Service Platform for Vehicular Networks

Pedro Cruz Sousa
pedro.cruz.sousa@ist.utl.pt
Instituto Superior Técnico, Taguspark
Lisboa, Portugal

Teresa Vasques Vazão
teresa.vazao@ist.utl.pt
Instituto Superior Técnico, Taguspark
Lisboa, Portugal

October 2012

Abstract

In this paper, we summarize a study on related Vehicular Ad-hoc Network (VANET) standards, some past research projects and propose a solution for a service platform in a vehicular environment. We present and analyse the measurements taken with our developed service supporting platform and evaluate the feasibility of store and forward mechanisms and also different types of communication: multi-hop, Vehicle-to-Infrastructure (V2I), Infrastructure-to-Vehicle (I2V) between two Road Side Unitss (RSUs) and an On-Board Unit (OBU) (installed in a vehicle). Our study shows the practicability and possibility of transmitting data in such an environment with little packet loss and latency.

Keywords: VANET, IEEE 802.11, Platform, Multi-hop communication, Store and forward, V2I communication, Experimental performance

1. Introduction

The population of the world, as well as the demand of vehicles for transportation, has been increasing in the past decades. Such demand results in higher congestion issues resulting in higher waiting times and higher pollution, thus degrading the social well-being. Such long waiting periods also wastes fuel and with its pricing soaring and a potential threat of its shortage, there is a clear need to improve road traffic.

Road safety is yet another problem. Every year, accidents cause 43.000 deaths in the USA, in Europe the number of people killed every day is comparable to a medium-haul plane crash (between 180 and 260 people) and in the Asia Pacific region 10 million people are severely injured or killed on the roads every year. Such death toll is major and grim.

In the past decade, numerous efforts that sought to mitigate the aforementioned problems, produced solutions like: information on traffic and hazardous situations being broadcast via FM radio band; variable message signs that warn drivers about changing conditions placed along freeways; electronic toll systems that collect fees with reduced or almost no disruption of traffic flow. These systems are currently being used in many countries, but are not enough to avoid or minimize the presented problems.

Also, with the rapid development of Internet there is need to integrate its applications which provide value-added services and extra comfort to trips

With the evolution of the automotive industry, sensors and Global Positioning System (GPS), vehicles are now equipped with various supporting technologies that can aid the driver. With the growth of mobile computing and communications, such information can be shared between vehicles and individual data can become a collective effort to support traffic in general. Also, with the cooperation of other systems one can enable the deployment of various types of applications and services, which can include: road and weather status information, congestion control, etc. Also, Internet applications, e-commerce or even vehicle operation services (such as vehicle inspection) can be included and further developed.

It is with this in mind that we proposed a solution for a services and applications platform. This platform provides means for their development in an easy and seamless way. It also supports communication between vehicles and existing systems, as well as facilities to easily obtain information of the surrounding environment and for the deployment of applications. One major contribution of the platform is the fact that - as we will see - it does not use a dual-stacked approach on the network and transport layer like most of the existing solutions, but rather makes use of a location protocol and runs it over the typical TCP/IP stack used on the Internet.

This document is divided as follows: Section 2 discusses the Related Work, Section 3 and Section 4 denotes the Architecture and Implementation Details, respectively, Section 5 shows the Results of

the performed tests and finally, we conclude and make the final remarks on Section 6.

2. Related Work

In this section we will discuss some of the most relevant standards, technologies and research projects that contribute to the design of a Vehicular Ad-hoc Network (VANET) architecture, as well as, the design of a platform to support communication within such architecture.

2.1. Mobile standards and technologies

IEEE 802.11 has been around for quite awhile and it is currently one of the most used wireless standard in the world. Depending on the needs - such as QoS guarantees or bandwidth - variations on the standard have been proposed, e.g., 802.11a/b/g/n. Unfortunately, and as shown in [6, 11], these standards are not fitting for a VANET environment, mainly because it was designed for low mobility network - such as people walking with their wireless devices. With this in mind, the ASTM and IEEE adopted the Dedicated Short-Range Communications (DSRC), which is based on the IEEE 802.11a PHY and MAC layer with a few modifications, in order to maximize reliability and provide a more robust and secure mean of transmission for an automotive environment [9]. DSRC is currently standardized as the IEEE802.11p protocol and it is used within the Wireless Access in Vehicular Environments (WAVE) standard.

WAVE defines an architecture, communications model, management structure, addresses security needs and physical access. Its architecture is composed of an OBU, RSU and a WAVE interface. The WAVE operation is regulated by the IEEE 802.11p and the IEEE 1609.x family of standards and it operates below a few management protocols. For more details, we refer the reader to [12].

Finally, the Communication Access for Land Mobile (CALM) architecture is a set of standards that support communication media and application diversity and allows all communication scenarios, such as Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication, multi-hop, unicast and multicast. Its communication system is composed of four sub-systems: roadside, vehicle, central and personal, each one of them has their own instances, each including an Intelligent Transportation Systems (ITS) station performing CALM-specific functions. Services and applications run over a Communication Kernel - which is used to exchange information between stations - such kernel is composed of a CALM: Manager, Networking and Media. The CALM Manager is the core of each ITS station and it ensures that a given data flow is matched to a given communication medium according to a couple of parameters and configurations. The CALM Networking layer is composed of both IPv6 and ITS-specific protocol called CALM FAST for non-IP communications and includes mobility management protocols, like NEMO. Finally, the CALM Media implements reliability parameters indicating the quality of the link. Each CALM radio channel is virtualized and, as a result, this layer can access its properties and set them on a packet-by-packet basis. For further details, we refer the reader to [2, 8]

2.2. Research Projects and Discussion

In the following section, we will give an overview of the following research projects: Co-operative Vehicle-Infrastructure Systems (CVIS), Network on Wheels (NoW) and Safe Intelligent Mobility (simTD). These were chosen among many others (Safespot and COOPERS [1] or WiSafeCar [11], to name a few) because they propose a solution that intends to answer the same problem as ours - that is to provide a suitable VANET platform.

CVIS aims to enable a flexible, harmonised and open communication between its architecture components - vehicles, roadside system, central system - to improve existing services and develop new ones [7]. Its involved parties use a multi-homed approach for a continuous network connectivity, involving infrared light, radio communication based on WAVE, radio communication at frequencies above 40GHz, 2G/3G cellular radio technology and DSRC. All of these technologies are unified and integrated by the CALM standard [2, 1]. CVIS conducted several tests to analyse the performance and handover behaviour of the involving technologies. Such study [8] demonstrated that a vehicle travelling towards a RSU and away (at 100km/h) could achieve throughputs of 5Mb/s with packet sizes of 1KB and that handovers would not interfere with the applications' session.

NoW has two involving parties: vehicles equipped with OBUs and fixed stations alongside the road (RSU). Such design enables V2V and V2I communication. NoW divides applications between safety (e.g. avoid car crashes) and comfort (e.g. web browsing). Each of these have different requirements and needs and, so, it uses different approaches for both of them [3]. Its platform uses a dual stack approach in which: safety-related applications use a novel network and transport protocols that provide ad-hoc communication and multi-hop - such as GeoBroadcast, GeoAnycast, GeoUnicast and Topologically-scoped Broadcast - among OBUs and RSUs over a IEEE802.11p radio protocol; comfort-related applications use the traditional TCP/IP protocol stack over IEEE802.11a/b/g pro-

tocols. Although each type of application has its own stack, that does not mean that comfort-related applications cannot use the stack of its counterpart. This way, one can define basic systems and extend already existing ones [3]. The results achieved by NoW contributed to activities of standardization bodies and the presented platform was used within other working groups such as: Safespot, SEVECOM and Aktiv.

Finally, simTD aimed at the enhancement of components from previous projects (like the NoW project), as well as, the development of new ones. Its system architecture is composed of a vehicle domain and an infrastructure domain, and thus, it uses V2V and V2I communication. simTD vehicle station architecture is composed of two components: the Communication Control Unit (CCU) and the Application Unit (AU). The CCU integrates all communication modules and offers different abstraction layers for application access and is connected with the AU using Ethernet, standard protocols and proprietary adaptations. It uses IEEE802.11p, Universal Mobile Telecommunications System (UMTS) and GPS-based positioning. Based on NoW, it uses message queuing and forwarding schemes and includes congestion control. Above this layer, there is the transport layer and the facilities layer. The latter one enables an abstraction of V2V and V2I communication for the AU component [10].

As we have seen, the presented research projects share a common approach on the VANET architecture: the use of V2V and V2I communication. We believe that both are needed for various reasons. Particularly, the use of V2V is necessary in order for time-critical applications to experience low delays when transmitting important data.The use of V2I is necessary in order to expand the transmission range of a given vehicle and to connect it to other existing services and networks. As for the approach taken in the network layer, our viewpoint diverges. The use of a dual stack network layer has the following problems: uses too much resources as it is more complex and thus less adequate for hardware with low computational power and has more processing and message header overhead. We propose the use of a proper geo-routing protocol and location-based communication and a unique stack instead of two, thus eliminating the aforementioned issues. The same applies to the dual stack transport layer approach.

3. Architecture

In the previous sections, we have discussed the existing problems of a VANET environment, as well as, the different approaches and solutions taken by either standardization groups and/or research projects. In this section, we will explain our efforts in building a supporting platform for applications and services in a vehicular environment. But before hand, we will describe, yet, another platform - the Multi-Functional Platform - in which ours works over, and then describe our Service Platform.

3.1. Multi-Functional Platform

The multi-functional platform [4] is a modular platform that provides means for the development of applications with great flexibility and reconfigurability. Figure 1 denotes its architecture.

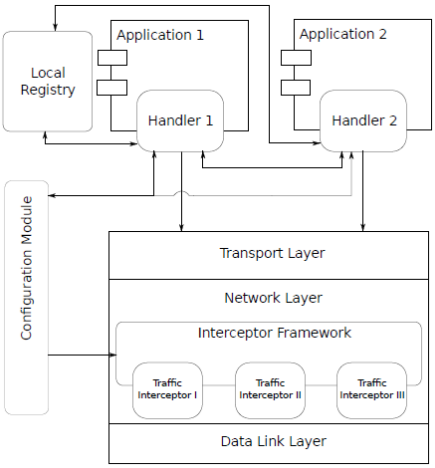


Figure 1: Multi-functional platform architecture overview

It is based on a group of key components that enable the development and configuration of applications. Such components are: *interceptor framework*, *handler factory* and a *configuration module*. The *interceptor framework* is based on a plug-in scheme that offers means for traffic manipulation by registering a set of rules (interceptors) in a rule manager. Such rules are composed of several filters that describe when and which packets should be intercepted. This scheme allows the implementation of features like security or packet manipulation. The *handler factory* creates handlers which are the main structure of the platform. The handlers goal is to abstract applications from transport details and traffic manipulations. Applications instantiate these handlers and configure them based on the type of transport they require. The platform provides two transport schemes, a unreliable and a reliable transport protocol based on a NACK scheme. A local registry is used to store the unique identifiers and the respective addresses of these handlers, in order for local and network communication to occur. Finally, the *configuration module* stores information about parameters used by each handler, such as how many messages the transport can

buffer.

3.2. Service Platform

When designing the platform, there were certain aspects that needed to be taken into account: (i) **scalability** - the system should work either in a sparse environment, as well as, in dense environments; (ii) **configuration** - the system should be configurable and allow as much configuration options, as possible; (iii) **flexibility** - applications should be provided with as much mean of communication and surrounding environment information as possible; (iv) **extensibility** - the system should provide ways to improve, extend and add features.

Taking into account the aforementioned requirements, we have designed a Distributed Architecture with asynchronous communication, in which, each component has a main focus. It is composed of four layers: Facilities Layer, Transport/Network Layer and Management. Figure 2 denotes our architecture and its integration with the multi-functional platform.

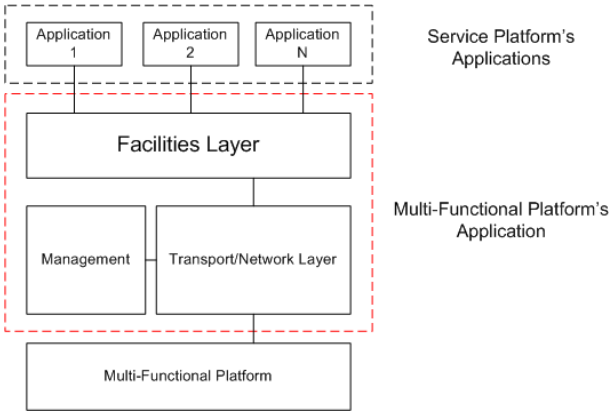


Figure 2: Service platform architecture overview

Further explaining, the *Facilities Layer* abstracts applications from lower layers by providing an API that facilitates messaging, neighbourhood information - such as surrounding vehicles and their correspondent speed, location, etc. - and real-time receival of nearby vehicles' information and data. The *Transport/Network Layer* is the core of the platform as it provides the typical TCP-based communication, a reliable scheme using NACKs and an unreliable scheme (UDP-based) for data transmission. It also provides means for broadcast, geocast and unicast transmission using location-based communication, as well as information of the surrounding environment. Finally, the *Management* is used to store the platform configuration, such as the time interval between the transmission of the vehicle's information (denoted as HELLO messages).

3.2.1 Management

The Management stores configuration parameters, so that the Transport/Network Layer can configure its components. Such parameters are as follows: an alphanumeric unique *ID*; *number of hops* which tells how many hops-distance should the platform store/send a given HELLO message; *timings* indicating the interval between each HELLO message and its removal; *data aggregation* which indicates the algorithms that should be used to aggregate the stored information; *cache cleanup* indicating the interval between each cache cleanup (applications' data); *buffer size* indicating the size of the local applications' data buffer.

3.2.2 Transport/Network Layer

The Transport/Network Layer is the core of the platform, and its architecture is shown in Figure 3. It is composed of: a *Network Manager*, *Table Manager*, *Sender/Receiver Module* and *Protocols*.

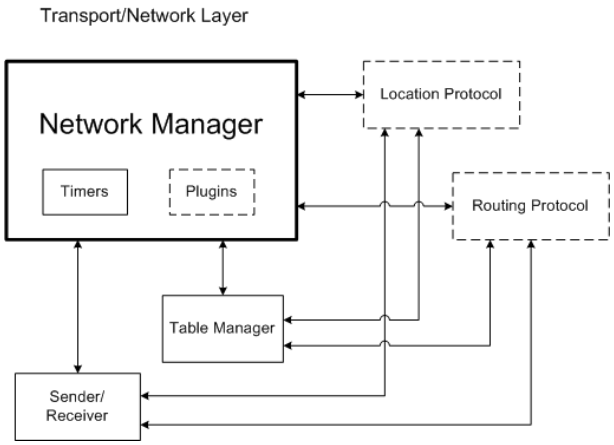


Figure 3: Transport/Network Layer components

As mentioned before, this architecture is mainly a Distributed System with asynchronous communication, in which parts of the layer can be replaced with similar components that can fulfil the task at hand. Such replaceable parts are denoted with dashed lines in Figure 3.

The main focus of the Network Manager is distributing all of the data between each of the components. Its timers are meant to either send the HELLO messages or to warn the Table Manager to remove any old instances of other vehicles and/or RSUs. The plugins are extensions to the platform and allow to subscribe to any received data and alter the times of each of the timers. Such subscription is made possible because of the design of the Sender/Receiver Module.

The Sender/Receiver Module is a simple component that communicates with the Multi-Functional

Platform and it is meant to send internal or receive external data. Because we have designed a Distributed System, we have developed a publish/-subscribe scheme within this module, allowing its components to receive data regarding HELLO, application data or protocol data - with a few restrictions, like protocols not allowed to receive application data and vice-versa.

The Table Manager is meant to store information regarding its own vehicle and the surrounding environment information at a hop-distance. Although the data is discretized, it is this component that makes use of the aggregation algorithms, so that it can aggregate data of the surrounding entities when sending data of its neighbours and thus allowing to send as much information, as possible within one message.

Finally, the Protocols are of two types: *Routing Protocols* and *Location Protocols*. They work closely with the Network Manager in order to find the location of a given vehicle/RSU or to route a given message. Such protocols can be replaced without altering the logic behind the Network Manager and thus allowing for the development of smarter and more efficient protocols.

3.2.3 Facilities Layer

The Facilities Layer abstracts applications from the lower layers. It includes a mechanisms to register with the platform, allow communication with other entities, as well as, subscription to received HELLO messages and requesting neighbours and their locations with the Table Manager.

4. Implementation Details

The implementation of the platform was made using the C programming language, alongside with Extensible Markup Language (XML) for configuration parameters and JavaScript Object Notation (JSON)-based formats for external and internal communication.

In order to understand how the Network Manager and the Protocols work together, we will show the implemented algorithms for the various types of message transmission.

Algorithm 1 shows the procedures used for broadcast. As observed, it is very simple and it is based on broadcasting a given message for a distance of N hops. The number of hops is defined by the application and this way we can avoid endless rebroadcasts. Another mechanism used to avoid network flooding and retransmitting the same message is storing them in a local cache. In order to avoid spending a great amount of memory in such storage, hashing and bloom filters are used.

Algorithm 2 shows the procedure for geocast transmission. The procedure is similar to broadcast in a way that uses the same mechanisms, but with the following differences:

1. if the vehicle is not within the range of the given location (in GPS coordinates) indicated by the received message, then the message is delivered to the routing protocol that will decide to which node should the message be forwarded;

2. if the vehicle is within range, then a broadcast procedure is started until the vehicles in the given radius are warned. In order to avoid flooding, the same cache procedure is used.

The fact that the decision of forwarding is transferred to the protocol, allows the detachment of the Network Manager from routing decisions, and, since the protocol can be changed, the platform can make better decisions with the improvement of such protocol without altering the Network Manager procedures.

Algorithm 1 Broadcast Algorithm

```
Require: code, message
1:  if code > 0 then                          ▷ Local generated message
2:      broadcast_message();
3:  else                                      ▷ Remote generated message
4:      if own_message() OR in_cache() then
5:          return;                           ▷ Discard message
6:      else
7:          message_hops ← message_hops − 1;
8:          broadcast_message();
9:          store_message();                  ▷ Store message in cache
10:     end if
11: end if
```

Algorithm 2 Geocast Algorithm

```
Require: code, message
1:  if code < 0 AND own_message() then
2:      return;
3:  end if
4:
5:  if in_cache() then
6:      return;
7:  else
8:      if vehicle_within_range() then
9:          broadcast_message();
10:     else
11:         send_to_routing_protocol();
12:     end if
13:
14:     store_message();
15: end if
```

Algorithm 3 shows the procedures used for unicast. While the previous algorithms always use an unreliable transmission channel, unicast can make use of the three types of transport provided by the Multi-Functional Platform. In all three cases, the algorithm is the same, only the transmission channel is changed.

The algorithm works as follows:

1. if the location of the destination vehicle is known, then the message is delivered to the routing protocol to forward the message;

2. if the location of the destination vehicle is not known, then the message is delivered to the location protocol, which will attach a location to the message, re-deliver it to the Network Manager and then, the previous item is started.

Once again, the fact that the Network Manager is detached from location decisions, allows for the platform to be loaded with better location protocols without altering the Network Manager procedures.

Algorithm 3 Unicast Algorithm

Require: *code*, *message*

```
1:  if code > 0 then
2:      found = check_local_table();        ▷ Check if destination is any of the neighbours
3:      if found == true then
4:          send_to_node(); ▷ Forwards the message to neighbour
5:      else
6:          if NOT message_has_location() then
7:              send_to_location_protocol();
8:          else
9:              send_to_routing_protocol();
10:         end if
11:     end if
12: else
13:     if NOT message_for_local_vehicle() then
14:         found = check_local_table();
15:         if found == true then
16:             send_to_node();
17:         else
18:             send_to_routing_protocol();
19:         end if
20:     end if
21: end if
```

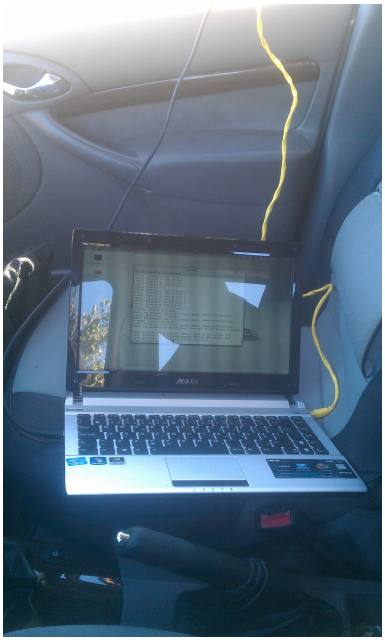
As for application development, we have defined an Application Programming Interface (API) for a key group of functionalities that allows the abstraction of communication, subscription with the Sender Module for HELLO messages and querying the Table Manager for any given neighbour.

5. Results

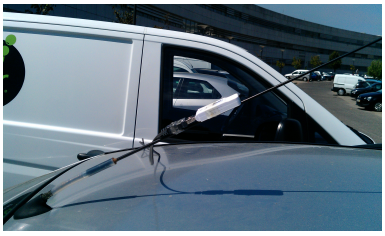
5.1. Equipment and software tools

To setup our test-bed, we used laptops running Linux operating system (Ubuntu 11.04) equipped with SMCWUSBS-N3 802.11b/g/n wireless pens (configured to run IEEE802.11g). We installed our platform within the laptops and developed a small application (using our facilities API) which, based on a configuration file, would either, send traffic to the network using our platform or receive the same transmitted traffic. Figure 4 shows our test-bed montage.

(a) Test-bed environment



(b) Laptop on car



(c) Wireless pen montage on car

Figure 4: Hardware and test-bed environment

5.2. Description of the tests

Our tests focused mainly on three important characteristics: I2V and V2I communication, store and forward and multi-hop. To do so, we performed the following test.

Observing Figure 5, the OBU (our vehicle) would start in point A to gain velocity (50 km/h would be our goal) and when reaching the RSU-B (at point B - 57m from point A), the transmission would begin from RSU-B to OBU. At this point, the OBU has not received any HELLO messages from RSU-D (at point D - 163m from point A) and so it does not know the existence of the messages' destination and thus, it would store them. Upon reaching RSU-D's transmission range (roughly at point C - 83m from point A) it would transmit the stored messages and then transmit all received data from RSU-B at the same rate that they are transmitted. The OBU would then stop at point E, turn back to point A and the test would be repeated. Point E was at 260m distance from point A.

We ran 4 different tests (shown in Table 1) with UDP, each with 5 runs, and thus making 20 travels.

Figure 5: Test route

Test	Packet Size (B)	Tx Rate (msgs/s)	Throughput (B/s)
1	64	16	1024
2	64	80	5120
3	512	4	2048
4	512	20	10240

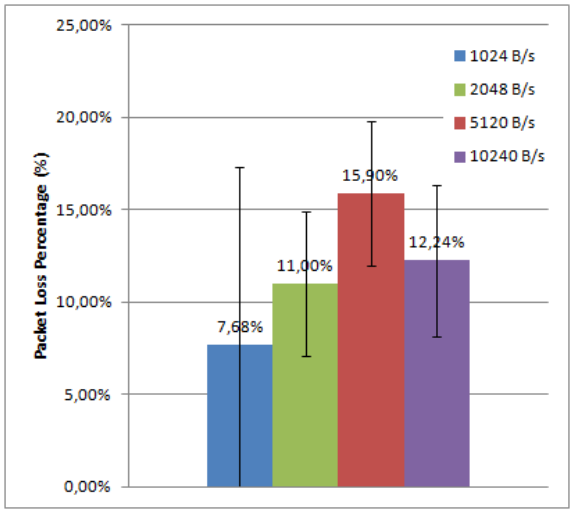
Table 1: Performed experiments

5.3. Results and analysis

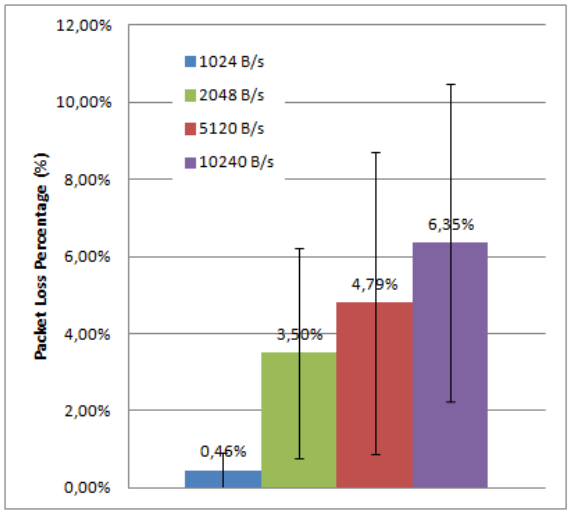
The performance metrics used to evaluate the connectivity between each involved party are: (i) **Packet Loss**: percentage of packet loss due to connectivity problems; (ii) **Latency**: the time that took a given message to travel to its destination; (iii) **Jitter**: the variation of the latency in the network.

Figure 6 shows an average packet loss (with error bars representing the calculated trust interval of 95%) for each test. We have discretized the total packet loss into two other plots, in order to understand where the most packets were lost. As we can see, The RSU-B transmitted, without great problems, most of the packets to the OBU, while the OBU and the RSU-D had the more losses. This could be because of the used protocol (802.11g instead of 802.11p) which is not meant for vehicular usage. Using the proper standard, the packet loss could decrease. Also, the surrounding environment could also have its influence; the street had a slight elevation which prevented the messages from being perfectly delivered, but also because the RSU-D was near a house with metal surroundings which could interfere with the transmission of the messages. Particularly, for 5120B/s case, in one of the runs other vehicles were passing by, during the execution of the test, thus increasing the packet loss percentage (to 15.58%), and therefore ruining the calculated average (which at the time was around 5%).

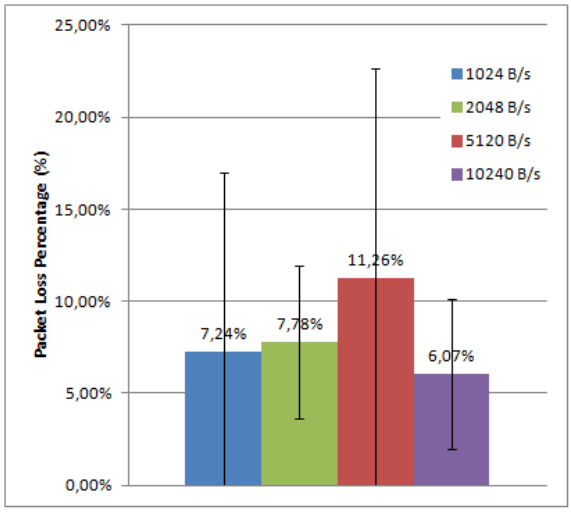
Figure 7 shows the behaviour of one of the performed tests. The points 1 and 2, show the store and forward behaviour of our platform; while in 1, the OBU had no knowledge of the RSU-D's existence, in 2, the OBU erased the RSU-D's reference from its local table and missed out a couple of HELLO messages. As such it stored the transmitted messages until the reception of a new HELLO message. The circled points show which packets



(a) Total Packet Loss



(b) From RSU-B to OBU Packet Loss



(c) From OBU to RSU-D Packet Loss

Figure 6: Packet Loss Total (a), from RSU-B to OBU (b) and from OBU to RSU-D (c)

were lost during the execution of the test and at the end of the plot, there were 3 undelivered messages that were stored within the OBU, but could not be transmitted because the OBU lost the reference of the RSU-D.

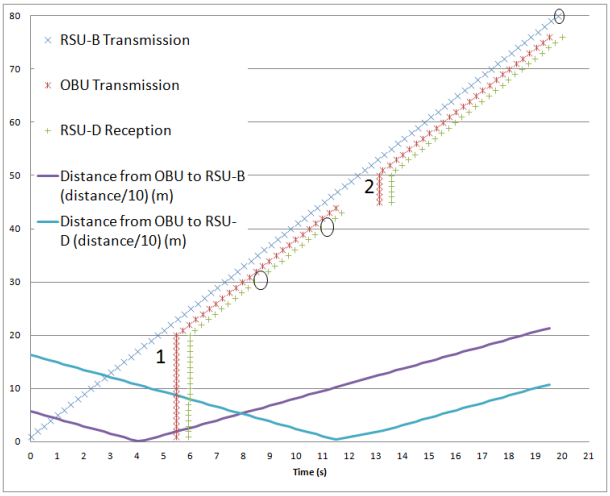


Figure 7: Packet Loss Distribution over Time

All of the presented graphics allow to calculate the desired performance metrics. Table 2 shows a summary of the packet loss with the respective trust intervals. From Figure 7 we can calculate the latency and jitter, which are 1,17s and 30,4ms respectively. It should be mentioned that the latency and jitter values are calculated only during a fraction of time that there was full connectivity between the RSU-B, the OBU and the RSU-D - that is, between 5 seconds and 12 seconds in. From the aforementioned calculations and values, we compared the packet losses with the study conducted by [5], analysed the other two metrics and verified that:

1. **Packet Losses**: Our packet losses are much lower than the ones presented in the study (ranging from 5% to 60%). This may be due to the fact that, although they used their RSU in a higher ground (which gives them a better transmission range), they also blocked the transmission with a bridge and thus increasing the packet loss percentage;

2. **Latency**: Our latency is around 1,17s and such values may be due to the fact that the laptops' clocks were not perfectly synchronized and thus increasing the value of this metric. So, we performed a ping test with a 64B data and the presented values fell into more realistic values; from the OBU to RSU-B we had an average of 3,437ms (with a deviation of 1,263ms) and from the OBU to RSU-D we measured an average of 6,792ms (with a deviation

	1024 B/s	2048 B/s	5120 B/s	10240 B/s
From RSU-B to OBU	0,46%	3,50%	4,79%	6,35%
Trust Intervals (95%)	0,44%	2,73%	3,92%	4,12%
From OBU to RSU-D	7,24%	7,78%	11,26%	6,07%
Trust Intervals (95%)	9,77%	4,18%	11,37%	4,09%
Total Packet Loss	7,68%	11,00%	15,90%	12,24%
Trust Intervals (95%)	9,66%	4,92%	10,01%	3,19%

Table 2: Packet Loss summary

of 4,431ms). These values are much more acceptable and plausible than those previously measured;

3. **Jitter**: From this metric, we can safely say that the presented values are acceptable and also, prove that the prior performance metric discrepancy is not due to lack of platform performance or bad measures, but to clock desynchronization.

6. Conclusions

In this study, we have presented the results and evaluation of the performance of I2V and V2I communication, alongside carry and forward mechanisms and multi-hop based on the IEEE802.11g technology. We have demonstrated that, it is possible and feasible to transmit data between two RSUs and an OBU without much loss and delay. Such experiments also proved the feasibility of extending the transmission range of an infrastructure by using multi-hop and carry and forward mechanisms. In the future and test-wise, we would like to test the platform with more vehicles and/or infrastructures, but also evaluate the TCP performance in a VANET environment. Implementation-wise, we would like to greatly improve our platform by either substituting the Multi-Functional Platform for a complete implementation of communication mechanisms or by improving its integration with our platform. Also, and given the easiness in developing protocols with our API, we would like to also refine the existing ones by taking into account other crucial factors (such as distance).

Acknowledgements

This work is supported by FCT (INESC-ID multiannual funding) through the funds of Programa PIDAC.

References

[1] J. Boussuge and C. Laurgeau. Comparative synthesis of the 3 main European projects dealing with Cooperative Systems (CVIS, SAFESPOT and COOPERS) and description of COOPERS Demonstration Site 4. *Traffic*, pages 809–814, 2008.

[2] T. Ernst, V. Nebehaj, and R. Sorasen. CVIS : CALM Proof of Concept Preliminary Results. *Media*, (December):80–85, 2009.

[3] A. Festag, G. Noecker, M. Strassberger, A. Lübke, and B. Bochow. NoW Network on Wheels : Project Objectives , Technology and Achievements. *Transportation*, (March):211–216, 2008.

[4] F. Gonçalves. Multi-Functional Platform for Indoor and Outdoor Monitoring. 2011.

[5] M. S. S. Jerbi, Moez. Characterizing Multi-Hop Communication in Vehicular Networks. 2008.

[6] A. Khan, S. Sadhu, and M. Yeleswarapu. A comparative analysis of DSRC and 802 . 11 over Vehicular Ad hoc Networks. *Ad Hoc Networks*.

[7] E. Koenders and J. Vreeswijk. Cooperative Infrastructure. pages 721–726, 2008.

[8] M. E. G. Moe, V. Nebehaj, and T. Ernst. CVIS Performance Test Results : Fast Handovers in an 802 . 11p Network. *Public Roads*, 2010.

[9] L. Stibor, Y. Zang, and H.-J. Reumerman. Neighborhood evaluation of vehicular ad-hoc network using IEEE 802.11 p. *Proceedings of the 8th European Wireless*, pages 1–5, 2007.

[10] H. Stübing, M. Bechler, D. Heussner, T. May, I. Radush, R. Horst, and P. Vogel. sim TD : A Car-to-X System Architecture for Field Operational Tests. *IEEE Communications Magazine*, (May):148–154, 2010.

[11] T. Sukuvaara, P. Nurmi, M. Hippi, R. Autio, D. Stepanova, P. Eloranta, L. Riihentupa, and K. Kaubo. Wireless Traffic Safety Network for Incident and Weather Information. *Network*, pages 9–14, 2011.

[12] Y. Toor, P. Muhlethaler, A. Laouiti, and A. De La Fortelle. *Vehicle Ad-hoc Networks: Applications and related technical issues*, volume 69. May 2008.