCBox Architecture is based on the GRCBox Connectivity Manager (GCM), which is responsible for creating an intra-vehicle WiFi network. User devices inside the vehicle can connect to this network to share contents and to reach any of the external networks, as depicted in Figure 1. GRCBox allows implementing Internet-independent solutions that focus on applications that exploit local connectivity to provide new services, such as platoon-oriented applications where friends or workers share information while travelling together in different vehicles. The GRCBox Architecture provides a Representational State Transfer (REST) interface [5] and it is based on basic IP networking, thereby minimising the modifications required to create GRCBox-aware applications. GRCBox also eliminates the dependency on car manufacturers when implementing V2V communications; by using GRCBox, users can now implement their own small-scale VANETs.

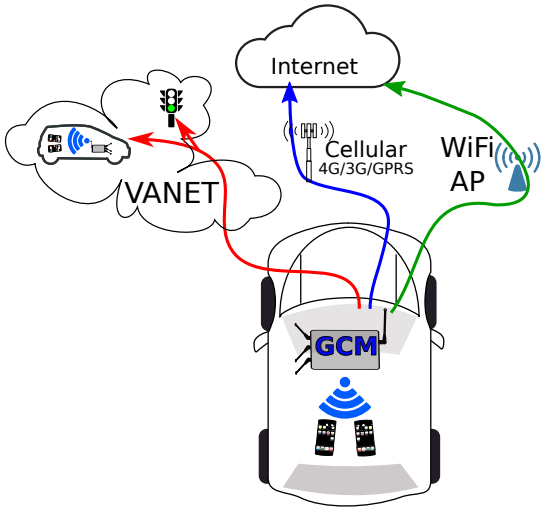


Fig. 1: A GRCBox Connectivity Manager (GCM) connected to several networks.

To illustrate the use and flexibility of the GRCBox architecture we have developed EYES. EYES is an application that streams a video captured from the windshield of the vehicle ahead to the overtaking vehicle before the overtake actually starts. This video aids the driver while starting the manoeuvre, thus he or she can abort it if it can not be performed safely. EYES is based on the GPS location of cars involved in the overtaking manoeuvre and uses the rear camera of a windshield-attached smartphone to capture the video. Its

978-1-4673-9292-1/16/$31.00 ©2016 IEEE

integration with GRCBox allows EYES to establish a direct ad-hoc link between user devices, reducing the delay and the cost of the communication.

The rest of this paper is organised as follows: in section II, we survey previous proposals in the fields of VANET communication devices and driver assistance mobile applications. Next, section III presents the GRCBox architecture. In section IV, the EYES application and its interaction with the GRCBox are presented. Performance tests for the GRCBox and EYES are presented in sections V and VI. Finally, section VII concludes the paper and presents some future work.

II. State of the Art

Previously, vendor-specific alternatives that integrate smartphones in VNs has been proposed. The Car Connectivity Consortium (CCC), which integrates companies from the automotive and the telecommunications sector, released Mirrorlink [6], a standard technology that moves the computing tasks from the OBU to the smartphone, and presents the information on the OBU's display. Users can also interact with the smartphone through the dashboard elements.

Google and Apple, two of the biggest technology companies, have also proposed their own solutions, Android Auto [7], and CarPlay [8], respectively. However, all these proposals rely on the Internet infrastructure to provide inter-vehicle connectivity, ignoring the advantages of V2V communication and opportunistic contacts. Moreover, these proposals are heavily dependent on content providers like Google or Apple, and on telecommunication operators.

To the best of our knowledge, GRCBox is the first effort to provide an interface that integrate off-the-shell smartphones into VANETs in order to allow users to create their own autonomous VN and to provide innovative peer-to-peer applications.

Concerning mobile applications for ITS, both academia and industry have shown a strong interest in the field. As a consequence many innovative applications have been developed. Focused on warning dissemination, the works [9, 10, 11, 12] aid drivers in different situations such as driving at high speed near schools, finding out the probability of accidents based on the location information, traffic incidents, or when an emergency vehicle is getting closer to the vehicle. In [13] and [14] the authors use the On Board Diagnostic 2 (OBD-II) [15] interface to detect incidents and inform other drivers.

SignalGuru [16] is another application which leverages collaborative sensing on windshield-mount smartphones, in order to predict the schedule of traffic signals. Another interesting approach is the CarSafe App [17], which analyses images from front and rear smartphones' cameras to monitor the status of the driver as well as the road.

In [18], its authors proposed and tested a system based on Dedicated short-range communications (DSRC) that, similarly to EYES, streams video during an overtaking manoeuvre from the vehicle ahead to the overtaking vehicle. They performed driving simulation tests that demonstrated the usability of such a system. Their proposal required the installation of expensive OBUs in the vehicles. On the contrary, the EYES-GRCBox combination is based on off-the-shell user devices such as smartphones and tablets.

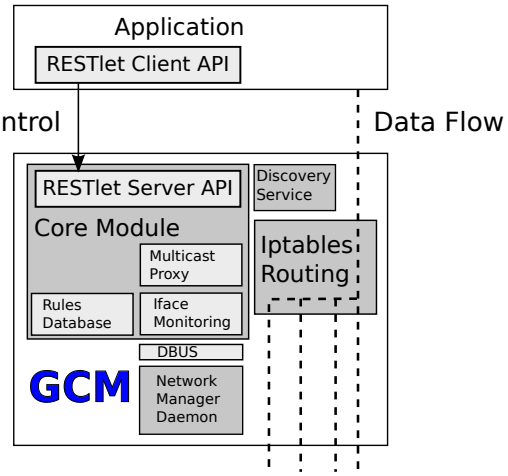


Fig. 2: GRCBox Architecture with GCM modules in detail.

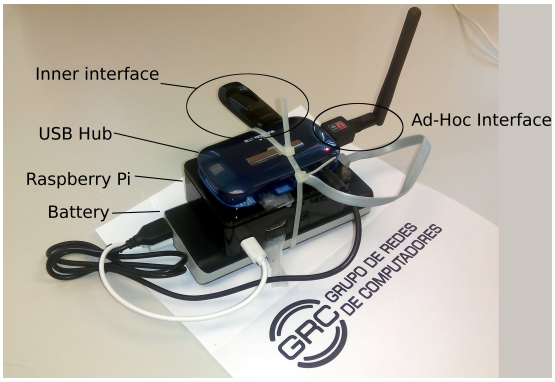
III. The GRCBox Architecture

The GRCBox Architecture defines both the GRCBox Connectivity Manager (GCM), placed in the vehicle, and, the client-server REST API, that allows applications to interact with the GCM to reach external networks. To implement the REST API we used the RESTlet framework [19], which simplifies the implementation. Figure 2 represents the architecture including both parts the GCM and the client API. An example of a GCM placed in a vehicle and connected to three different external networks is shown in Figure 1. In this example an application running in the user device may choose to connect to the VANET for local communication, or connect to either the cellular network or the WiFi network to reach the Internet. In this section, we first detail the different software modules running in the GCM. Then, we offer a general overview of the interaction between the GCM and the User Application.

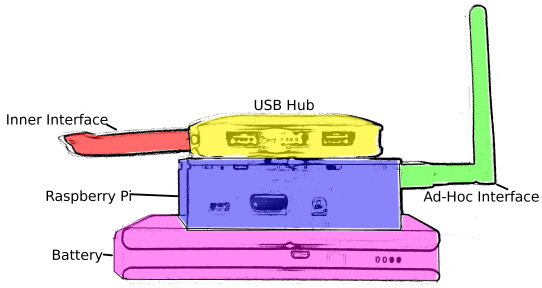
A. The GCM

The GRCBox Connectivity Manager (GCM), which is placed inside vehicles, must have at least one WiFi interface to which user devices are connected to (called *inner interface*), and one or more *external interfaces* used to provide connectivity to external networks. The GCM is composed of several modules that work together. A scheme of the different components, their connections, and the paths traversed by data flows is presented in Figure 2. The GCM software is based on a Linux operating system, and it takes advantage of several well-known Linux services to provide the desired functionality. The different components running in the GCM are the following:

- **Discovery Service:** The Linux daemon *dnsmasq* is used to answer DHCP and DNS requests. It is configured to resolve the *"grcbox"* domain name to the GCM inner interface. This way clients on the inner network can connect to the GCM without information about its IP address by attempting a connection to *"http://grcbox/"*.
- **Packet Forwarding:** To define fine grained, per connection routing, GCM uses Iptables for connection filtering



(a) The GRCBox connected to a battery and an ad-hoc interface.



(b) Schematic view of GRCBox.

Fig. 3: Minimum GRCBox setup used for testing.

and labelling, and the Linux kernel support for "Policy Routing".

- **Ifaces Monitoring:** To monitor the status of the network interfaces, GCM connects to the NetworkManager daemon using the DBUS interface to perform event-subscribing tasks.
- **Core Module:** The most important part of the GCM is its core module. The core module performs several activities: it mainly listens to clients' requests through the REST API, maintains a database of all registered rules, starts and stops multicast proxies when needed, and performs actions when events on the interfaces are notified.

To implement the first version of GCM we chose a Raspberry Pi 2 Model B[1] single board computer. The RaspberryPi is a credit-card sized computer whose cost is only 35$, but that has enough CPU power to perform low-scale network routing. In this computer we installed a Raspbian [20] distribution, which is a general-purpose Linux distribution based on Debian and optimised for the RaspberryPi. Raspbian supports most current networking hardware, avoiding common problems of other embedded operating systems. Figure 3 shows a picture of the GCM with an external ad-hoc network interface attached to a battery.

B. User Device-GCM Interaction

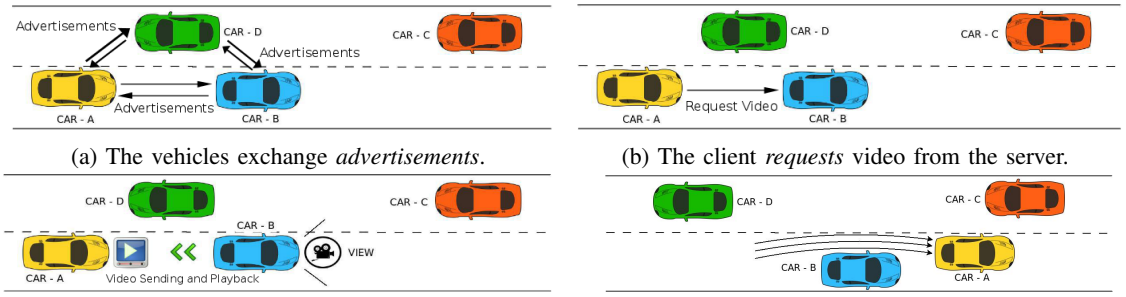
The GCM creates a WiFi access point to which smartphones, tablets, and other user devices in the vehicle will associate. Once the user devices connect to the GRCBox's wireless network, all of them are part of the same local network and, at the same time, can access to services running on the external networks. By default, every new connection is forwarded at the GCM through the default Internet connected interface. When an application requires the use of any other available interface, it must notify it to the GCM. The steps an application must perform in this case, are enumerated in this section. First of all, we need to introduce the concept of "rules": A rule enables applications to choose the outgoing interface for a certain connection, or to register listeners for a defined incoming connection. A rule is defined by the following elements:

- Rule Type: The GRCBox Architecture defines three different kind of rules, *Incoming*, *Outgoing* and *Multicast*. Multicast rules define bi-directional multicast packet flows between the internal interface and one of the external interfaces.
- Interface Name: The name of the outer interface to which the rule applies.
- Protocol: The protocol of the connection. Currently, GRCBox supports UDP and TCP.
- Source Port: The source port of the connection.
- Source Address: The source IP address of the connection.
- Destination Port: The destination port of the connection.
- Destination Address: The destination IP address of the connection.

The steps that a GRCBox application must perform are the following:

1) **Check GRCBox availability:** Once the device is associated to the GRCBox wireless network, the application must check if a GCM is available. To do so, the application will try to connect to the "http://grcbox/" url to check the status of the GCM.
2) **Application Registration:** After checking the availability of the GCM, an application must register itself to get a key. This key will be used for later application-server interactions to ensure no other application but the owner of a rule can renew, remove, or modify it.
3) **Check the Status of the Interfaces:** The next step is to check the status of the different network interfaces to identify if the desired interface is available. At this point the application can also check other previously registered rules to avoid conflicts.
4) **Register the desired rule:** Now the application can register as many rules as required to configure the GCM to forward specific incoming and outgoing connections, or to forward multicast packets to external interfaces.
5) **Transmit Data:** At this point the application can effectively use the registered connections which will be forwarded according to the defined rules.
6) **Close the Connection:** When a rule is no longer required, it must be removed from the GCM. This step is optional since rules are always removed from the GCM

[1]More info at:https://www.raspberrypi.org/



(a) The vehicles exchange *advertisements*.

(b) The client *requests* video from the server.

(c) The server streams the video to the client.

(d) After overtaking, CAR-A stops playing the video and may stream video to CAR-B.

Fig. 4: Functional overview of EYES application.

database if the application is disconnected.

7) **Application Disconnection:** Once the application ends its interaction with the GCM, it should notify it to allow removing its registered rules.

The interactions between GRCBox applications and the GCM rely on the RESTlet API exposed by the GCM. The details of this API are described in [21].

GRCBox also supports the integration of third party applications by providing a management application that enables the interactive definition of new rules for non-GRCBox applications. Thereby, the user can define rules for well-known application protocols such as HTTP, POP3, etc. To illustrate the new applications that can be implemented based on the GRCBox, we present the EYES application in the next section.

IV. Using the GRCBox: The EYES Application

The goal of the EYES application is providing assistance during overtaking by streaming real-time video from the leading vehicle to the one trying to overtake. From the user point of view, EYES operation is represented in Figure 4 and is described as follows:

1) EYES announces the presence of a ready-to-broadcast smartphone (ideally attached to the dashboard of a vehicle) by sending multicast *advertisements*. Those *advertisements* include information such as the GPS location of the vehicle, its speed and its direction.
2) After receiving a multicast *advertisement*, the EYES application running in the smartphone placed in the car whose driver is trying to overtake requests a video stream from the origin of the advertisement. From now on, we will refer to these nodes as *client*, for the one trying to overtake, and *server*, for the one leading. Before sending the request, the client discriminates between different available streams based on the GPS location and other info included in the *advertisements*.
3) Once the server receives the request, it sends some control information over the UDP control connection to the client and listens for incoming TCP connections at a certain port. Then, the client establishes the TCP connection for video streaming. The UDP control connection is used during the communication to monitor the relative locations of both server and client, and to stop the video streaming when the overtaking manoeuvre completes.

To adapt EYES to the GRCBox network environment, only four rules are required. The first rule is registered when the application starts. It defines multicast forwarding from the inner interface to the external ad-hoc interface. At step 2, the request triggers the registration of three more GRCBox incoming rules at the client's GCM: a TCP rule for video streaming, and two UDP rules for the control channel established between server and client.

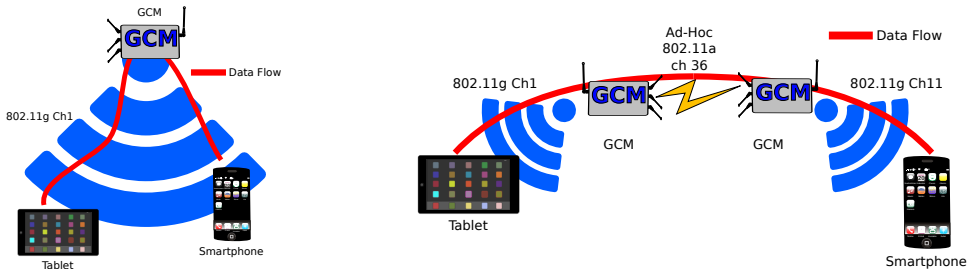
V. Testing the GRCBox

We have run experiments to evaluate the performance of the GRCBox Architecture in 2 different scenarios: the first experiment is an analysis of the maximum throughput, and the second one analyses the UDP Round Trip Time (RTT) between client and server. In the first scenario, we used an Android Nexus 7 tablet and a BQ4.5E smartphone connected to the same GCM, which acted as a standard WiFi Access Point, since connections can be established without interaction with the GCM; this is the baseline scenario. In the second scenario we connected each device to different GCMs, which were then connected to the same ad-hoc network. In this case the connection must be established through the GCMs. Table I and Figure 5 summarize the configuration of both scenarios.

The GRCBox management application was used to configure the required rules on the GCMs.

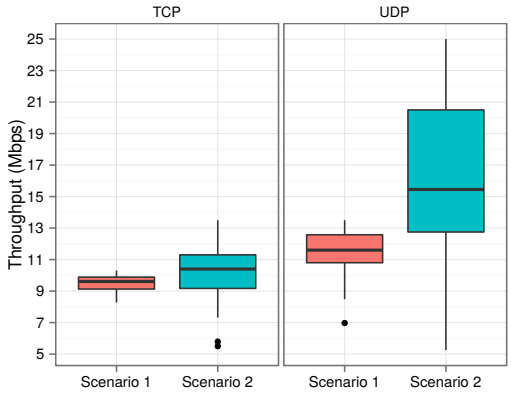
a) Maximum Throughput: To evaluate the impact of the GRCBox Architecture on the maximum throughput experienced by a client-server connection when using the GRCBox, we tested the network performance using the *iperf* [22] tool. We have collected measurements for both UDP and TCP protocols. Each experiment was repeated 60 times to discard random effects, and the role of the user devices was interchanged after half of the experiments to discard the effects associated to the device's performance. Results are shown in Figure 6a in a boxplot chart.

Notice that, no matter whether TCP or UDP is used, the maximum throughput achieved when using the GRCBox Architecture (Scenario 2) is slightly better than the one achieved when both devices are connected to the same WiFi Network (Scenario 1). The main cause behind this difference is the use of a different channel for each wireless network when using the GRCBox. This setup avoids collisions between nodes, when transmitting requests and responses. The high variability

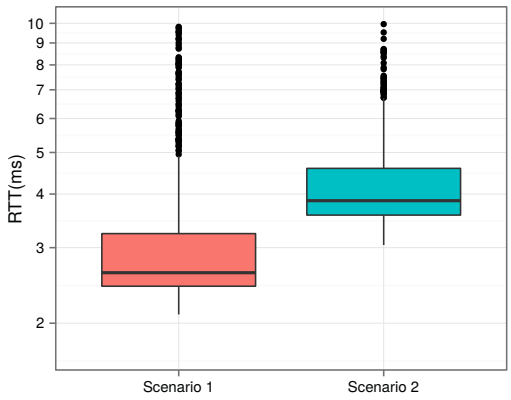


(a) Scenario 1 (b) Scenario 2

Fig. 5: Scenarios used in our experiments.



(a) Throughput obtained for UDP and TCP tests.



(b) UDP RTT results.

Fig. 6: Direct VANET communication results.

TABLE I: Devices Configuration

Element	Characteristics
Tablet	Google Nexus 7 (2012)
Smartphone	BQ Aquaris 4.5E
GCMs	RaspberryPi 2 Model B
Ad-Hoc Network	802.11a, Frequency:5.18 GHz
WiFi 1	802.11g, Frequency:2.462 GHz
WiFi 2	802.11g, Frequency:2.412 GHz

experienced in all the experiments is due to the presence of interference, which heavily affects throughput in wireless networks.

b) UDP Round Trip Time (RTT): To test the delay introduced by the GRCBox Architecture when comparing against an infrastructure network we have developed a small application that sends an UDP messages to a server running on another device. The server will then send a new UDP message as a response, so the RTT can be measured at the first sender. We performed the test on both scenarios presented before, collecting more than 500 measurements per scenario. Figure 6b shows a boxplot that summarises the results we obtained. We have used a logarithmic scale to be able to clearly represent infrequent values in both the low and high ranges. It can be observed that, on average, the RTT is about 2 ms higher when using GRCBox. This effect is due to the multi-hop nature of the communication: adding an extra hop between sender and receiver increases the RTT. During this experiment we discovered that the main source of delay in Android devices is the WiFi interface power management performed by the Android operating system, which in some cases increased the RTT by up to 508 ms. If we compare the delay introduced by the GRCBox against the delay introduce by the OS, we can conclude that the GRCBox impact on the RTT is negligible.

Fig. 7: Testing EYES in a real scenario.

VI. Testing EYES

To test EYES we have run on-road tests with real hardware. Those test allowed us to evaluate the user experience[2]. Figure 7 shows a picture from one of our outside test with a GRCBox and a tablet running EYES attached to the windshield. After interviewing the drivers, we noticed several problems: the users pointed out that the video sometimes froze and that, on the video, it was hard to identify the cars coming in counter direction. Moreover, we also noticed that the sight of the driver was pointing at the screen of the tablet for long periods, distracting them from the road, which increases the risk of an accident.

[2]Video available on youtube(Sp):https://youtu.be/eUQfalxPK0o

We are working on solving the problems described previously. To solve the problems related to video quality we are exploring different video codec-framerate-resolution combinations. Concerning the potential risk of driver distraction, we are establishing contacts with in-car-user-interface specialists. We envision different ways to warn the driver, like audio signals or image recognition to highlight the presence of vehicles coming in counter direction.

VII. Conclusions

In this paper we presented the GRCBox architecture and EYES. The GRCBox is an architecture that allows smartphone application developers to implement applications that exploit inter-vehicle device-to-device communications. The GRCBox is based on a cheap and low power computer including basic Linux networking. EYES is an application that runs on the top of the GRCBox architecture to stream video from a vehicle to another overtaking vehicle. Our results in section V showed that, for device-to-device communication, the performance of the GRCBox architecture is comparable to an infrastructure network, and that it has a negligible impact. The results in section VI showed the viability of EYES and also pointed out problems related to in-car user interfaces and security.

Acknowledgments

This work was partially supported by the *European Commission* under *Svāgata.eu*, the Erasmus Mundus Programme, Action 2 (EMA2), the *Ministerio de Economía y Competitividad, Programa Estatal de Investigación, Desarrollo e Innovación Orientada a los Retos de la Sociedad, Proyectos I+D+I 2014*, Spain, under Grants TEC2014-52690-R and BES-2012-052673.

References

[1] M. Gerla and L. Kleinrock. “Vehicular networks and the future of the mobile internet”. In: *Computer Networks* 55.2 (Feb. 2011), pp. 457–469. ISSN: 13891286. DOI: 10.1016/j.comnet.2010.10.015.

[2] K. C. Lee and M. Gerla. “Opportunistic vehicular routing”. In: *Wireless Conference (EW), 2010 European*. IEEE, Apr. 2010, pp. 873–880. ISBN: 978-1-4244-5999-5. DOI: 10.1109/EW.2010.5483530.

[3] J. Santa and A. Gomez-Skarmeta. “Sharing context-aware road and safety information”. In: *Pervasive Computing, IEEE* (2009), pp. 58–65.

[4] O. Trullols-Cruces et al. “Cooperative Download in Vehicular Environments”. In: *IEEE Transactions on Mobile Computing* 11.4 (Apr. 2012), pp. 663–678. ISSN: 1536-1233. DOI: 10.1109/TMC.2011.100.

[5] R. T. Fielding. “Architectural styles and the design of network-based software architectures”. PhD thesis. University of California, 2000.

[6] Car Connectivity Consortium (CCC). *MirrorLink*. http://www.mirrorlink.com/. Feb. 2015.

[7] Google Inc. *Android Auto*. http://www.android.com/auto/. Feb. 2015.

[8] Apple Inc. *CarPlay*. https://www.apple.com/ios/carplay/. June 2014.

[9] J. Whipple, W. Arensman, and M. S. Boler. “A public safety application of GPS-enabled smartphones and the android operating system”. In: *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*. IEEE. 2009, pp. 2059–2061.

[10] J. Yang, J. Wang, and B. Liu. “An intersection collision warning system using Wi-Fi smartphones in VANET”. In: *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*. IEEE. 2011, pp. 1–5.

[11] S. Diewald et al. “DriveAssist-A V2X-Based Driver Assistance System for Android.” In: *Mensch & Computer Workshopband*. 2012, pp. 373–380.

[12] S. M. Tornell et al. “Implementing and testing a driving safety application for smartphones based on the eMDR protocol”. In: *Wireless Days (WD), 2012 IFIP*. IEEE. 2012, pp. 1–3.

[13] J. Zaldivar et al. “Providing accident detection in vehicular networks through OBD-II devices and Android-based smartphones”. In: *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*. IEEE. 2011, pp. 813–819.

[14] J. Wideberg, P. Luque, and D. Mantaras. “A smartphone application to extract safety and environmental related information from the OBD-II interface of a car”. In: *International Journal of Vehicle Systems Modelling and Testing* 7.1 (2012), pp. 1–11.

[15] I. O. for Standardization. “ISO 14230-1:1999: Road vehicles, Diagnostic systems, Keyword Protocol 2000”. In: 1999.

[16] E. Koukoumidis, M. Martonosi, and L.-S. Peh. “Leveraging smartphone cameras for collaborative road advisories”. In: *Mobile Computing, IEEE Transactions on* 11.5 (2012), pp. 707–723.

[17] C.-W. You et al. “Carsafe app: Alerting drowsy and distracted drivers using dual cameras on smartphones”. In: *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. ACM. 2013, pp. 13–26.

[18] C. Olaverri-Monreal et al. “The See-Through System: A VANET-enabled assistant for overtaking maneuvers”. In: *Intelligent Vehicles Symposium (IV), 2010 IEEE*. June 2010, pp. 123–128. DOI: 10.1109/IVS.2010.5548020.

[19] J. Louvel, T. Templier, and T. Boileau. *Restlet in Action: Developing RESTful Web APIs in Java*. Greenwich, CT, USA: Manning Publications Co., 2012. ISBN: 193518234X, 9781935182344.

[20] M. Thompson and P. Green. *Raspbian Home Page*. http://http://www.raspbian.org/. June 2014.

[21] S. M. Tornell et al. “GRCBox : Extending Smartphone Connectivity in Vehicular Networks”. In: *International Journal of Distributed Sensor Networks* (2014), Article ID 478064.

[22] *Iperf homepage*. https://iperf.fr/. Feb. 2015.