On the feasibility of a context-aware application development platform for VANETs

Rui Costa
Instituto Superior Técnico
Av. Prof. Dr. Anibal Cavaco Silva
2744-016 Porto Salvo, Portugal
Email: ruicosta@ieee.org

Teresa Vazão
Instituto Superior Técnico
Av. Prof. Dr. Anibal Cavaco Silva
2744-016 Porto Salvo, Portugal
Email: teresa.vazao@ist.utl.pt

***Abstract*—The development of vehicular communication technologies and networks allowed the further development of Intelligent Transportation Systems (ITS). Such systems are heavily based in the creation of applications and services that will be design to serve multiple purposes, either for delivering more pleasant driving experience to users and also for enhancing the safety and security of road users. In this paper we propose a platform for facilitating the agnostic development of vehicular-environment oriented applications by offering a common API for both safety and comfort-oriented applications. Our solution features application and service's flow control in order to give priority to most relevant applications in every moment, supporting scenarios such as Normal, Emergency and Accident. A protocol for disseminating vehicle safety status is proposed in order to determine the current execution context. Results from a field-tests in controlled environment shown reduction in non-relevant traffic volume in safety situations while reduction of relevant service's response time.**

I. Introduction

Intelligent Transportation Systems (ITS) had its inception and expansion based on the development of several communication technologies and protocols of vehicular networks. With such, the number of possible services and applications that could be used in those systems grown exponentially, opening space for market opportunities but also for the enhancement and creation of services that have as mission the overall improvement of road users safety. Several standards and technologies were developed and designed having in mind those two realities, one concerned with offering a more pleasant driving experience to the user and the other directed to augment its safety and security while circulating on all kinds of roads. Besides the two categories of applications that can be developed for ITS, more challenges arise with the constant change of road topology and possible safety-related situations that can happen, such as accidents or emergency scenarios.

The proposed work aims to unite the aforementioned two application paradigms in one application development middleware that offers common functions to both kinds of applications while maintaining awareness regarding the road situations that may affect how the driver, the user or even applications should behave.

Our solution helps accelerating the process of application's development by abstracting those of the several implementation details taken by the underlying support platforms and technologies. While offering a common API to applications, this platform can control application's flow and therefore decide which applications have priority, based upon several criteria, for all kinds os scenarios that can happen in vehicular environments, such as normal circulation, traffic, emergency or accidents.

Therefore, the major contributions of the proposed work are:

1) API for agnostic development of applications to be used in vehicular networks, regardless of its type.
2) Application coordination and priority-based scheduling in emergency situations.

This paper presents a solution that is heavily based in the standards for the European ITS communication architecture, focusing on using and grouping and extending, rather than changing, the recommended functionalities towards meeting our goals.

The rest of this document is organized as follows. Section II will depict a brief introduction to ITS, the available standards and main contributors and a classification of the applications that can be created to this systems. Section III details the proposed architecture that is then subject to a experimental evaluation performed in a real controlled scenario whose results are presented in Section IV. Conclusions and future work notes are depicted in Section V.

II. Intelligent Transportation Systems

As a subset of traditional Mobile Ad-hoc Networks (MANETs), Vehicular Ad-hoc Networks (VANETs) have specific properties that distinguish those from the traditional MANETs[1], for instance the deployment scenario the mobility pattern, properties of the communication nodes as well as some other challenging characteristics as the network scale, network topology variation or even the degree of connectivity [2]. With the work performed in enhancing VANETs the inception of ITS architectures, technologies and protocols was finally possible [3].

In order to grant that ITS communication technologies and systems are disseminated throughout all the road infrastructure, standardization became a mandatory path to unveil. Several initiatives from both academia and industry started developing work in order to conglomerate technologies, applications, protocols and security and management mechanisms into one widely accepted architecture. A European project, named COMeSafety[1], was responsible for consolidating, gathering and fostering the several contributions from the several initiatives [4] for later proposal to relevant standardization bodies such as Internet Engineering Task Force (IETF), International Standards Organization (ISO) and IEEE.

With that work, a proposal of a standardized ITS architecture, ETSI EN302665[5] was drafted. That architecture takes into account four (4) different components on the communication infrastructure. Vehicles, mobile consumer devices as Personal Stations,Roadside Unit (RSU)s, representing a fixed installation along the road with communication capabilities and sensors and Central Stations that can supply some value-added applications and services.

Two (2) types of communication scenarios were conceived: Vehicle-to-Vehicle (V2V), where vehicles communicate among themselves through a Ad-hoc network, and Vehicle-to-Infrastructure (V2I), for accessing services for exchanging relevant information with the RSUs.

Despite the fact that the standard was approved, most current initiatives focus in either creating adaptations of the standard, by adding architectural changes, towards meeting their needs or only focus in safety-related applications.

Regarding communications, Wireless Access in Vehicular Environments (WAVE)[6] and IEEE 802.11p[7] standard, unveiled the possibility of creating communication capabilities adapted demands of the vehicular communication environment. One important characteristic of the 802.11p standard is that it makes available two different channels of wireless communication, one for relevant prioritized information, later used for safety-related applications, and other for standard, non-priority traffic.

With the European ITS communication architecture a new layer, appeared[8]. The Facilities Layer aims to provide functionality for the rapid application development while ensuring that each application running in the same station is using the same data and information.

This layer, poses an important role in defining an architecture for applications development systems for vehicular environments, to be addressed in the work presented on this document.

A. Application Types

Within ITS, several applications and services can be developed. Traditional applications may be used, but generally vehicular oriented applications tend to be the option taken by developers working with ITS applications and services. Applications and services can be classified in several different fashions[9][10], both most of those take into account the objective, priority and technical requirements for those applications.

Two (2) major categories can be defined for applications and services characterization within vehicular-communication environments:

a) Safety-related: intended to prevent accidents by warning and assisting drivers about traffic or road conditions. Safety applications can for instance provide information such as road conditions, weather information, traffic lights scheduling and speed limits as well as some traffic coordination and monitoring. Such applications can even provide collision notification or support for emergency management purposes.

b) User-Confort: intended to provide a more pleasant travel experience to the driver or other individuals traveling. This applications traditionally depict applications usual to be used in the traditional TCP/IP model. For instance, these applications can provide information about points of interest such as restaurants museums and other local information. Applications with high network requirements such as video and audio streaming, web browsing, e-mail access, voice and instant messaging may also be developed.

By having this two, quite distinct application types, different approaches should be taken due to the different application's requirements of each kind. Safety-related applications consist mainly of time-critical requirements while for instance delay sensitive services such as multimedia data transfer or instant messaging should also be considered when defining an architecture for application's development in vehicular communication scenarios.

Several assessments regarding both application types requirements have been performed in the literature [10][11][12]. That work served as basis for the work presented in this paper, defining the relevant features and services that one architecture must follow to cope with both comfort and safety-related applications needs.

III. Vehicular-Application Development Middleware

In order to offer a Application Programming Interface (API) that offered applications relevant yet common functions to be used while maintaining those agnostic from the underlying execution environment we developed the Vehicular-Applications Development Middleware (VADM). This platform had in mind the following requirements:

1) Low-cost and modular platform;
2) API for agnostic services and applications development and support;
3) context-aware configurable platform.

To cope with a low-cost and modular platform, the authors developed a general architecture for both roadside units and vehicles, that can be used with of-the-self embedded systems and communication technologies.

The developed API, extends and uses the ITS communication standard, through combination of a set of functionalities of the Facilities Layer. Also, in order to grant compatibility with other solutions for vehicular environments, the proposed solution makes use of the SAE J2735 messaging standard[13]. This standard defines message's format and structure to be used in the IEEE 802.11p wireless vehicular-communication standard.

The developed architecture has de possibility to adapt itself to the scenario in which the vehicle is at the moment, and therefore adapt the applications and service's flow to give priority to the most relevant in such situations. The developed platform has an internal state that mirrors the above mentioned scenarios, representing the vehicle status at a given time.

We considered three (3) possible states:

- **Normal -** Vehicle is traveling normally;
- **Emergency -** Vehicle declared an emergency situation such as, flat tyre, engine damage, emergency driving or irregular driving;
- **Accident -** Vehicle declares as being in an accident or eminent accident;

Every time a vehicle is in a *Emergency* or *Accident* situation that also affects the surrounding vehicles and road participants. A status dissemination protocol is also proposed in this work and is further described in III-C.

A. Solution Architecture Design

Figure 1 depicts the overall architecture of the proposed solution. Our architecture poses as a middleware between the applications and the physical world, that by following the principles of the Facilities Layer, gathers information from sensors, communication technologies and all the available resources in order to offer such information to applications in a parsed, organized and relevant manner.

Therefore, a set of decoupled modules was developed in order to offer several functionalities within the VADM platform to applications. Modules can be updated separately in order to enhance their functionality, while keeping the overall functionality of the VADM. Bellow follows a brief description of each module:

- **Application Request Module (ARM)-** Management of all the underlying modules by orchestrating interactions among them and also acting as a coordinator of the overall inner-VADM architecture. All communication is made through the ARM that contains a scheduler to define the relative priority of each message being sent based on the application type and the current state of the platform. Case a application does not have priority to access the platform, the scheduler blocks the request. Several policies can be configured, e.g. discard or queuing of the request.
- **Service -** Several system services were identified as common requirements for applications that execute in vehicular environments and therefore are made available as default. Services such as retrieving the current coordinates, speed or any other available vehicle information (*GetNodeInformation*), as well as getting the current platform status (*Normal*, *Emergency* or *Accident*) (*GetPlatformStatus*) or getting the current type of node (RSU or vehicle) (*GetNodeType*). Also there is a service that provides the current neighbors list and retrieves information of those, such as the address or last known position (*GetNeighboursInfo*). Some system support services, that can only be accessed by the platform and its protocols, were also developed. With this already available services it is possible to grant basic functionalities for applications, that then can compose several services or create and install their owns in order to cope with the application's needs.
- **Application Registry Bank (ARB)-** Storage of registered application's information.
- **System Analyzer -** Alter the current execution state of the VADM platform (*Normal*, *Emergency*, *Accident*). VADM platform state can be altered in three (3) different fashions: through analytical methods[14], that determine the need of altering the state based on data analysis gathered from several sources such as in-vehicle sensors, navigation data, modules malfunctioning or any other relevant information. Also the state of the platform can be altered through user-input within a human-machine interface. Besides the aforementioned methods for changing the platform state, state can be also altered by proximity to vehicles in accident situation using the protocol described in III-C, the Epidemic Status Plausibility Dissemination (ESPD).

In order to maximize compatibility to whatever support platform or architecture supplies the communications technologies and sensors,while keeping the platform overall functionality, only two modules have access to the underlying facilities support platform:

- **Call Manager -** Storage of parsed and updated information retrieved from several sources such as sensors, navigation and data analysis.
- **Communication Manager -** Establish connection with all the available communication facilities to send and receive information and act as scheduler between several of the communication technologies. In case several communication technologies are available, Communication Manager can use its scheduler to decide upon sending or retrieving information from one or several communication technologies based on several possible policies. One example of a policy, is in case of accident situation, information can be send through both Wi-Fi and cellular technologies.

B. Application Development API

The API proposed in this paper has several primitives that can be used by applications regardless of their types of requirements. When a application starts, it should being its setup procedure by using the following primitives in such order:

1) **Register Application -** Register the application within the platform in order to use its functionalities.
2) **Retrieve Node Information -** Get information regarding the current node characteristics, such as node type (Vehicle/RSU), license plate, current IP, platform version.
3) **Request Service List -** Get list of available services currently active in the platform.
4) **Subscribe to Platform Services -** Subscribe to the services available at the platform that will be needed by the application.

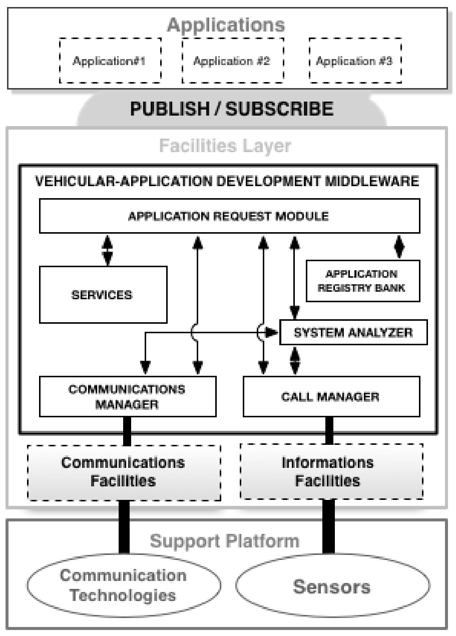


Fig. 1. VADM overall architecture and its interaction with the European ITS Communication layered standard

5) **Deploy and Run Application-Service -** Deploy and run a service specific of the application. This service can be a new implementation or a composition of other services.
6) **Publish Information -** Publish information and data to the platform that will send it through the available communication resources using TCP or UDP transport protocols with the available communication technologies.
7) **Unregister Application -** Unregister application from the platform, remove all application deployed services and unsubscribe to all subscribed platform services.

Applications interact with the platform in a decoupled fashion, by sending information through the above referred function, and retrieving all the information, from the own platform or from other vehicles or nodes in a Publish/Subscribe fashion. Applications can subscribe to services installed in the platform or depoy, run and subscribe their specific services. Those services can result by the composition of other services or simple updates to the existing ones. Applications are default listeners of a service that publishes all the information received from other nodes of the network using the same application. Therefore it is assured the communication of the same application in different nodes, while keeping the decoupled event-driven paradigm of the indirection between applications and the platform.

C. Epidemic Status Plausibility Dissemination Protocol

Whenever a vehicle is in an *Accident* or *Emergency* situation, that also affects the surrounding vehicles and individuals. Therefore, the authors developed a protocol for disseminating the state of a vehicle within a region of relevance, henceforth referred as Region of Plausibility (ROP), called ESPD. The criteria used for determining that region can be varied. The ESPD protocol functions in a multi-hop fashion where each node determines the relevance of changing its own VADM platform state. When a vehicle is in a *Accident* or *Emergency* situation, it triggers the sending of beacons to other vehicles, sending its position, the emergency state and other variable information. Each vehicle, that receives that beacon determines if it is within the ROP and also retransmits that information to the surrounding nodes. ESPD makes use of the safety channel of the 802.11p standard and is defined using the Basic Safety Message (BSM) of the J2735 standard.

IV. Experimental Evaluation

A set of applications were created in order to be used in a controlled scenario where a vehicle has an accident and triggers the usage of a application that will request for Emergency Vehicles dispatch to its location. The response time of the request for assistance application was used as benchmark in order to determine the effect of using several other applications, with different characteristics, at the same time.

A. Developed Applications

Five (5) applications were developed, each one with different behaviors and network requirements:

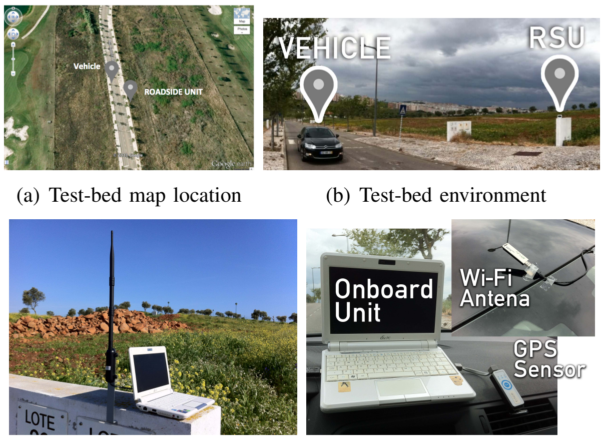
1) **RequestAssistance -** Application that every time a vehicle set its status to *Accident*, triggers a broadcast request to the nearby RSU in order to have emergency vehicles sent to its location.
2) **VehicleStream -** A vehicle streams video in a broadcast fashion to the RSU nearby. RSU answers with an ACK for every video packet received. Video is streamed at 800 kbps.
3) **Chat -** Provides bidirectional text communication with the surrounding vehicles through the RSU.
4) **Points of Interest (POI) -** The RSU broadcasts information regarding the nearby POI (Restaurants, Hotels and Gas Station locations) and other relevant local information, such as the maximum speed limit on that road. POI messages are broadcasted at 1 Hz. Creates a service when on a RSU to pinpoint the distance to the nearby hotels and restaurants through a remote database call.
5) **RSUStream -** RSU streams video to all the surrounding vehicles in a broadcast fashion at 480 kbps.

Table I shows the default services available on the platform that are used by the developed applications. By analyzing this table it is possible to understand that although applications have different purposes, they end up requiring fairly the same services when being developed. When the needed services are not available they can create their own services, or compose existing ones.

The aforementioned applications cover situations in which the vehicle is the main source of both high and medium traffic volume generation and also situations where the RSU, not the

TABLE I
PLATFORM SERVICES USED BY THE DEVELOPED APPLICATIONS

	GetNodeInformation	**GetPlatformStatus**	**GetNodeType**	**GetNeighboursInfo**	**Creates Own Service**
RequestAssistance	Yes	Yes	No	No	No
VehicleStream	No	Yes	No	No	No
Chat	No	Yes	Yes	No	No
POI	Yes	No	Yes	Yes	Yes
RSUStream	Yes	No	Yes	No	No



(a) Test-bed map location (b) Test-bed environment

(c) RSU node equipment (d) Vehicle node equipment

Fig. 2. Test-bed components, location and overall position of the communication nodes

station in need of using the safety application, is the most relevant traffic volume generator of the network.

B. Tests Description

These applications were tested together in a scenario with one vehicle and one RSU. The controlled scenario situation consisted of a 30 seconds experience where at second 15 the vehicle has an accident and triggers the *RequestAssistance* application. Figure 2 depicts the test-bed components location and position using in this controlled scenario testing procedure. On the vehicle was used a SMCWUSBS-N3 802.11b/g/n wireless pen, while for the RSU was used a high-gain PowerLink Ultra High Power, both running IEEE 802.11n and with chipset chipset rt3070. Both stations, vehicle and roadside consist of ASUS eeePC equipment with Intel Atom running at 1600 MHz and 2 GB of RAM.

C. Discussion of Results

The developed applications were combined and executed in the controlled scenario 10 times. To evaluate the overall performance of the proposed solution, 2 different metrics were taken into account, the response time of the *RequestAssistance* application and also the generated and received traffic volume. *ResquestAssistance* response time was used in order to determine how relative priority scheduling can affect safety-related services performance. The traffic volume was used as metric due to being a direct consequence of non-relevant applications being discarded in emergency or accident situations, since generated and received traffic volume reduction could have impact on the safety-application's response time.

Two different scenarios were considered: first in which no scheduling regarding the send and receiving of information is performed by both RSU and vehicle; second where based on the status of the platform a scheduling between traffic from safety and comfort applications is performed. For this evaluation the scheduling decision for not priority traffic adopted was full discard without any caching of content. Also, the ESPD protocol is active in both situations, meaning that when in the accident situation, RSU adapts its inner status to be compliant with the vehicle's status.

Figure 3 depicts the response time of the *RequestAssistance* application when in used with several other applications covering situations in where the vehicle in accident is the main generator of traffic in the networks and also the situation where the RSU is the most relevant generator of traffic in the network.

Table II depicts the average traffic volume for all the combined application's sets used in the controlled scenario. The node used as reference is the vehicle.

The results regarding the traffic volume generated and received by the applications in the several situations shown that, by using the scheduling based on the VADM platform status, it is possible to reduce the traffic in the network, more emphasized in high demanding applications such as the video streaming. The resulting reduction in the generated traffic is mostly due to the discard of information produced by comfort applications that in a *Accident* or *Emergency* scenario, have less priority in access the platform and the available means.

By analyzing the results regarding the response time of the *RequestAssistance* application in the various situations, one can conclude that, the service response time is only enhanced by using scheduling when the node in need of using such safety service, is also the node generating the most considerable amount of traffic in the network, therefore able to control the generation of such information. In the remaining situations, the overall response time of the application increased due to wireless channel being occupied with applications with traffic volume generation, and therefore, the information to start the scheduling procedures never reaching the other node. Also it is possible to conclude that applications that have low requirements regarding the bandwidth, have a response time similar to the response time used as benchmark due to its low traffic volume generation.

V. CONCLUSION AND FUTURE WORK

The work presented in this paper offers a fast methodology for applications and service's development to be used in vehicular communication scenarios to facilitate the creation and

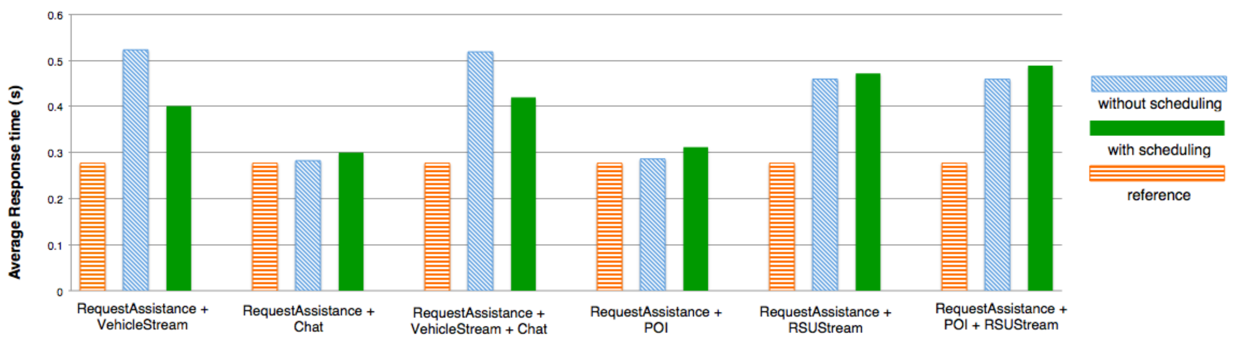


Fig. 3. RequestAssistance average response time comparison with several application's sets

TABLE II
TRAFFIC VOLUME GENERATED AND RECEIVED BY THE SEVERAL APPLICATIONS REDUCTION BY USING VADM APPLICATION LEVEL SCHEDULING

Applications	**No Scheduling (KBps)**		**With Scheduling (KBps)**		**Difference**	
	Generated	**Received**	**Generated**	**Received**	**Generated**	**Received**
RequestAssistance + VehicleStream	172.719	7.878	62.582	7.682	-63.767 %	-2.492 %
RequestAssistance + Chat	0.024	0.010	0.010	0.009	-61.114 %	-9.176 %
RequestAssistance + VehicleStream + Chat	110.678	5.043	40.215	5.026	-63.665 %	-0.334 %
RequestAssistance + POI	0.003	0.055	0.002	0.044	-9.191 %	-20.474 %
RequestAssistance + RSUStream	7.582	66.089	7.217	64.354	-4.809 %	-2.625 %
RequestAssistance + RSUStream + POI	5.041	45.338	4.571	38.693	-4.809 %	-2.625 %

expansion of Intelligent Transportation Systems. The VADM platform offers a set of simple functions and functionalities that allow applications to be developed in such a way that the developers do not need to know advanced details about the platform in which applications and services are being executed. Also our platform permits the development of context-aware applications without incorporating complex context-detection mechanisms in application's design and structure. Applications flow control and priority mechanisms are maintained and coordinated by the VADM platform, that by having a modular design approach can be improved in such a way that several functionalities can be added in the future without compromising the overall functioning of both applications and the platform itself.

Field-test results shown that, by passing the application's flow control to the VADM platform, besides making easy the application development, improvements in the generated traffic to the network and services response time in safety-critical situations were achieved.

The next steps towards evaluating the proposed solution is by adding complexity to the field-test scenarios, either by increasing he number of communicating vehicles as well as by creating complex traffic safety-critical situations with even more applications executing and then evaluate the scalability of our platform and developed protocols. Also, adding one more wifi antenna to each node in order to evaluate the use of separate channels for safety and comfort applications will be addressed on the next experimental evaluation iterations.

REFERENCES

[1] A. Dahiya and R. K. Chauhan, "A Comparative study of MANET and VANET Environment," *DBMS*, vol. 2, no. 7, pp. 87–92, 2010.

[2] M. Kafsi, P. Papadimitratos, O. Dousse, T. Alpcan, and J.-p. Hubaux, "VANET Connectivity Analysis," *IEEE Workshop on Automotive Networking and Applications*, 2008.

[3] J. Jakubiak and Y. Koucheryavy, "State of the Art and Research Challenges for VANETs," *5th IEEE Consumer Communications and Networking Conference*, pp. 912–916, 2008.

[4] T. Kosch, I. Kulp, M. Bechler, M. Strassberger, B. Weyl, and B. M. W. Group, "Communication Architecture for Cooperative Systems in Europe," *IEEE Communications Magazine*, no. May, 2009.

[5] ETSI EN 302 665, "Intelligent Transport Systems (ITS); Communications Architecture," pp. 1–44, 2010.

[6] R. A. Uzcátegui and G. Acosta-Marum, "WAVE : A Tutorial," no. May, pp. 126–133, 2009.

[7] D. Jiang and L. Delgrossi, "IEEE 802 . 11p : Towards an International Standard for Wireless Access in Vehicular Environments," pp. 2036–2040, 2008.

[8] COMeSafety, "The European Communications Architecture for Co-operative Systems A Key Enabler for the Development and," 2009.

[9] L. S. MIHAIL and M. KIHL, "Inter-vehicle Communication Systems: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 111–111, Jun. 2008.

[10] Y. Toor, P. Mühlethaler, A. Laouiti, and A. de La Fortelle, "Vehicle Ad Hoc Networks: Applications and Related Technical Issues," pp. 74–88, 2008.

[11] H. Hartenstein and K. Labertaux, "VANET - Vehicular Applications and Inter-Networking Technologies," 2010.

[12] K. Dar, M. Bakhouya, J. Gaber, and M. Wack, "Wireless Communication Technologies for ITS Applications," *IEEE Communications Magazine*, no. May, pp. 156–162, 2010.

[13] D. Kelley, "DSRC Implementation Guide A guide to users of SAE J2735 message sets over DSRC."

[14] C. Oh, E. Jeong, K. Kang, and Y. Kang, "Hazardous Driving Event Detection and Analysis System in Vehicular Networks (HEAVEN): Methodology and Field Implementation," vol. TRB 2013 A, pp. 1–18, 2013.