Modeling Interoperability Channel using UPPAAL

Marwane Ayaida, Haytem EL Mehraz, Lissan Afilal and Hacène Fouchal

Centre de recherche CReSTIC
Université de Reims Champagne-Ardenne : UFR Sciences Exactes et Naturelles
Moulin de la Housse, BP 1039; 51687 REIMS Cedex 2, France
Telephone: +33 (0)3 26 91 81 74; Fax: +33 (0)3 26 91 31 06
Email: FirstName.LastName@univ-reims.fr

Abstract: In modern vehicles, there are a lot of devices (GPS, Tachograph, GSM / GPRS...) with different functions. These peripherals communicate with each other thanks to various media and protocols. The weakness of some protocols and standards obstructs the interoperability. They have to exchange useful information about the vehicle status and understand what is being shared.

This paper aims to design and to model a smart channel of communication and interoperability between heterogeneous embedded systems on transportation vehicles. This channel identifies connected peripherals and allows them to exchange data automatically with any kind of device. Furthermore, the smart channel builds services which can be customized by the end-user. The model has been proposed and specified thanks to the toolbox UPPAAL. With this toolbox we perform a simulation and verification (we considered critical situations of the system).

Introduction

Nowadays, communication technologies in the transportation area are growing quickly. Many devices are embedded on vehicles in order to control the engine (CAN, sensors...), to catch positions (GPS...) or to entertain (Wi-Fi...). To be more powerful, each peripheral focuses on its core functionality and handles a part of the vehicle information (Position, Wheel pressure, Speed, Fuel Level...). Thus, the peripherals require to communicate and to exchange data in order to meet user needs which become more and more strict. This communication is complex due to the divergence between protocols and interfaces used by peripherals. Consequently, it is hard to satisfy the end-to-end interoperability even whith a standard. Most standards specify only the way to share data and do not handle neither the semantic nor the processing issues. Generally, data pass through bridges or ad hoc transceivers.

In order to reduce the allocated resources (hardware and software), we suggest to manage this communication into a single and a smart channel. This channel will be able to identify a device when it is being connected. In addition, the channel will allow devices to communicate and to exchange information in order to build new and efficient services for end-users. These services will be launched automatically after the connection of a new

device. Then, the user may customize its services online. This channel has been specified and verified thanks to the UPPAAL toolbox since it is a powerful tool used to design, to simulate and to verify important properties of systems. Its usual model is a Timed Automata Networks. These automata communicate and synchronize in order to express the functioning of all entities involved in the channel.

The paper starts by presenting the lack of interoperability between embedded devices in transportation and the standards weakness to guarantee it. Then, it presents the channel model details and describes some important properties to be verified. The paper ends by a conclusion and a presentation of the future works.

1 Standards & Interoperability

1.1 Interoperability between Embedded Devices

There are several devices embedded on vehicles. They are split into four groups :

1. CAN/FMS systems : are able to communicate via CAN-Bus within the vehicle. Most sensors on vehicles use this protocol. Tailor-made for transportation, the Tachograph is an example which records the vehicle speed and whether it is moving or not. It became mandatory in trucks and buses on 1986 in Europe for public safety.
2. Communication systems : allow the transportation vehicle to send or receive data, from its company. Actually, most trucks are equipped with GPRS connection which allows the driver to send and receive useful information.
3. Positioning systems : help the driver to find his way and companies to track their fleet in real time (GPS...).
4. Security systems : focus on protecting drivers such as gas detector, RoadBox (a camera which records events 14s before and 6s after an accident)...

The main purpose in the design of such devices is the ability to share useful information when connecting them and building new services to the end-user. This principle is called "Interoperability".

Thus, the interoperability is not only the ability of two heterogeneous systems to exchange data, but also the understanding of what is being exchanged and the capability of interaction and joint execution of tasks [YCBL03].

Peripherals communicate using many different standards and protocols. If they need to exchange data, they have to pass through bridges for the translation between protocols. Furthermore, some standards can not guarantee end-to-end interoperability. So, devices which are in conformance to a standard may have some troubles to communicate [TM03].

1.2 Interoperability Weakness

Despite multiple attempts of standards to guarantee the end-to-end interoperability, most of them are not sufficient to attend it [LMSW08].

1.2.1 Standards Contributions

Standards have a major contribution to satisfy the first and second levels of interoperability. Actually, we can not imagine that we need different browsers to attend different Web sites thanks to standards like TCP/IP, HTML...

Standards, when they are well adopted, enable more abstraction of lowest levels to developers and offer them less difficulties to build their programs regardless of the hardware platform on which their programs will be implemented. So, they can focus on their added value which is the core functionality of their service or the program they are building.

1.2.2 Standards Failures

It is too difficult to develop a language rich enough to cover a specific domain targeted by a standard. Standards also neglect some aspects when systems interoperate. As an example, we can talk about QoS (Quality of Services) which describes characteristics of running systems (response time, security, performance, availability...). Few standards require a threshold time response in communication between systems which may lead to incompatibility and prevent the interoperability.

In addition to that, standards are specifications and requirements. So, they are useless until the transformation into final products or services. This transformation may involve with customization and then leads to trouble to interoperate. Standards are useful when they are well adopted. But, like any other technology, they have a life cycle and will become obsolete, by evolving to a non backward-compatible version or by using alternative standards. Some standards become widely used because they were the first ("de facto") standards and not necessarily the best ones. Unfortunately, organizations cannot economically explain migration to better standards and are often locked into such standards. There are also bad standards (under-specified, over-specified, unstable...) and conflicting standards because they are mutually exclusive or competing. A well-known example of incompatible and competing standards are HD-DVD and Blu-Ray.

2 C2A : Connect to All System

2.1 Origin of the Idea

As we mentioned in **1.1**, there are a lot of devices embedded on vehicles. These devices follow different standards which are not necessarily compatible as we have seen in **1.2.2**.

Thus, the idea was to design a smart channel. This channel is shown in Figure 1.

Figure 1: Connect to All system : a smart channel which allows embedded devices to communicate and exchange data.

Connect to All (C2A) system must not only be able to identify the peripheral when being plugged in, but also, it has to allow it to communicate. The system scans continuously all ports to detect the connection of devices, it tries to identify it (storage media, GPS...). Then, it looks for the communication parameters of this device. After this, the channel makes use of the data provided by the peripheral or looks for other peripherals which will be able to share data with it.

2.2 Modeling : First Step in Design C2A System

We present a model about C2A smart channel. Then, we will check the validity of the model using the UPPAAL tool box which is a well-known model checker [BDL04].

2.2.1 Introduction to UPPAAL

UPPAAL is an integrated tool environment for modeling, validation and verification of real-time systems (real time controllers, communication protocols, multimedia applications...) modeled as networks of timed automata. It was named UPPAAL with UPP standing to Uppsala University in Sweden and AAL for Aalborg University in Denmark which develop it. The first version was proposed in 1995. It consists of a graphical description tool, a simulator and a model checker. It serves as a modeling or a design language to describe a system behavior as networks of automata extended with clocks and data variables. It has been used for the simulation of the system and checks if there is any error on the system. Properties -to be verified- can be specified using a subset of CTL (computational tree logic) formalism.

2.2.2 Definition of Network Timed Automata

A Timed Automaton (TA) is a finite-state machine (FSM) extended with clock variables. Clock variables evaluate to real numbers and all of them progress synchronously. A sys-

tem is modeled as a parallel composition of TA. An automaton may perform a transition separately or synchronizes with another automaton (channel synchronization).

More precisely, a TA is a tuple (L, l_0, C, A, E, I) , where :

1. L is a set of locations
2. $l_0 \in L$ is the initial location
3. C is the set of clocks
4. A is the set of actions (e.g. press!), co-actions (e.g. press?) and internal τ-actions
5. $E \in L \times A \times B(C)^1 \times 2^C \times L$ is a set of edges between locations with an action, a guard and a set of clocks to be reset
6. $I : L \longrightarrow B(C)$ assigns invariants to locations

A Timed Network Automata (TNA) over a common set of clocks and actions consists of n TA $(L_i, l_i^0, C, A, E_i, l_i)$ with $1 \leq i \leq n$.

2.2.3 The Query Language

UPPAAL uses a simplified version of CTL as its query language. The query language consists of state formulae and path formulae. A state formula describes individual states. It could be written as expressions (x>3, i==2, x<=3, i==5...), location (porcess.location) or deadlock (there are no enabled transitions). For path formulae we use four symbols : E (exists a path), A (for all paths), [] (all states in a path) and <> (some states in a path). The following combinations are supported : A[], A<>, E<>, E[]. A path formula quantifies over paths of the model. Let p and q two states formulae. The properties that can be verified are (Figure 2):

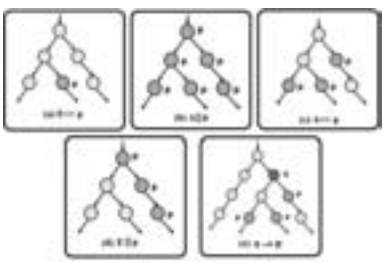

Figure 2: Queries in UPPAAL

1. E<> p - "p Reachable" : it is possible to reach a state in which p is satisfied (p is true in -at least- one reachable state).

[1]Boolean guards $B(C)$ are defined as follows : $B(C) = true \parallel$ false $\parallel x \bowtie c \parallel x - y \bowtie c \parallel B(C) \wedge B(C) \parallel B(C) \vee B(C) \parallel \neg B(C)$ where : $x, y \in C$, $c \in \mathbb{N}$, and $\bowtie \in \{<, \leq, =, \geq, >\}$.

2. A[] p - "Invariantly p" : p holds invariantly (P is true in all reachable states).

3. A<> p - "Inevitable p" : p will be inevitable become true : the automaton is guaranteed to eventually reach a state in which p is true (P is true in some states of all paths).

4. E[] p - "Potentially Always p" : p is potentially always true (There exists a path in which p is true in all states).

5. q $\dashrightarrow$ p - "q leads p" : q will inevitably leads to p (p is true in -at least- one reachable state in all paths from q).

2.3 The Model Description

In this section, we will present the TNA of the C2A model and the synchronization events between automata. For the model needs, we choose to classify the peripheral-devices in three groups according to the way of sharing data as follows :

- Device IN : These devices broadcast continuously or after a request from the system. Generally, there are in read only mode. Data are sent directly or via transceivers. Here, we call "Device IN" the peripheral and the transceiver. For example, CAN/FMS Bus, Sensors, GPS, RFID Reader...
- Device OUT : These devices consume the data sent by the system. As an example we can use Flash Memory, Hard Disk, USB Flash Drive, Loudspeaker...
- Device INOUT : These devices are smart for communication. They open a channel, whether the system is source or destination for sharing data outside the system.

Also, the peripheral-devices -when plugged to C2A- drive different types of information (Speed, Driver's State, Fuel Level, Time, Position...), denoted as "Signals". Every Signal has a defined value at any time fixed by a peripheral or the system. All signals describe the vehicle status. The model is shown on the Figure 3.

The groups of peripherals are modeled by three processes. The smart channel C2A is divided into six processes (Initialization, Identification, Update PerphTable, Signals Update, Data Loader, Data Management). The blue arrows represent the signals and the red ones represent the synchronization events between different processes within the C2A smart channel.

C2A identifies the group of connected devices. Then, it acquires the signals provided by the first and/or the third group. After data processing, it dispatches them to the second and/or the third groups. The signals configuration is saved on the system and can be changed by some peripherals known as "Configuration devices". So, the user can customize services by choosing signals and the processing results outputs. In this study, we choose to model only four signals (Time, latitude, longitude and number of satellites provided by a GPS receiver) and two services (data logging in a Hard Disk and sending data

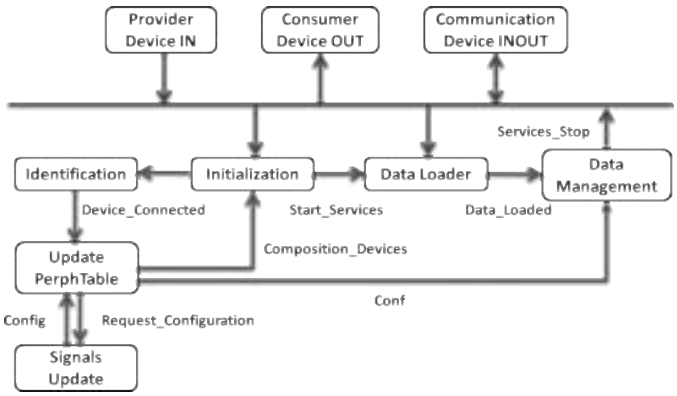

Figure 3: Model of the smart channel C2A

via GPRS connection). As a lot of peripherals in transportation use a serial protocol (USB, RS232...) or may have an interface with a transceiver, we choose to model the connection of such peripherals. Every device connect to a serial port of the system and communicates with a unique Baud Rate.

First of all, the channel detects the connection thanks to the process Initialization. This process launches the process Identification to recognize the connected device by identifying the port and the Baud rate of the connection. Then, the process Identification sends 'Device_Connected' to the process Update PerphTable which adds it to the table of connected devices. After sending 'Request_Configuration' to the process Signals Update, the process Update PerphTable waits for 'Config'. The configuration will be the stored one or given by a configuration device after the verification of the device security parameters. When the process Initialization receives 'Composition_Devices', it has all the information needed about the service to launch. It sends 'Start_Services' to the process Data Loader which collects the useful data from input devices. At its end, it sends 'Data_Loaded' to the process Data Management which performs the appropriate processing (conversion, thresholding...) on these data and sends the result to output devices. Finally, it sends 'Services_Stop' to all other processes to restart them and waits for the next service.

The rest of the section present -as an example- two automata modeling device for communication and the process identification. All other processes were modeled in the same way.

2.3.1 Device INOUT Model

The model is shown on Figure 4. Those peripherals are smart enough to send or receive data. But, they can not execute both of them in the same time. When the device is plugged, it keeps waiting in 'Wait_Detection' state for a detection from the system (sending the synchronization order 'Detection_IN' to the system). Then, it waits for sending Signals or receiving data from the system on the state 'Wait_Send_Reception'. The sent

signals are available when receiving the synchronization event 'DeviceDataToSystem'. The sending of the data finishes in the state 'End_Sending' before returning to the state 'Wait_Send_Reception' for another sending or reception. The event 'Detection_OUT' launches the reception of the signals synchronized by the event 'DataSystemToDevice' in the state 'Reception_Data' from the system to save or to send them outside. The reception of the data from the channel finishes in the state 'End_Reception'. Finally, it returns to the state 'Wait_Send_Reception' when receiving 'Services_Stop' event from the process Data Management and the device waits for other data.

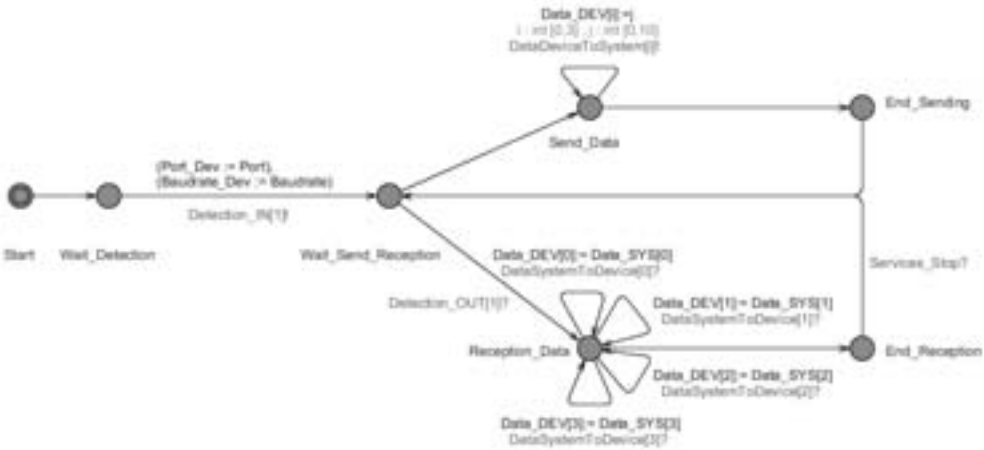

Figure 4: Automaton modeling Device INOUT

2.3.2 Process Identification Model

The model is shown on Figure 5. After receiving 'Device_Connected' from the process Initialization, this process identifies the peripheral after its connection by finding the correct port and Baud rate. As an example, we suppose that the C2A system has three ports which communicate with three different Baud rates. It opens the port 1 in the state 'Port_1' with Baud rate 1 in the state 'Baudrate_1', then with Baud rate 2 in the state 'Baudrate_2' etc... The process opens each port with all Baud rates. If it receives a valid data it sends the scan result 'Device_Identified' to the process Update PeriphTable to add this peripheral to the table of connected devices.

2.4 Simulation & Verification

The simulation aims to check whether the system works as expected. We choose to instantiate an USB Drive as a configuration device, a GPS module as Device IN, a Hard Disk as a Device OUT, a GPRS module as a Device INOUT and C2A processes. We checked that

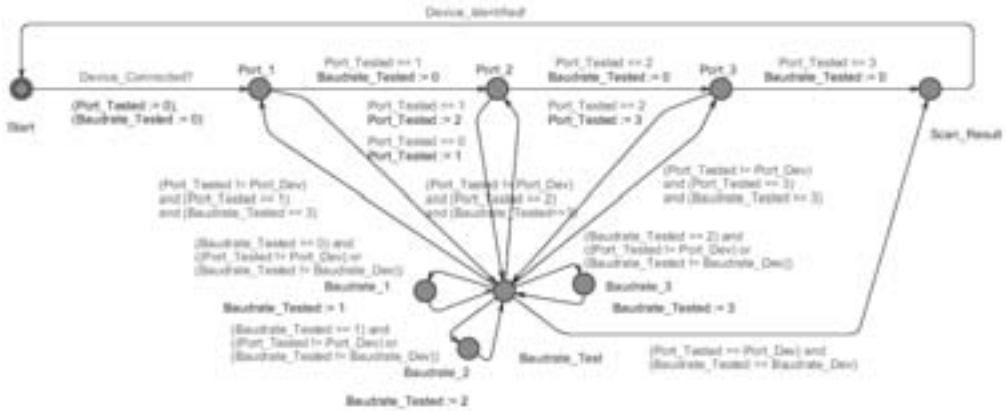

Figure 5: Automaton modeling process Identification

the system works properly using step by step and random simulations. We also verified the critical properties of the system. Some of these properties are described in the Table 1.

An example of property number 6 in the table 1 is that all connected devices must be identified (which gives in UPPAAL : "Identification.Scan_Result -> Identification.Baudrate_Tested == Baudrate_Dev and Identification.Port_ Tested == Port_Dev"). This means that the system finds the right port and Baud rate for the connected device. This allows the reception of valid characters for the recognition of the group and the type of connected device. Therefore, these settings will be used later to establish communication with the device.

We observe here that the simulations and the properties verifications confirm the validity of the proposed model (over 20 millions of state transitions tested). UPPAAL allow us to verify properties but suffer from several limits. One limit is that the events used for FSM synchronization are exclusively binary channels which can't carry useful data. Other limit is that the proposed model can't be translated in an executable code to measure real performances.

3 Conclusion & Future Works

There are a lot of embedded devices in vehicles. They have various interfaces and protocols which are in general incompatible. The goal of this work is to reduce these differences by designing a smart channel C2A which supports the most important wired and wireless peripherals. This solution could be very useful since it reduces hardware and software resources in a vehicle and increases the compatibility between heterogeneous systems. This compatibility makes the combination of peripherals possible to offer new functions to users.

N°	Properties	Comments	Verification	Critical
1	A[] not deadlock	System is deadlock free	Satisfied	Yes
2	P_Identification.Scan_Result –> P_Identification.Baudrate_Tested == Baudrate_Dev	The system identify the right Baud Rate of the device	Satisfied	Yes
3	P_Identification.Scan_Result –> P_Identification.Port_Tested == Port_Dev	The system identify the right Port of the device	Satisfied	Yes
4	A<> ((USB.Send_Configuration and P_Signals_Update.Config_Sent) imply Conf == Conf_IN)	The system takes the input configuration and not the stored one	Satisfied	NO
5	A<> (USB.Send_Configuration imply P_Signals_Update.tempo <= 3)	The input configuration is catched in less than 3 time units	Satisfied	NO
6	A<> (P_Initialization.Detection imply P_Identification.Scan_Result)	The connected device will be identified	Satisfied	Yes
7	A<> (GPRS.Send_Data imply P_Data_Loader.Data_Acquisition)	All data sent by GPRS module will be received	Satisfied	Yes
8	A<> (GPS.Send_Data imply P_Data_Loader.Data_Acquisition)	All data sent by GPS module will be received	Satisfied	Yes

Table 1: Properties verification

In the model presented above, the system detects devices connections. The actual work aims at improving the model by identifying when devices are disconnected and protecting the system of resulting bugs. Furthermore, the implementation of this model was coded in C language and works properly using an USB Drive, a CAN/FMS frames simulator, a GPS and a GPRS modules. This should be extended to support other devices and protocols (CAN, Bluetooth, Wi-Fi, ZigBee, RFID...) giving access to new services. In the future, we aim to model C2A system with Specification and Description Language (SDL) which can be translated automatically into executable C language after simulation and verification.

Bibliography

[BDL04] Gerd Behrmann, Re David, and Kim G. Larsen. A tutorial on uppaal. In M. Bernardo and F. Corradini, editors, *International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004. Revised Lectures*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–237. Springer Verlag, 2004. Freely available at http://www.uppaal.com/.

[LMSW08] Grace A. Lewis, Edwin Morris, Soumya Simanta, and Lutz Wrage. Why Standards Are Not Enough to Guarantee End-to-End Interoperability. In *ICCBSS '08: Proceedings of the Seventh International Conference on Composition-Based Software Systems (ICCBSS 2008)*, pages 164–173, Washington, DC, USA, 2008. IEEE Computer Society.

[TM03] Andreas Tolk and James A. Muguira. The Levels of Conceptual Interoperability Model (LCIM). In *Proceedings IEEE Fall Simulation Interoperability Workshop*. IEEE CS Press, 2003.

[YCBL03] Paul Young, Nabendu Chaki, Valdis Berzins, and Luqi. Evaluation of Middleware Architectures in Achieving System Interoperability. *Rapid System Prototyping, IEEE International Workshop on*, 0:108, 2003.