TagusVanet: A low-cost, integrated framework for development, validation and deployment of VANETs

João Duarte Gomes, Ricardo Chaves, Teresa Vazão
Instituto Superior Técnico
Av. Cavaco Silva, Taguspark, Oeiras, Portugal
Email: {joao.d.gomes, ricardo.chaves, teresa.vazao}@tecnico.ulisboa.pt

***Abstract*—Vehicular Ad-Hoc Networks are an emerging mobility paradigm focussed on improving road safety and traffic efficiency. However the current available solutions are very expensive and developing applications for this environment can be a daunting task. Also given the number of different manufacturers and the number of vehicles, deploying and validating these applications can be extremely difficult.**

In this work, the issues of developing, validating and deploying applications are addressed. A low-cost platform, which uses commercial, off-the-shelf hardware is used. This platform supports a software architecture which is designed to address heterogenous application support and validation. Applications are supported by using a policy-based approach which allows the platform to be configured, on a per-application base, in a simple way. Integrated with this approach, is a simple API which allows developers to focus on their application, not on how their application's requirements are met. To complete this solution, several tools are made available to allow for an easy validation of the applications, both in field and lab settings through emulation of field conditions. Testing is done to assure the emulator's behaviour is correct and a proof of concept application shows the capabilities of the proposed solution.

***Keywords*—TagusVanet, low-cost platform, policy-based approach, integrated framework, vanet deployment, vanet validation**

I. Introduction

The advances of wireless communications and embedded systems lead to the emergence of Vehicular Ad-Hoc Network (VANET), which is seen nowadays as a mean to increase road safety and travel efficiency. This new type of networks introduces new challenges that have driven the attention of the research community and standardisation bodies, as surveyed in [1], [2]. Also new business opportunities have been identified by auto manufacturers, road operators, fleet transportation and software development companies.

The currently foreseeable VANET applications can be divided into three main categories: safety-oriented, convenience (or traffic management) and commercial [3]. Safety-oriented applications focus on assisting the driver in handling potentially dangerous situations by actively monitoring the environment and listening to the messages being exchanged between nodes. Convenience applications enable a more efficient traffic flow and increased vehicle throughput, by sharing information with the road infrastructure, centralised traffic control systems and even other vehicles. Finally, commercial applications focus on providing the driver with ways of improving satisfaction, entertainment and productivity.

These new types of applications demand for new communications paradigms due to the specific requirements they have: safety-oriented applications need to guarantee the dissemination of time-critical information in a very short period of time; in traffic management applications, each node broadcasts information gathered from the vehicle sensors, to all the neighbour nodes, in a timely accurate way; and finally, commercial applications need to deal with the transport, routing and location problems that arise in VANET due to the nodes' mobility pattern and scenario characteristics. As all these problems are very challenging, a significant research activity is been driven to solve them. Therefore, a wide range of routing protocols [4], location services [5], efficient data dissemination strategies [6] and medium access control mechanisms [7] have been proposed.

Standardisation efforts are being conducted in the Institute of Electrical and Electronics Engineers (IEEE), International Standard Organization (ISO) and European Telecom Standardization Institute (ETSI). These efforts lead to the design of a new protocol stack specifically targeted to support safety-oriented applications, the Wireless Access in Vehicular Environments (WAVE). The IEEE also proposed a new 802.11-based standard, which is able to cope with the specificities of VANET: the very short connection time that may arise in several mobility conditions and the existence of multiple applications with different priorities that need to share the wireless medium. Nevertheless, they are is still in the early stage of development and several testbeds have been designed with the existing IP protocol stack and available wireless technologies to validate the concept [8], [9], [10]. Aside from the design cost, one of the main problems in building a demonstrator is the difficulty of assessing the physical conditions of it's operation. This is mainly related with signal propagation and node mobility. Hence, field tests are quite expensive and impossible to repeat in similar conditions and, due to this, there is a huge overhead associated with process of design and developing new applications and protocols.

Our work aims to solve this problem, by providing a VANET platform that offers an integrated framework for development and validation of applications. In order to facilitate the development of such applications, standard APIs are used and a set of policies that cover the most important requirements of a wide set of applications are implemented. Applications can be validated by emulating real VANET behaviour, through the use of adequate propagation and mobility models, and by using traffic analysis.

II. Related work and Background

A. VANET Communication

The data produced by applications in a VANET can be transmitted in many different ways depending on the addressing scheme used [11]. Current networks use unicast, anycast, multicast and broadcast communication modes.

However, these addressing modes are unable to fulfil the needs of all the emergent VANET applications, as a significant set of them requires the dissemination of information within a given region of interest. For this type of applications a new communication mode is needed: the geocast communication [12]. In geocast communication, data is disseminated within a certain area and, for this, nodes need to be identified by their position, instead of by their IP addresses.

The use of geocast communication may cause a situation, usually known as Broadcast-Storm [13], where all nodes in the destination region rebroadcast the message at the same time. This leads to a state of frequent contention and increased packet collisions, causing an increase in backoff timers. To address this issue, a probability based scheme was developed [13], where each node computes a rebroadcast probability using Formula 1, on a per-packet basis, where D_{ij} represents the relative distance between node *i*, the current node, and *j*, the node that last broadcasted that packet, and *R* the average transmission range.

$$p_{ij} = \frac{D_{ij}}{R} \tag{1}$$

This way it's ensured that the nodes closer to the Source are less likely to rebroadcast the message when neighbours farther away have already done it.

As stated previously, VANET-oriented applications have multiple requirements, regarding addressing and message lifetime.

B. VANET standards

As stated previously WAVE is a set of standards proposed by IEEE, composed by the IEEE 802.11p and IEEE 1609.1 through 1609.4 standards. These standards have the purpose of easing wireless access in vehicular environments, and their reference architecture is presented in Figure 1.

802.11pDefines the PHY layer and MAC sublayer, improved for vehicular environments.

1609.1 Describes the resource manager, which is in charge of managing access to the system's communications resources.

1609.2 Defines the security services associated with the network stack defined by the other 1609.x standards.

1609.3 Specifies the LLC sublayer, Network and Transport layers and the WAVE Short Message Protocol (WSMP).

1609.4 Extends the 802.11p MAC sublayer to support multichannel operation and service priority management.

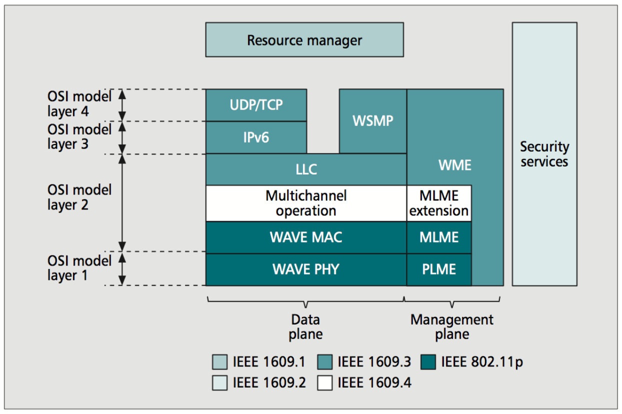

Fig. 1: WAVE communication stack indicating the standard that covers each set of layers. [14]

In order to be used with the WSMP, an application message set was defined, the SAE International (SAE) J2735 standard. In this standard a set of messages are defined, which are intended to support most VANET-oriented applications [15]. This message set is composed of a number of *Message Types*, being each *Message Type* composed by *Data Elements* or *Data Frames*:

1) *Data Elements* are the most basic type of information defined in this standard, for example, the vehicle's heading or latitude. They convey only a single type of information and in the standard there are over 150 *Data Elements* defined.
2) *Data Frames* are composed of one or more *Data Elements* or *Data Frames*, for example a vehicle's 3D location is composed of its latitude. longitude and altitude above sea level. They have the purpose of relaying more complex pieces of information and more than 70 are defined in the standard.

Among the messages defined by the standard are a Basic Safety Message (BSM). Used to carry vehicle-state information for safety-oriented application support, carrying information like the vehicle's coordinates, current speed and vehicle size. Another message us the A La Carte Message (ALCM), a generic message with flexible content chosen by applications according to their needs.

In this standard, all the elements are defined in the formal language ASN.1 [16]. For the over-the-air translation, the DER encoding is used, in this format every data item is encoded in three fields, *Tag*, *Length* and *Value* [17].

C. Platforms and testbeds

VANET experimental solutions are essential for validation purposes and several testbeds have been proposed with the aim of assessing the technology and applications [18]. Technology focussed solutions experiment on the applicability of the currently available technology in the vehicular field and application focussed solutions analyse how current existing applications behave in vehicular environments. Next we will describe some relevant projects in the field in order to define the design goals of our system.

In the COM2REACT project [19] Vehicle to Vehicle (V2V) based communications are evaluated, with IEEE 802.11b based communications, with the conclusion that Line of Sight (LOS) is one of the most important issues in vehicular communications. A multi-hop testbed created at the University of Murcia, Spain [18]. It comprises up to 4 vehicles and their communication is based on the Optimized Link State Routing (OLSR) protocol. The authors point out that while OLSR is a good reference point for VANET research, it fails to address all the issues related to vehicular communication. Vehicle to Infrastructure (V2I) communications are studied in FleetNet [20] and at the University of California, Davis [21] in different ways, but come to the same conclusion on the use of UDP instead of TCP due to TCP's poor performance in mobile wireless environments.

In C-Vet [10], V2V, V2I and Infrastructure to Infrastructure (I2I) communications are used, the VANET testing platform is supported by a mesh network of routers, working as a backbone. This solution focuses on testing application behaviour and how VANETs can support applications. The authors concluded that VANET-specific applications are mostly based in geographic routing and require the existence of a Location Service.

Another platform that studies application behaviour in VANET environments is called CarTorrent [22]. This work focusses on V2V information sharing and it addresses information retrieval strategies in highly mobile scenarios.

In Orbit [23] a large scale testing solution is presented. The solution incorporates up to 100 nodes and focuses on the study of packet delivery rates in dense VANET environments. The authors study the behaviour of some VANET applications but also state that the simulation of node mobility is an issue that remains to be addressed.

Finally, in the Network on Wheels project [24], a hybrid simulation and emulation solution is presented. This solution emulates the VANET network stack on top of the Linux network stack, and the emulated nodes run on a single machine. This solution does not enable field testing, meaning that it is only usable in a laboratory environment.

III. TAGUSVANET PLATFORM

Given the interest that VANETs have aroused in different communities, it is expected that in the near future, the development of applications for these networks will have a huge growth. Despite the wide diversity of applications that can arise, the set of requirements that differentiates them is very limited and can be grouped in a very simple way.

Hence, the main challenge is to find an easy way of defining the application requirements and enforce the mechanisms needed to satisfy them.

We decided to use a policy-based management approach to support our view and implement our platform. Instead of specifying policies related to specific applications, we specify a set of policies related to each application type. These policies will be than translated into network-level policies that enforce the requirements.

The next sections provide additional insights into our platform.

A. SW architecture

Our aim is to create a low cost platform for application development and test that simplifies the creation of VANET-oriented applications and VANET-related research. This is achieved in two ways: use of current standards and use of policies. The current standards are used to communicate between nodes and abstract the inner workings of sensors installed in the node. The policies strive to provide an application developer with a simple interface, which is used to define the application's needs and abstract some of the inner workings of a VANET. This enables the developer to focus more on the applications functionality, freeing it to create better applications. Network behaviour is modelled according to the application needs, by mapping higher layer policies into network layer ones.

The software architecture of the platform is depicted in Figure 2, comprising the main modules: *Application Developer Interface*, *Emulator*, and *Main System*.

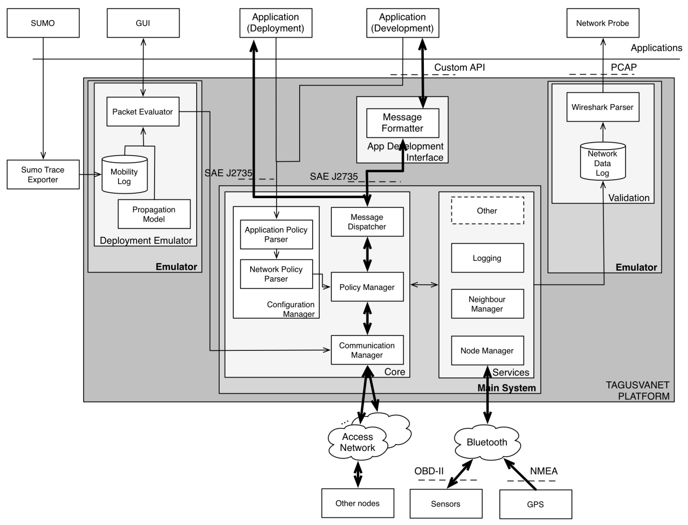

Fig. 2: Platform architecture.

a) Application Developer Interface: This module comprises a single component: the **Message Formatter**. The **Message Formatter** receives/sends messages to applications during development phase, using a very simple custom Application Programming Interface (API). These messages are converted into the standard SAE J2735 format so that they can be processed by the **Main System** module.

b) Emulator: The aim of the **Emulator** module is to create a set of tools that allow the developer to emulate a real VANET.

These tools should allow an easy validation of the application in the deployment environment. These two aspects are spliced in the module into two different components: the **Validation** and the **Deployment**.

Concerning the **Validation**, we can consider that the validation process is twofold: first, we need to guarantee that the messages are adequately built; second, we need to verify whether the application's requirements are met or not. The validation of the message format may be easily performed by capturing and visualising the generated message with a network probe. To verify the requirements fulfilment, the entire message flow must be monitored. This means that, every node must be able to store in a log file all the information that passes through it (received or generated by the node itself). Logging is a service provided by the **Main system** module and used by the **Emulator** module. An integration with Wireshark[1] tool facilitates the debugging.

Concerning the **Deployment**, we can think that the application's deployment is validated if we can reproduce in the laboratory real working conditions of the VANET. These conditions imply that each node in the lab has a behaviour similar to one in a real environment. Hence, real data gathered from vehicles (vehicle position) and adequate propagation models are provided. The platform can also be supplied with simulated data from vehicles. This data is provided by Simulation of Urban MObility (SUMO)[2], which can be seeded with real data from road networks. The propagation models implemented so far are the Free Space Path Loss Model (FSM) [25] and the Single Knife Model (SKM) [26].

c) Main system: The **Main System** is divided into three components, the **Configuration Manager**, which is in charge of configuring the system's behaviour, the **Services**, in charge of providing support for the **Core** and applications as required and the **Core**, which implements the network policies and handles data from and to the applications.

The **Configuration Manager** is in charge of parsing the applications needs, in form of an application policy, and converting them to network-level policies which are used by the **Core**. To achieve this, the **Configuration Manager** is composed of two elements, the **Application Policy Parser** and the **Network Policy Parser**.

All the services supported by the platform are grouped into the **Services** component. The current version, includes the basic services needed to support the majority of applications, namely: the **Logging**, **Neighbour Manager** and **Node Manager**.

The **Core** component is responsible for the coordination of the communication process. For this, three different core services are needed: the **Message Dispatcher**, **Policy Manager** and the **Communication Manager**.

The **Message Dispatcher** is in charge of information flow. It receives data from the lower layers and sends it to the **App Interface** block, or it selects and sends messages from the applications or from the services.

The **Policy Manager** is in charge of enforcing the *Network-oriented Policies* needed to support the applications and defined according to the associated *Application-Policy*.

[1]http://www.wireshark.org/

[2]http://sumo-sim.org/

To enable the platform to support multiple access networks, the **Communication Manager** is used. It allows the platform to be used in multiple contexts, from production systems, equipped with IEEE 802.11p radios, to testbeds, which can be based in any access network available.

IV. Policy-based Approach

In order to simplify the development of new VANET applications, several types of policies were defined. While some policies like the Application-oriented Policies define what an application developer has to worry about when developing a new application. Others, like the Network-oriented Policies define how each message is handled. The next sections will detailed them.

A. Application-oriented Policies

As stated in Section I, VANET-oriented applications have multiple requirements. For instance, a given type of application may need to use broadcast to disseminate information during a short period of time, while, other type, may need to send data using geocast communication, for a single event. These requirements must be introduced in the platform in order to create the appropriate processing rules, at the network level.

Next, we will present the policies that were defined at the application-level.

1) Policy-Definition: Regardless of the purpose of the application, all VANET-oriented applications have some of the same requirements. These requirements can be expressed in the following policy:

App Name | App ID | Communication Type | <Optional>

The **App Name** field is defined by the developer and is used to provide the developer with a simple, user friendly way of identifying the application in case there is a need to update the application's requirements.

The **App ID** is used by the policy enforcer when handling messages from each application. Because it is used for each application this field has to be chosen in a way that, while it may not be user friendly, the enforcer is able to use it efficiently. A possible solution is to attribute a different integer value to each application.

The **Communication Type** field is set by the developer according to the application's addressing requirements. This field's value can be one of the following: *Unicast*, *Geocast* or *Broadcast*.

The **Optional** field is, as the name implies, optional and can be used to specify useful information that does not fit anywhere else. This field can be repeated as necessary, the different information is separated by colons, in order to improve readability. So far this field can be one of the following: *Message Lifetime*, *Region Format*, *Region Size*, *Region Direction*. All of these fields, need to have values. For this end, the optional fields have the following format: *Field = Value*.

In order to further ease the developer's work, the most common numerical values of these fields have been converted to a small set of keywords. These keywords can be used by the developer to ease the development but these fields can also be defined using a custom numerical value if the developer sees fit. In the following tables this capability is represented by the *Custom* keyword.

In the *Message Lifetime* and *Region Size* fields, the conversion between keywords and value is shown in Table I.

Keyword	Value	Keyword	Value
Short	100 ms	Small	200 m
Medium	2 min	Medium	500 m
Long	10 min	Wide	2 km
Custom	*Input* in ms/s/min/h	Custom	*Input* in m/km

TABLE I: *Message Lifetime* and *Region Size* keywords.

The *Region Format* and *Region Direction* fields have no need for numeric values, in this case the keywords possible are shown in Table II.

Format	Direction
Circular	Back
Rectangular	Front
Polygonal	Left
Sector	Right

TABLE II: *Region Format* and *Region Direction* keywords.

For the *Region Format* keywords, the destination region specification in the IEEE 1609.2 standard [27] was used and a new keyword was defined. The Sector keyword is defined as a circular sector, with a 45° angle.

The *Region Direction* field is to be used with the Sector keyword. This field defines the orientation of the circular sector in relation to the current heading of the source node.

2) Policy-Examples: This formal policy can be applied to the applications referenced in Section II-A.

For example, in the case of an *Emergency Vehicle Warning (EVW)* application, a formal policy will take the following form:

EVW | 0 | Geocast | Message Lifetime = Short, Region Format = Sector, Region Size = Small, Region Direction = Front

What this policy states is that the application with the **App Name** EVW uses *Geocast*-based communication and its **App ID** is 0. This policy also defines the following geographical characteristics: the region of interest encompasses all the nodes within a *200 m* long circular *Sector* in *Front* of the source node and that it's messages have a lifetime of *100ms*.

Since not all VANET-oriented applications have the same requirements, other applications' requirements may be associated with simpler policies.

For a *Fleet Management (FM)* application targeted at sparse rural networks, where the nodes send their data to whoever is available, is based in *Broadcast* communication. In spite of not needing to address a specific node in the network, this application still has very specific message lifetime needs. In this case these needs are expressed using a numerical value for the message lifetime.

FM | 3 | Broadcast | Message Lifetime = 24h

B. Network-oriented Policies

1) Policy-Definition: As noted in the previous sections, each application has its own requirements, in terms of addressing and message lifetime. The applications also expect that every message is delivered only once and as such they don't have be prepared for duplicate messages.

These requirements can be summarised in some policies. These policies can be applied to each received message, with the objective of meeting the application's requirements and assuring the application developer of how each message is handled.

More formally all policies have the following format:

Id | Target | Trigger | Rules | Actions | Constraints

The **Id** is used to refer to the policy. The **Target** is who the message is intended to, the target can be a single node, the *Destination Node*, a *Set of Nodes*, the Region of Interest, or *all the nodes* in the source's neighbourhood.

The **Trigger** is the event that prompted the execution of this policy. The event can be either a new message is *Received*, an *Internal Action* or a *Timeout*, both generated by another policy.

The **Rules** are applied in a chain, where each rule has two possible outcomes, true or false. When a rule returns true, the next rule is evaluated, otherwise the according action is taken. In the last rule, two actions are taken, one for a positive outcome and one for the negative outcome.

An **Action** specifies what is to be done in case a **Rule** fails. In the case of the last **Rule**, two **actions** are specified, one for each possible output. The actions taken can be as simple as *Discard*, *Send to Network* or *Deliver to Application*, which don't need additional processing, or more complex actions, which are described by other policies, for example *Forward* or *Update Broadcaster*.

The **Constraints** are applied to each **Action** the policy has, these can be of any type or format that apply to the action they pertain. For example, the constraints of the *Forward* action are in the form of a probability, with possible values between 0 and 1.

2) Policy-types: This formal policy can be used to specify how each message is treated according to it's application's needs. The following policies show how the application's needs are addressed at a network level.

The more simple case is when a message is broadcasted, a Policy for handling Broadcast messages can be:

RxBroadcast | All Nodes | Received Message | { isValid?; notDuplicate?; isSource? } | { Discard; Discard; Deliver to Application; Forward() } | { ;;; where $p = 1$ }

This Policy, named *RxBroadcast*, is targeted at *All Nodes* in the current nodes neighbourhood and is only triggered when a Broadcast message is received. Next the message is passed through a chain of rules, where some verifications are applied. The first check is if the message is *Valid*, then if it is not a *Duplicate*. If the message fails one of these checks, it is discarded. Next the current node is check if it is the *Source*, if this check fails the message is *Delivered to the Application*. Otherwise the message is *Forwarded* to all the nodes in the sources neighbourhood, with the constraint that the probability defined in Equation 1 is equal to one, meaning that this message is always transmitted.

In the case of Unicast, this Policy looks like this:

RxUnicast | Destination Node | Received Message | { isValid?; notDuplicate?; isNextHop?; isDestination? } | { Discard; Discard; Discard; Forward(); Deliver to Application } | { ;;; where $p = 1$; }

What this means is that the Policy, which is named *RxUnicast*, has a target, that is the *Destination Node*, which is called when a new unicast message is *Received*. This message goes through a chain of rules to ensure that the application's requirements are met. These rules verify is the message is *Valid*, *Duplicate* or if the current node is the *Destination Node*. If the message is *Invalid* or *Duplicate*, the message is discarded. Next the current node checks if it is the message's *Next Hop*, the message is discarded if it isn't for this node. The choice of *Next Node* can be done using one of the many Unicast-oriented protocols available. Next if the current node is the *Destination Node*, the message is delivered to the application it belongs. Otherwise the message is *Forwarded* to the next hop, as in broadcasting messages, these messages have the constraint that the forwarding probability is equal to one.

In Geocast, like in broadcast and unicast, every message has to be checked if it is still relevant. However when a message isn't relevant, in the case of Geocast, some actions need to be taken in order to prevent the broadcast storm problem described in Section II-A. The Geocast-oriented Policy is the following:

RxGeocast | Node Set | Received Message | { isValid?; inRegion?; notDuplicate?; } | { Discard; Discard; UpdateBroadcaster(); Forward() + Deliver to Application } | { ;;; where $p = p_{ij}$ }

So the Policy named *RxGeocast* is targeted at all the *Nodes in Region of Interest* of this message. The message is then checked for its *Validity* and *Discarded* if it is *Invalid*. Next the current node checks if it is *In* the Region of Interest, if this message is irrelevant to the current node, the message is *Discarded*. Then if the message is duplicate, the *latest broadcaster* information is updated to so that the Broadcast prevention mechanism is used. Otherwise, the message is *Forwarded* to the neighbouring nodes, using the Broadcast prevention mechanism, and *Delivered* to the application it belongs.

In order to fully implement the Broadcast-Storm prevention mechanism, three more policies have to be defined. These specify the *Forward*, *UpdateBroadcaster* and *PreventDieOut* policies, that support such mechanism.

The *Forward* policy is the following:

Forward | Node Set | Internal Action | isProbOverThreshold? | { PreventDieOut(); Send To Network } | { Where $timeout = t$; }

The *Forward* policy is targeted at a *Node set* and triggered by an *Internal Action*. Next the message forwarding probability is checked *if it's above* 50%, if it is, the message is *Sent to the Network*, otherwise, the PreventDieOut policy is called.

The PreventDieOut policy is used, as it's name implies, to prevent the message dying out while preventing Broadcast-Storms, as specified by Tonguz and Wisitpongphan [13].

PreventDieOut | Node Set | Timeout | Rebroadcast? | { Discard; Forward() } | { ; Where $p = 1$}

Since the *PreventDieOut* policy is only used when in Geocast, the policy is targeted at the *Set of Nodes* in the region of interest and triggered by a *Timeout*. The message is then checked if it's suitable for *Rebroadcasting*. The message is *Discarded* if it's not and *Forwarded* with a probability of one, if it is.

For a message to be validated for Rebroadcasting, it is necessary to keep track of messages rebroadcasted by other nodes. The policy that handles this behaviour is expressed in by the following:

UpdateBroadcaster | Node Set | Internal Action | isCloser? | { Discard; Discard } | { ; }

The *UpdateBroadcaster* policy, as the *PreventDieOut* policy, is targeted at the *Set of Nodes* in the region of interest and, like the *Forward* policy, triggered by an *Internal Action*. This policy checks if the message broadcaster *is Closer* to the current node than the last broadcast of the same message, in either case the message is *Discarded*.

V. Evaluation

In order to evaluate the developed platform, several tests were performed. The tests were conducted in both real and emulated scenarios, in order to access the usability of the features included in the platform. All the tests were performed using the software platform detailed in Section III. The following sections detail these tests.

A. Emulator Module

Several tests were performed on the emulator modules to assess its behaviour. These tests were focussed on the propagation models used.

In order to test how each model behaves, each model was tested by itself. For each the physical characteristics of the 802.11b physical layer were used, in this case a frequency of 2.4GHz and a channel bandwidth of 20MHz. The test consisted on varying the value of the distance between sender and receiver. After each model was tested individually in the previously detailed setting, for each value of distance, the Packet Delivery Ratio was computed. This computation consisted of summing all the ”would-be” delivered packets, for a distance value, and dividing that value for the number of test packets for that value. The resulting PDR graphs are depicted in Figure 3.

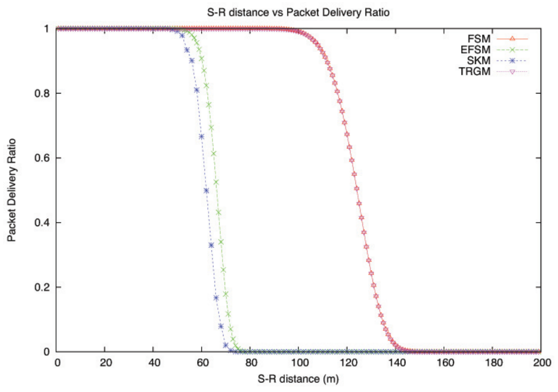

Fig. 3: Propagation model behaviour.

From these results, it is also concluded, that the results for FSM are consistent with the findings in [28], where the authors found that the Packet Delivery Ratio experiences a sudden and steep drop at a distance of 140m.

Next the models were implemented in the platform and the impact of two models in the Control Channel (CCH) delivery success rate was tested. In this test, two propagation models were used, FSM and SKM, the latter configured as in the previous test. These models were chosen for their representation of both the best and worst possible communication conditions. In this test, the nodes were emulated to be stationary and within LOS.

The results are shown in Figure 4.

As expected, the more conservative model has a greater impact on the delivery success rate. Another result, this test shows is that when the number of nodes is increased, the success rate is decreased, albeit little, meaning that the number of packet collisions increases and this may lead to critical information being lost.

B. Field Demonstrator

In order to demonstrate the platform’s functionalities a proof of concept application was developed. This application was geocast-based and used three nodes, two On-Board Units (OBUs) and one Road Side Unit (RSU), one of the OBUs was stationary and the the RSU was positioned in a second floor window of the main building. This application was used to show the platform’s information dissemination capabilities and sensor data gathering.

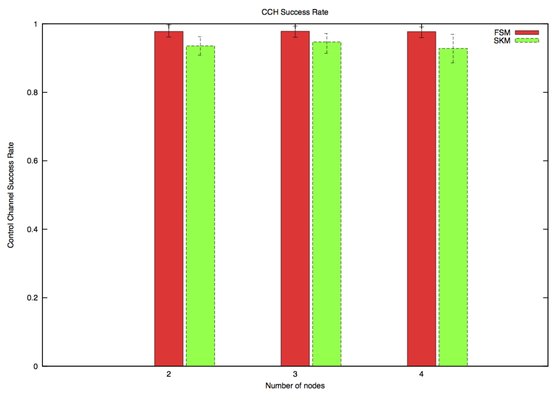

Fig. 4: Propagation model impact on the CCH.

The following Application-level policy was created to support this proof of concept:

PoC | 0 | Geocast | Region Format = Circle, Region Size = Medium, Message Lifetime = Medium

In Figure 5, the RSU is shown in cyan, the stationary OBU in pink and the mobile OBU in yellow. The figure also shows the OBU’s sensor information, as received by the RSU.

VI. Conclusions

This focuses on VANET application development, validation and deployment. To address these issues a software architecture is defined, which is split into three main components, the **Application Developer Interface**, **Emulator** and **Main System**. In order to ease the developers work in defining and configuring the platform to meet his applications requirements a policy-based solution for this problem is also defined. The *Application Developer Interface* defines a Custom API, which allows the developers to inform the platform of their applications’ needs. The *Emulator* supplies a number of tools to support the development of applications. Finally, the *Main System* is the core the platform, it implements the policy enforcement mechanism and is the platform’s contact with the other nodes.

After the platform was defined and developed, a series of tests were devised to test its features, the main focus was the *Emulator*. The tests showed that the used propagation models were correctly emulating the field conditions and showed the impact on the CCH of the different propagation models. One proof of concept application was developed to demonstrate the platform’s functionalities in the field, with great results.

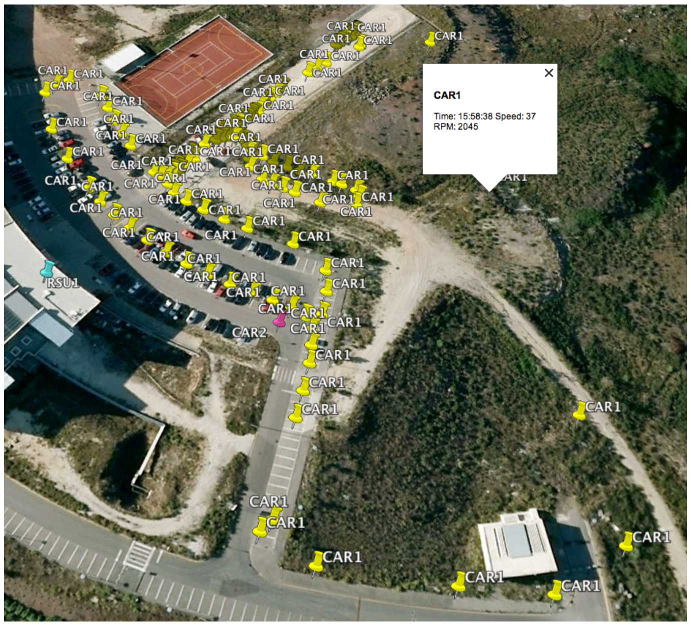

Fig. 5: Example of information retrieved from the platform.

References

[1] H. Hartenstein and K. P. Laberteaux, "A tutorial survey on vehicular ad hoc networks," *Comm. Mag.*, vol. 46, no. 6, pp. 164–171, Jun. 2008. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2008.4539481

[2] Y. Toor, P. Mhlethaler, A. Laouiti, and A. de La Fortelle, "Vehicle ad hoc networks: Applications and related technical issues," *IEEE Communications Surveys and Tutorials*, pp. 74–88, 2008.

[3] F. Bai, H. Krishnan, V. Sadekar, G. Holland, and T. ElBatt, "Towards characterizing and classifying communication-based automotive applications from a wireless networking perspective," 2006.

[4] X. Hong, K. Xu, and M. Gerla, "Scalable routing protocols for mobile ad hoc networks," *Network, IEEE*, vol. 16, no. 4, pp. 11–21, Jul 2002.

[5] R. Friedman and G. Kliot, "Location services in wireless ad hoc and hybrid networks: A survey," *Techanical Report, Technion Computer Science*, 2006.

[6] W. Chen, R. K. Guha, T. J. Kwon, J. Lee, and Y.-Y. Hsu, "A survey and challenges in routing and data dissemination in vehicular ad hoc networks," *Wireless Communications and Mobile Computing*, vol. 11, no. 7, pp. 787–795, 2011.

[7] H. Menouar, F. Filali, and M. Lenardi, "A survey and qualitative analysis of mac protocols for vehicular ad hoc networks," *Wireless Communications, IEEE*, vol. 13, no. 5, pp. 30–35, 2006.

[8] K. C. Lee, S.-H. Lee, R. Cheung, U. Lee, and M. Gerla, "First experience with cartorrent in a real vehicular ad hoc network testbed," in *2007 Mobile Networking for Vehicular Environments*. IEEE, 2007, pp. 109–114.

[9] C. Pinart, P. Sanz, I. Lequerica, D. García, I. Barona, and D. Sánchez-Aparisi, "Drive: A reconfigurable testbed for advanced vehicular services and communications," in *Proceedings of the 4th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities*, ser. TridentCom '08. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 16:1–16:8. [Online]. Available: http://dl.acm.org/citation.cfm?id=1390576.1390595

[10] M. Cesana, L. Fratta, M. Gerla, E. Giordano, and G. Pau, "C-vet the ucla campus vehicular testbed: Integration of vanet and mesh networks," in *Wireless Conference (EW), 2010 European*, April 2010, pp. 689–695.

[11] H. Krishnan, F. Bai, and G. Holland, "Commercial and public use applications."

[12] A. Festag, G. Noecker, M. Strassberger, A. Lübke, B. Bochow, M. Torrent-Moreno, S. Schnaufer, R. Eigner, C. Catrinescu, and J. Kunisch, "NoW - Network on Wheels: Project Objectives, Technology and Achievements," *Proceedings of 5th International Workshop in Intelligent Transportation*, no. March, pp. 211–216, 2008. [Online]. Available: http://www.network-on-wheels.de/documents.html

[13] O. Tonguz and N. Wisitpongphan, "On the broadcast storm problem in ad hoc wireless networks," 2006. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4374403

[14] R. Uzcátegui and G. Acosta-Marum, "WAVE: A Tutorial," *Communications Magazine, IEEE*, vol. 47, no. 5, pp. 126 –133, May 2009.

[15] S. International, "DSRC Implementation Guide," pp. 1 –210, Feb 2010.

[16] ITU-T, "Information technology Abstract Syntax Notation One (ASN.1): Specification of basic notation," *Recommendation X.680*, pp. 1 – 194, Nov. 2008.

[17] ——, "Information technology ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)," *Recommendation X.690*, pp. 1 – 38, Nov. 2008.

[18] J. Santa, M. Tsukada, T. Ernst, and A. Gomez-Skarmeta, "Experimental analysis of multi-hop routing in vehicular ad-hoc networks," in *Testbeds and Research Infrastructures for the Development of Networks Communities and Workshops, 2009. TridentCom 2009. 5th International Conference on*, April 2009, pp. 1–8.

[19] V. González, A. L. Santos, C. Pinart, and F. Milagro, "Experimental demonstration of the viability of ieee 802.11b based inter-vehicle communications," in *Proceedings of the 4th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities*, ser. TridentCom '08. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 1:1–1:7. [Online]. Available: http://dl.acm.org/citation.cfm?id=1390576.1390578

[20] A. Festag, H. Fußler, H. Hartenstein, A. Sarma, and R. Schmitz, "FLEETNET: Bringing Car-to-Car communications into the real world," *Computer*, vol. 4, no. L15, p. 16, 2004.

[21] F. Hui and P. Mohapatra, "Experimental characterization of multi-hop communications in vehicular ad hoc network," in *Proceedings of the 2nd ACM international workshop on Vehicular ad hoc networks*. ACM, 2005, pp. 85–86.

[22] K. Lee, S.-H. Lee, R. Cheung, U. Lee, and M. Gerla, "First experience with cartorrent in a real vehicular ad hoc network testbed," in *2007 Mobile Networking for Vehicular Environments*, May 2007, pp. 109–114.

[23] K. Ramachandran, M. Gruteser, R. Onishi, and T. Hikita, "Experimental analysis of broadcast reliability in dense vehicular networks," in *Vehicular Technology Conference, 2007. VTC-2007 Fall. 2007 IEEE 66th*, Sept 2007, pp. 2091–2095.

[24] R. A. W. R. Costa, S. Sargento, "Development of a Hybrid Simulation and Emulation Testbed For VANETs," June 2009.

[25] ITU-R, "P.525 : Calculation of Free-Space Attenuation," *Recommendation P.525-2*, pp. 1 – 3, Aug. 1994.

[26] ——, "P.526 : Propagation by diffraction," *Recommendation P.526-13*, pp. 1 – 43, Nov. 2013.

[27] "IEEE standard for wireless access in vehicular environments security services for applications and management messages," *IEEE Std 1609.2-2013 (Revision of IEEE Std 1609.2-2006)*, pp. 1–289, April 2013.

[28] D. Pham, Y. ekerciolu, and G. K. Egan, "Performance of IEEE 802.11b Wireless Links: An Experimental Study," in *TENCON 2005 2005 IEEE Region 10*, Nov 2005, pp. 1–6.