HAL
open science

Standards for Cooperative Intelligent Transportation Systems: a Proof of Concept

Rodrigo Silva, Satoru Noguchi, Thierry Ernst, Arnaud de La Fortelle, Walter Godoy Junior

To cite this version:

Rodrigo Silva, Satoru Noguchi, Thierry Ernst, Arnaud de La Fortelle, Walter Godoy Junior. Standards for Cooperative Intelligent Transportation Systems: a Proof of Concept. The Tenth Advanced International Conference on Telecommunications (AICT), International Academy, Research, and Industry Association (IARIA), Jul 2014, Paris, France. hal-01109115

HAL Id: hal-01109115

https://minesparis-psl.hal.science/hal-01109115v1

Submitted on 26 Jan 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Standards for Cooperative Intelligent Transportation Systems: a Proof of Concept

Rodrigo Silva, Satoru Noguchi,
Thierry Ernst, Arnaud de La Fortelle
Mines ParisTech
France
rodrigo_silvabr@yahoo.com, satoru.noguchi@mines-paristech.fr
{thierry.ernst, arnaud.de_la_fortelle}@mines-paristech.fr

Walter Godoy Junior
Federal University of Technology Paraná - UTFPR
Brazil
godoy@utfpr.edu.br

***Abstract*—In recent years, a wide variety of stakeholders have been working for the development of Intelligent Transportation System solutions. Cooperation among the various actors of transportation (vehicles, but also pedestrians, roads and infrastructure, traffic control centers, etc.) is seen as promising to enhance the efficiency of transportation and reduce its negative impacts (e.g., fatalities). However, it means that all communicating entities have to talk the same language, hence the need for Cooperative Intelligent Transportation Systems standards. There are now lots of standards being produced by standardization organization, e.g., International Standardization Organization (ISO) and European Telecommunications Standards Institute (ETSI) and there is a real need to understand how these standards can be implemented. This paper overviews the Intelligent Transportation System station reference architecture and presents a way of practical implementation of a toy Android application based on these standards as a proof of concept implementation. To our knowledge, this is the first implementation description compliant with these standards.**

***Keywords*—Cooperative Intelligent Transportation System; ITS; Standards; ISO; Wireless Networks.**

I. Introduction

Transportation systems are increasingly stressed all around the globe, especially in urban areas, and there is a clear need to optimize them. An Intelligent Transportation System (ITS) is seen as a solution to provide innovative services relating to different modes of transport and traffic management. The intelligence is brought by the ability of the system to react using sensors and information processing. Most of the ITS systems deployed today are autonomous in the sense that they are stand-alone and dedicated systems (e.g., traffic lights at an intersection, smart braking systems in a vehicle, etc.). According to the functionality and their purpose, the ITS applications can be classified in three primary categories [1]:

- **Safety:** Improve driving safety, e.g., preventing collision and accident reporting;
- **Efficiency:** Traffic monitoring and traffic management;
- **Infotainment:** Video streaming and Internet access.

Tee next step is to connect these systems through communication and having them cooperate, at least the two first mentioned above (safety and efficiency) [2] [3]. In recent years, a wide variety of stakeholders have been working for development of ITS communication, such as the CAR-2-CAR Communication Consortium gathering most of the European car makers and suppliers, the European Commission through several research projects ([4] [5] [6]), US DoT and Japan. Aside the need for efficient communication, despite limited bandwidth provided by physical carrier, one of the most important things is the interoperability, because system components can be developed by different stakeholders and ITS system shall support modular-based integration.

There are mainly two ways to ensure interoperability: industry standards or open standards produced by standardization organization such as ISO, ETSI, Internet Engineering Task Force (IETF), Institute of Electrical and Electronics Engineers (IEEE) or European Committee for Standardization (CEN). Our work is based on the second option and refers mainly to the common ITS architecture designed by ISO 21217 [7]. This architecture is the basis for several standards within ISO and beyond (ETSI and CEN notably) and a set — hopefully consistent — of ITS standards is under developments.

However, standards never look like a developers guide and they give some room for freedom in the way to implement. To the opposite, all standards are not produced by the same people and always result from compromises, so that there is no guarantee all standards are consistent even though great efforts are made to do so. Therefore, it is of high importance to understand what are the relevant standards for a given application, how they can be translated into functional components and what are the choices a designer of a cooperative application can safely do.

Based on a set of the ITS standards mentioned on Section II, this paper presents a way of practical implementation of the necessary ITS functions, why these set was chosen. We intentionally choose a very simple application (position sharing application) since the focus is not on the application part but on the underlying functions described by the standards: we refer to the implementation work that is "below" the application. To demonstrate this implementation, an Android application was created allowing several mobiles to exchange information (i.e., the location of each mobile). Within each mobile, an interface represents one's own location and the nearby mobiles' location. To our knowledge, this is the first

implementation based on the ITS Standards and it shows a proof of concept, demonstrating how ITS standards can actually be implemented.

The paper is organized as follows. Section II overviews the related ITS standards that have been used for this work. The potential system architecture is described in Section III; then, Section IV details the implementation of the ITS standard-compliant application on Android. After discussing the outcomes and potential issues of this development in Section V, Section VI concludes the paper and proposes future directions.

II. Relevant standards

The ISO 21217 standard *ITS - Communications access for land mobiles (CALM) - Architecture* [7] is fundamental for our application since it gives the reference frame for our implementation and the other ITS standards refer to it. It was prepared by Technical Committee ISO/TC 204, ITS subcommittee and describes the communications reference architecture of nodes called *ITS stations* designed for deployment in ITS communication networks.

Figure 1 shows the general ITS Station reference architecture, including interfaces between the various blocks with informative details.

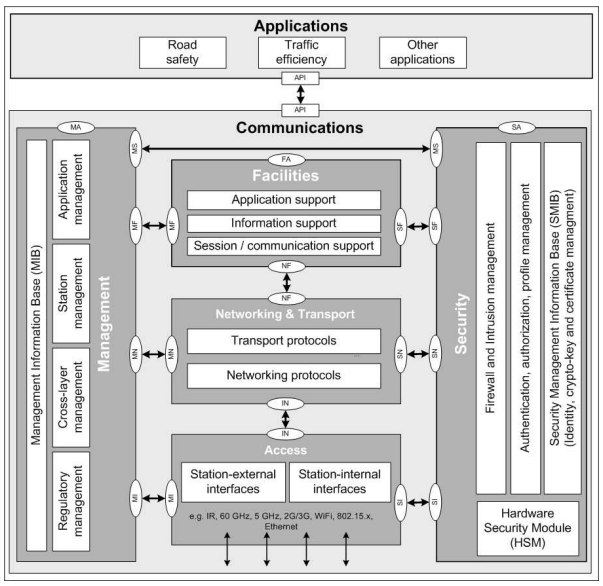

Fig. 1. ITS Station reference architecture [7].

The ITS architecture [7] is composed by: **"Access"** layer, comprised of OSI layers 1 (Physical) and 2 (Data Link); **"Networking & Transport"** layer, comprised of OSI layers 3 (Network) and 4 (Transport); **"Facilities"** layer, provides application, information and communication supports and it is comprised of OSI layers 5 (Session), 6 (Presentation) and 7 (Application); **"Management"**, a cross entity that containing station management functionality; **"Security"**, a cross entity that provide security services to others entities and **"Application"**, a horizontal entity, which provides Human-Machine Interface.

The ISO/TS 17423 standard *ITS Cooperative systems - ITS application requirements for automatic selection of communication interfaces* [8] is relevant for our application because it specifies the requirements which we will use (e.g, FlowType and transmission/reception Port Number). It relates to ISO 21217 describing the ITS application requirements for automatic selection of communication interfaces by *System Management* entity. To select this communication profile, the *System Management* entity uses the communication requirements, objectives of applications, communication protocol status, regulations and policies. The requirements are divided on five main classes: Operational, Destination, Performance, Security, and Protocol.

The ISO/TS 17419 standard *ITS Cooperative systems - Classification and management of ITS applications in a global context* [9] is used for our application to give the identifiers for each application process or entity. It illustrates and specifies global classification and management of ITS applications.

The ISO/NP 17429 *ITS - Cooperative systems - Profiles for processing and transfer of information between ITS stations for applications related to transport infrastructure management, control and guidance* [10] is necessary for our application since it gives us the procedure to exchange data between our mobile devices. It defines procedures useful to designers and developers of ITS applications exchanging data between ITS stations based on the ITS station reference architecture (ISO 21217).

The ISO 24102-3 standard *ITS Communications access for land mobiles (CALM) - ITS station management Part 3: Service access points* [11] is used for our application to implement the service access points. It specifies the management service access points between the entities and layers described by ISO 21217 (e.g., Management, Facilities, Access, etc.).

III. Design of a Test ITS Application

To evaluate how ITS standards can be implemented, we develop a toy ITS application on personal mobile devices in compliance with the ITS station reference architecture. We implement a simple *position sharing* application, tested with pedestrians, which sends its position and shows neighboring pedestrians' location on Android devices.

In this section, at first, we show the basic requirements of our application, then describe a number of design choices to adapt the application to ITS standards. Because of a wide variety of ITS standards, there are two essential decisions: (i) *which standards should be implemented to satisfy application's requirement*, and (ii) *how they can be adapted to actual implementation*. This section, therefore, explores a set of key ITS standards and design choices to implement them.

A. Application requirements

The proposed application is designed to detect nearby pedestrians with smartphones and/or tablets, and to detect and show the geographic position of surrounding devices.

To realize it, firstly, the application needs to obtain device's current location, e.g., from GPS. At the same time, it is necessary to detect nearby devices and exchange positions with each other. The received position information shall be time-stamped and validated, then displayed on a graphical interface. Regarding the communication, from the application's point of view, any type of access technology and protocol may be used as long as the location information can be exchanged.

In summary, the functional requirements of the proposed application are as follows:

- **Location management**: obtain device's own location
- **Discovery**: detect the presence of nearby devices
- **Communication**: exchange the current location between nearby devices
- **Database**: store and manage location information
- **Human Machine Interface**: display devices' location on map

B. Architecture design

To develop applications on the ITS station reference architecture [7], the first step is to identify which requirements should be inside application or other entities. The potential architecture designs are, therefore, as follows:

- **Application-based solution**: a traditional self-contained solution, in which each application contains all the necessary functions inside itself. Applications do not use the Facilities layer and management entity functions but, directly use the networking and transport layer features.
- **Facility-based solution**: most of the common functions are integrated into the Facilities layer and Management entity, while applications simply use them. In this solution, the discovery, communication, and database functions can be supported by ongoing ITS standards: *Generic message distribution handler*, *Communication profile handler / System management entity*, and *Local dynamic map*, respectively [10].

Although the application-based solution is simple, it lacks interoperability, extensibility, and incurs the cost of communication handling; it is inefficient that each application has redundant features, especially in hardware with limited capability. Furthermore, as the ITS station reference architecture accepts multiple access technologies and communication protocols, it is burdensome to support all of them in each application. A solution is to static configuration: exclusively use a certain type of protocol, however, it prevents to adapt dynamic mobile networks.

On the other hand, in the latter solution, most common functions are supported in the Facilities layer. Redundant development are minimized so that applications can concentrate on their specific task. Moreover, applications do not need to take care the diversity of the access technologies and communication protocols, because it is also handled by facilities layer services. Thanks to its efficiency of development and the adaptability of ITS standards, we adopt the facility layer solution to implement the proposed application, as depicted in Figure 2. The following sections detail the relevant components and their interaction.

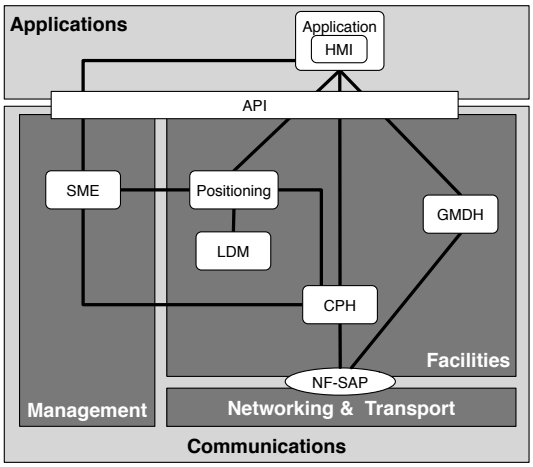

Fig. 2. Selected system architecture in compliance with the ITS-S reference architecture with the 5 conceptual components: Positioning, Communication Profile Handler (CPH), Generic Message Distribution Handler (GMDH), System Management Entity (SME) and the Local Dynamic Map (LDM).

C. Application

The application is composed of its main logic and Human Machine Interface; for this reason, it belongs the *Application* horizontal entity from ISO 21217. When the application is started, it requests position information to *Positioning* entity. Once the position is received, the application shows it on a graphical user interface (map) and communicates with *Communication profile handler* to send the Application Data Unit, which contains device's current position, timestamp, accuracy and its identifier.

D. Facilities and Management

As described previously, we implement five conceptual components in the Facilities layer and Management entity: *Positioning, Communication Profile Handler, Generic Message Distribution Handler, System Management Entity*, and *Local Dynamic Map*:

Positioning is an entity to process the device's position information. This entity provides application and information supports, for this reason it belongs to the Facilities layer. It complies to the ITS standards as a data source for *Local dynamic map* and *Application*, abstracting this entities from the diversity of the source of position (GPS, static configuration file, CAN Network, Internet, etc.). This way, these entities do not need to take care each data sources.

Communication Profile Handler (CPH) based on the ISO 17429 [10], it enables applications to abstract the diversity of communications. With this component, applications can transparently use multiple communication protocols and access technologies. Only applications need to do is to register their *communication requirements*, the type of communication, destination, quality, priority, etc. This communication requirement is mapped to each application, and then *Communication*

profile handler configures the underlying communication stack to satisfy the requirement.

Generic Message Distribution Handler (GMDH), based on the ISO 17429 [10], it is used to share a specific message among multiple applications by means of the publish/subscribe scheme. This scheme enables the push-based commutation, in which each receiver application *subscribes* a certain type of message while sender application *publishes* any message regardless of the presence of receiver. The message is delivered only when there are applications that subscribe to this message. We implement *Generic message distribution handler* to exploit its publish/subscribe mechanism for supporting information sharing and discovery.

System Management Entity (SME) this entity can be used by all entities. It enables cross-layer services by storing information from any communication layers [7]. In this paper, we implement *System management entity* to manage the communication profile as described by ISO 17423 [8].

Local Dynamic Map (LDM) is a database containing static maps, static information not yet part of the above maps, temporary and dynamic information and dynamic information concerning moving objects as defined by ETSI [12]. In this paper, we implement *Local dynamic map* as a data store of the position information, which can be requested by other applications. As described above, the position information is provided by the Positioning entity.

E. Interaction model

The proposed application and facilities communicate in compliance with the following operational steps: *flow assignment*, *position management*, *message transmission*, *message reception*, and *user interaction*.

In the flow assignment operation, the application presents its communication requirements to *System management entity*, then it generates and returns a *FlowTypeID*, an identifier to map the application to communication requirement [8] [13]. When the application wants to send messages, it registers the destination with previously-assigned FlowTypeID to *Communication profile handler*, which configure an appropriate communication stack and generates/replies a *FlowID* (an identifier of communication flow mapped to the application), its communication requirement, and destination. To transmits messages, the application passes message body to *Communication profile handler* with FlowID. The flow assignment is depicted in Figure 3.

To perform the position management operation, the application at first establishes connection with the *Positioning* entity, and then obtain the device's current position at any time. The *Positioning* entity also performs the flow type registration and flow registration as an application, because application may access the positioning entity in the remote host.

Once the *Positioning* is started, it communicates with *Local dynamic map* to search possible position information previously stored. If there is no position information, *Positioning* communicates with GPS to get them and store this position on

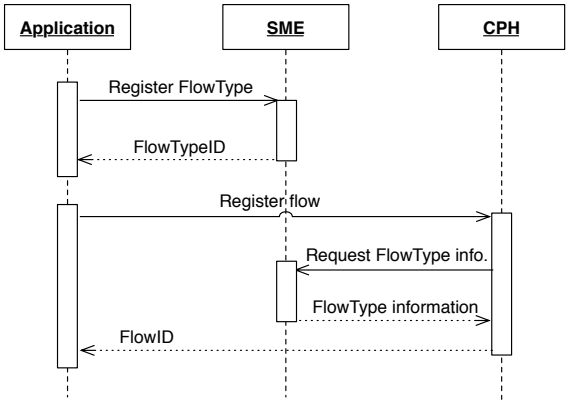

Fig. 3. Flow Assignment.

the *Local dynamic map*. In this paper, *Positioning* and *Local dynamic map* were specified as a pair of separate functions.

Once FlowID is assigned, the application sends its current position obtained from the above operation. In contrast to using traditional socket APIs, the application passes *Application Data Unit* (ADU) to *Communication Profile Handler* (CPH) with FlowID, then it publishes *Application data unit* to a specific destination. Figure 4 shows the message transmission operation. For the proposed application, the destination is single-hop broadcast.

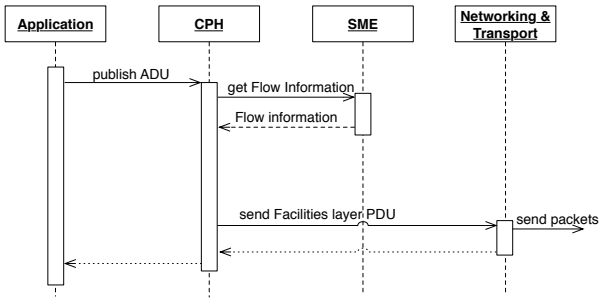

Fig. 4. Application Data Unit Transmission.

To receive location information, each application *subscribes* this message to *Generic message distribution handler*, then it distributes the message to applications only when the corresponding message is received. This communication follows the passive, push-based manner.

The application interacts with users via human machine interface to display its own location and the nearby mobile devices' location.

IV. Implementation

All the components are implemented using Android Software Development Kit (SDK), targeted to Android 4.0 or later. The application is a set of foreground Android activities, while the Facilities layer and Management entity components are *Android services*, application-independent background processes. To get position information, device's built-in GPS is used via Google play services library. As a Human-machine interface,

we use Google Map. Wi-Fi direct [14] is used for device-to-device direct communication.

A. Boundary of entities

Local dynamic map and Positioning are implemented as two independent Android services, while *System management entity*, *Communication profile handler* and *Generic message distribution handler* are a single service; because these three entities are dedicated for communication, we coupled them for performance reason (data can be more directly accessed among the entities). Note that such a decision, i.e., coupling the conceptual entities, does not affect the compliance to the standards: the definition of the entities in the standards are conceptual, therefore, how to couple the functions are developers' choice as long as it is compliant to the standardized interfaces.

Since each entity is stand-alone process, the interaction among the application and each component is performed as Inter-process-communication using Android Interface Definition Language (AIDL) [15]. This way, each facility provides APIs to uses as AIDL interface file; then, the users transparently use the provided functions via APIs without considering the inter-process-communication.

B. Communication

Although the traditional way to share information between devices is indirect communication using centralized hosts, we chose the device-to-device direct communication without servers because the intended scenario is transient communication among nearby pedestrians (mobile devices) which only requires single-hop broadcast. For this reason, Android's built-in Wi-Fi direct (called *Wi-Fi Peer-to-Peer* in Android), which enables to discover and connect to other devices, is used. The manipulation of Wi-Fi direct is implemented in the *Communication profile handler/Generic message distribution handler* service.

Regarding the flow assignment, in the current implementation, we introduce some experimental *well-known* FlowTypes: statically configured communication requirements stored in the local storage, i.e., the type of transport layer protocols and source/destination address and port number. These static settings are loaded when the *Communication profile handler* service is started, and then users specify one of the requirement by FlowTypeID (assuming well-known FlowTypes are publicly available a priori).

C. Application Programming Interface (API)

Each component provides a number of APIs to interact with applications. *Positioning* provides

```
messengerToServicePositioning.send(msg)
```

which returns the current position, where `msg` is an Android's *Message* object for inter-process communication, which contains description of the request from applications, such as the type of request and application's Identifier.

To manage position, *Local Dynamic Map* provides

```
messengerToServiceLDM.send(msgToLDM)
```

which stores or returns position requested by *Positioning* entity, where `msgToLDM` is an object containing request description, such as the type of request and ITS station's Identifier.

On the other hand, to send *Application data unit*, *Communication Profile Handler* provides:

```
publish(int flowId, List messageIds,
byte[] adu)
```

where `flowId` is an identifier of a destination mapped with communication requirements, `messageIds` and `adu` is the type and serialized sequence of bytes of *Application data unit*. How to encode and decode `adu` is application's responsibility.

D. Message format

The *Application data unit* object is composed of `latitude`, `longitude`, `accuracy`, `timestamps` and `stationID` attributes. To efficiently exchange data between devices, *Application data unit* is formatted, encoded and decoded according to ASN.1. We use *BinaryNotes* [16], an open source ASN.1 framework, to encode and decode the *Application data unit*.

E. Initial demonstration

We installed the application and services into three Android tablets (Samsung Galaxy Tab 10.1 GT-P5100, Android 4.1.2). Initial demonstrations have been performed using these devices, and shown each device properly exchanges its location information. Figure 5 shows the screenshot of the application displaying the location of neighboring devices.

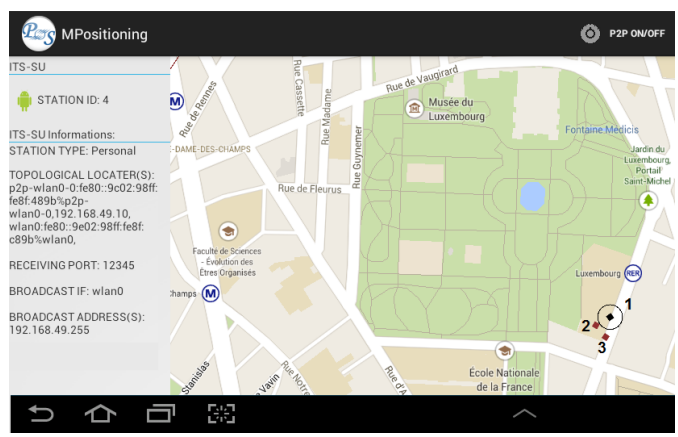

Fig. 5. Application's user interface displaying two neighbors.

In Figure 5, the device's own location is identified by *Station ID 1*, while the nearby nodes' locations are *2* and *3*, respectively.

V. Discussion

This paper demonstrated how conceptual entities defined by ITS standards can be implemented. As the ITS standards describe a framework to develop ITS applications, in this paper, we proposed a simple application to evaluate how applications and the underlying functions can be implemented,

and also their interactions. This section describes practical consideration of the implementation of ITS standard, and issues of ITS applications on Android.

Because standards describe minimum sets of essential features for interoperability, we have studied a number of design choices. A main choice is how to integrate a wide variety of conceptual elements in the standards into actual implementation. A simple solution is to make a single self-contained software component, while the other way is to actually separate each entities. In other words, it is the choice of single binary or multiple modules. In general, the former solution is superior in terms of performance: if we couple all conceptual entities into a single component, the interface between the entities are much simpler. However, this solution lacks extensibility and is difficult to maintain, specifically if it is developed by multiple stakeholders. In other words, the modular solution is efficient in terms of interoperability. The important design choice in this solution is the granularity of modules: implementing each conceptual module to exactly one component may need redundant interaction. The number of stakeholders and capability of target devices, therefore, should be carefully considered.

We used Android's WI-FI Direct for device-to-device direct communication; however, during the demonstration, we observed weaknesses of this technology: whenever a device is detected by other Wi-Fi Direct enabled devices, it requires users' interaction (a confirmation box is pop-upped and users need to tap the button to accept connection for each device). Although it is secure to prevent unwanted silent connections from unknown devices, it cannot be used by ITS applications which needs quick and automatic/silent connection establishment. It is necessary to investigate the way of secure ad-hoc communication without users' interaction.

In this paper, our first application did not concern privacy issues as specified by [7]. Since, identity information, such as a pair of device/application/user identifier and its position, should not be unnecessarily broadcasted over the air; as a next step it is necessary to integrate ITS security related functions e.g., authentication and encryption.

VI. Conclusion and Future Work

Cooperative Intelligent Transportation Systems is an increasingly important topic to enhance our stressed transportation systems and address some safety and efficiency problems. Based on the Internet OSI layered model, transportation and communication communities are designing a modular architecture intended to be deployed soon. Standards are being written and this will hopefully ensure interoperability of very different systems. The goal of this paper is to share the knowledge we got from the design and implementation of a simple application compliant with theses new ITS standards. We have shown that there are some challenges in the organization of the modules and the concepts associated. Since this is true for a very basic application, we expect more problems when real applications will be implemented, especially for safety critical applications. There is a clear need for clarification of the concepts and module to be used and this papers intends to pave the way in that direction.

However, these difficulties should not elude the most important result of our work: it is possible — and finally, not that difficult, if carefully handled — to implement a Cooperative ITS application using the best of the modular approach described in the standards. This validates the standardization effort. The direction is promising and need more exploration. Some directions are clearly shown by this paper: imagine a set of applications sharing the same services of the Facilities layer; refine conceptually the 5 components we introduced so that more applications can exploit them maybe introducing additional components; tackle the issues of the platform functions needed for ITS applications (Android had some drawbacks; but, is a platform to address, at least for pedestrians). Moreover other functions provided by standards are to be carefully linked to ITS application: encryption, privacy, etc. We hope to see in the near future more and more works describing how to best deploy the promising Cooperative ITS architecture.

References

[1] R. Michoud, A. M. Orozco, and G. Llano, "Mobile ad-hoc routing protocols survey for the design of VANET applications," Intelligent Transportation Systems Symposium (CITSS), 2012 IEEE Colombian, pp. 1 – 6, 2012.

[2] P. Muhlethaler, Y. Toor, A. Laouiti, and A. de La Fortelle, "Vehicle ad hoc networks: applications and related technical issues," IEEE Communications Surveys and Tutorials, vol. 10, no. 3, pp. 74–88, Quarter 2008, URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?isnumber=4625798&arnumber=4625806&count=7&index=6 [accessed: June 2014].

[3] P. Papadimitratos, A. de La Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, "Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation," Communications Magazine, IEEE, vol. 47, no. 11, pp. 84–95, November 2009.

[4] "IPv6 ITS Station Stack (ITSSv6) European project," URL: https://project.inria.fr/itssv6/ [accessed: June 2014].

[5] "Drive C2X European project," URL: http://www.drive-c2x.eu [accessed: June 2014].

[6] "Cooperative Vehicle-Infrastructure Systems (CVIS) European project," URL: http://www.cvisproject.org/ [accessed: June 2014].

[7] "ISO 21217:2014 Intelligent transport systems - Communications access for land mobiles (CALM) - Architecture," March 2014.

[8] "ISO/DTS 17423 Intelligent transport systems - Cooperative systems - ITS application requirements for automatic selection of communication interfaces," February 2013.

[9] "ISO/DTS 17419 Intelligent transport systems - Cooperative systems - Classification and management of ITS applications in a global context," February 2013.

[10] "ISO/NP 17429 Intelligent transport systems - Cooperative systems - Profiles for processing and transfer of information between ITS stations for applications related to transport infrastructure management, control and guidance," December 2012.

[11] "ISO 24102-3:2013 Intelligent transport systems - Communications access for land mobiles (CALM) - ITS station management - Part 3: Service access points," June 2013.

[12] "ETSI TR 102 893 V.1.1.1 Intelligent Transport Systems (ITS) - Security - Threat, Vulnerability and Risk Analysis (TVRA)," March 2010.

[13] "ISO 24102-1:2013 Intelligent transport systems - Communications access for land mobiles (CALM) - ITS station management - Part 1: Local management," June 2013.

[14] Wi-Fi Alliance, "Wi-Fi Direct," URL: http://www.wi-fi.org/discover-wi-fi/wi-fi-direct [accessed: May 2014].

[15] "Android Interface Definition Language (AIDL)," URL: http://developer.android.com/guide/components/aidl.html [accessed: June 2014].

[16] "BinaryNotes :: ASN.1 framework," URL: http://bnotes.sourceforge.net/ [accessed: May 2014].