A poor man's VANET channel sounder

Hervé Boeglen, Benoît Hilt
Laboratoire MIPS
EA 2332
Université de Haute Alsace, France
herve.boeglen@uha.fr

Jean-François Cailbault, Rodolphe Vauzelle
Laboratoire XLIM-SIC
UMR CNRS 6172
Université de Poitiers, France

Abstract— **This paper describes how it is possible to design a low-cost Vehicular Ad hoc NETworks (VANET) channel sounder using a popular Software Defined Radio (SDR) equipment. We explain in details the main steps we followed in order to obtain a fully 802.11p compliant transceiver. We then show preliminary results obtained during a VANET measurement campaign. We also underline the limits of this approach which are mainly fixed by the SDR equipment used.**

Keywords: VANET, channel modeling, Software Defined Radio

I. Introduction

In recent years, Vehicle-to-Vehicle (V2V) wireless communications have received a lot of attention as they are going to be a crucial issue in Intelligent Transportation Systems (ITS). In particular, different applications will emerge enabled by the exchange of information between cars. The main ones concern the enhancement of road safety and the reduction of the traffic impact on the environment. In the near future, this technology is expected to allow the setting of car networks called Vehicular Ad-Hoc Networks (VANETs). In order to transmit information reliably on rapidly changing vehicular channels one has to rely on a robust physical layer. This is precisely the challenge the 802.11p working group was facing in designing a physical layer standard for V2V communications [1]. This physical layer has to be evaluated by means of real-world measurements but also by means of less costly simulations implementing realistic channel models. Finding an accurate channel model for VANETs is still a research issue since vehicular wireless channels exhibit specific characteristics that makes them quite different from the very well characterized mobile telephony channels [2]. One of the major issues when using simulators for VANETs concerns the vehicular environment and therefore the realistic modeling of the wireless propagation channel. Classical network simulation tools like ns-2 or ns-3 are not accurate for VANET simulation since they provide too simplistic propagation models. In reference [3], we propose a semi-deterministic VANET channel model which was integrated into ns-2. This model is a compromise between a highly accurate ray-tracing propagation simulator requiring a very high processing time and a 3G mobile telephony statistical channel model having a low processing time. Our simulations showed a good agreement with the results compared to the ray-tracing propagation simulator (taken as a reference). To go a step further we decided to perform real world measurements by to setting up a 802.11p platform which would serve as a channel sounder. Using commercial 802.11p card cannot suit our needs since they are made of chips which do not allow accessing the OFDM IQ vectors. Of course we could have used specialized equipment like the RUSK channel sounder [4] but we did not have the budget. In fact, we could only afford to spend 4500€ for this project. This is precisely what this paper tries to address: is it possible to build a valuable VANET channel sounder with such a small budget? The rest of the paper is organized as follows. Section II describes the electronic equipment we have chosen for this project. Section III deals with the software we have designed to interface with the electronic equipment. Section IV presents the organization of the measurement campaign and shows some results we have been able to obtain. Finally, section V concludes the paper.

II. The hardware

The evolution of computers' processing power following Moore's law has led to the concept of the Software Defined Radio (SDR). The term was first coined in 1992 by J. Mitola [5]. A SDR radio is a device which samples the desirable signal coming from the antenna after a suitable band selection filter. We can refer to a transceiver as a software radio if its communication functions are realized as programs running on a suitable processor. One of the major advantages of a SDR is its reconfigurability. By simply changing the software, one can adapt the system to different standards. This feature can be fully utilized to design a 802.11p compliant physical layer. Constrained by our limited budget we finally decided to buy two USRP2 SDRs from Ettus Research [6]. These relatively low cost SDRs are among the most popular ones and have the advantage that they can be equipped with several RF daughterboards covering a large part of the radio spectrum. The USRP2 core is a low cost Xillinx Spartan 3E FPGA (see Figure 1).

The code inside the FPGA is built around a 32 bit soft processor which controls 4 IQ baseband transmit and receive chains. The transmit chain uses 400MS/s 16 bits DACs whereas the receive chain relies on 100MS/s ADCs. The data are transferred to and from the computer via a Gigabit Ethernet (GiE) interface. One problem we have faced with USRP2 is the fact that the FPGA is used at 95% by the baseband transceiver code. This means that the system cannot be completed with user designed signal processing functions.

For the RF part, XCVR2450 daughterboard was chosen since it allows covering the VANET frequency band around 5.9GHz. We had to check first by a link budget analysis that the transceiver range was enough for our VANET applications. The transmitter power is 18dBm at 5.9GHz. The receiver is based on Maxim's MAX2629 IEEE802.11a compliant chip. The transmitter and the receiver antennas have a gain of 3dBi. The calculation gave us a maximum range of 500 meters for 16QAM in a Line Of Sight (LOS) situation. When considering a fading margin of 20dB this range falls to 100 meters. According to these figures a set of 2 USRP2 can therefore be used for 802.11p transmissions. The USRP2 comes with an open source software called Gnuradio. It is with this API that we designed our 802.11p compliant transceiver. The steps we followed are described in the next section.

Figure 1. Ettus Research USRP2

III. THE SOFTWARE

The Gnuradio API is based on two languages. Python is in charge of setting up the signal processing flow graph forming the transceiver chain and for the synchronization between blocks. The signal processing blocks which require a high processing power are written in C++. The relation between Python and C++ is made by SWIG [7].

There already exist a lot of applications running on USRPs in several domains ranging from ZigBee to GSM. There is also a 802.11p transmitter application designed by Fuxjäger et al. [8]. We have used this implementation as a starting point for our design.

The IEEE 802.11p physical layer has similar specifications as IEEE 802.11a with some changes. In IEEE 802.11p, a 10MHz frequency bandwidth is used, instead of 20MHz bandwidth in IEEE 802.11a, thus all parameters in the time domain for IEEE 802.11p are doubled compared with the IEEE 802.11a (see Table 1). As frequency selectivity is a characteristic of VANET channels, 802.11p uses OFDM.

When implementing an 802.11p transceiver, the best method we found was to use Annex G of the standard [9]. It is a complete example of a packet encoding which gives the results of the encoding at the end of each block ending up with the baseband IQ vectors. Most of the blocks shown in Figure 2 for the encoder where designed in C++ using the IT++ library [10].

We did the same for the corresponding blocks of the decoder. At this point, we had everything needed to simulate a 802.11p transmission. But in order to perform a real transmission, Automatic Gain Control (AGC) and synchronization at the receiver is needed. Before being able to decode a packet a 802.11p receiver has to accomplish the tasks presented in Figure 3.

TABLE I. IEEE 802.11P KEY OFDM PARAMETERS

Number of data subcarriers	48
Number of pilot subcarriers	4
Subcarrier frequency spacing	156.2kHz
Occupied Bandwidth	8.28125MHz
Short training sequence duration	16μs
Long training sequence duration	16μs
Training sequence guard interval	3.2μs
PLCP preamble duration	32μs
Guard interval duration	1.6μs
OFDM Symbol duration	8μs

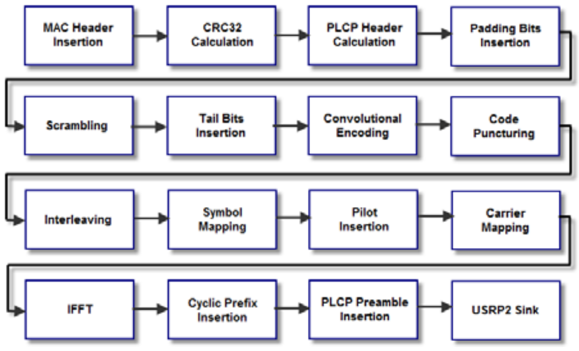

Figure 2. 802.11p encoder blocks

OFDM synchronization is a complex task. Many algorithms for timing and frequency synchronization have been proposed in the literature. Most of them are correlation based algorithms making use of the short and/or long preamble of the OFDM packet. For a complete treatment of this subject we refer the interested reader to the following references [11][12], for these are the one we found the more useful. Figure 4 gives an example of the usage of the synchronization algorithms found in references [11] and [12].

At this point of the project, all the receiver algorithms were tested successfully but we reached the processing limits of our PC. At rates up to 1MS/s everything went fine but there was not possible to work at 10MS/s. To solve this issue there are two solutions. The first one is to tune the IT++ code in order to make it faster (IT++ code has not been designed for real time applications) using a tool like Oprofile [13]. The second consists in placing some of the algorithms (namely the AGC and the synchronization algorithms) in the FPGA. This is not possible with the USRP2 since there is not enough room left in the FPGA but it is possible with a newer version called USRP N210. Does it mean that we cannot use this equipment as a VANET channel sounder? No, of course. Here is how we overcame the problem. For a channel sounding application, we only need to capture the signal after it has passed through a 10MHz bandpass filter and the AGC. The synchronization and the decoding of the packets can therefore be performed off line. Let us now put the system into action and describe the measurement campaign we performed with it.

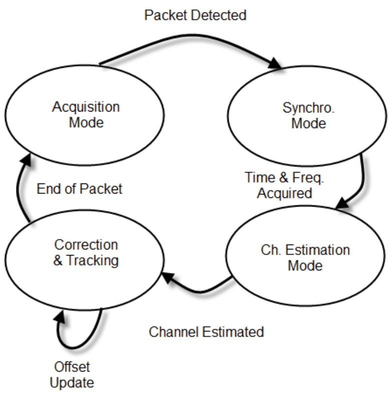

Figure 3. 802.11p receiver modes

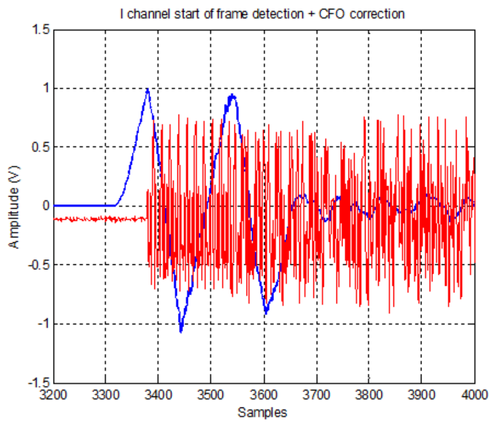

Figure 4. Example of coarse time and Carrier Frequency Offset (CFO) correction using the algorithms found in [11] and [12].

IV. The Measurement Campaign

A. *System setup*

The campaign took place during the last week of October 2011. We could benefit of two instrumented cars owned by the MIAM team of our lab which is specialized in car automation systems. These two cars are equipped with a dSpace Autobox [14] linked to an embedded PC running Windows XP. The Autobox system can monitor all the car parameters such as speed, GPS coordinates etc. As our Gnuradio 802.11p implementation requires Linux, we had to bring two more PCs (laptops), one for the transmitter and the other for the receiver in charge with the packet recording. The two USRP2 were fixed on the car roofs by a magnetic system (see Figure 5).

Concerning the radio communication side, the transmitter was continuously transmitting data packets at 5.86GHz at maximum power (i.e. 18dBm). Two types of packets were sent according to the scenarios tested. It is well known in the VANET community that on highways the channel coherence time Tc can be as low as 0.3ms [15]. This means that when the OFDM packet exceeds this time (typically for a packet longer than 200 bytes; the maximum possible being 4095 bytes), the packet suffers time selectivity. In order to check this effect, we used a long packet on highway scenarios (Tp = 1.16ms, 143 OFDM symbols coded in QPSK at 9Mbits/s). For the other scenarios a short packet was used (Tp = 88.1µs, 9 OFDM symbols coded in QAM16 at 18Mbits/s). In both cases, we leave a 0.13ms space between packets.

Figure 5. The cars used for the campaign

B. *Tested scenarios*

Before starting with the actual VANET scenarios we had to find a safe area for system check. We found it near Mulhouse with an abandoned military airbase rescue runway of 2km long. This site was also suitable to test suburban scenarios. During the week we successfully tested 12 typical VANET scenarios we identified in Urban, Suburban and Highway situations. Figure 6 shows an example of a Highway scenario called A2 where the two vehicles run at 110km/h on opposite sides of the road.

At the end of the week we had collected more than 150GB of data for the different scenarios for which the duration ranged from 20s to 2mns. The acquisition data rate was 10MS/s which was the maximum we could reach with the USRP2.

C. *First results*

We are currently working on analyzing this big amount of data but we can already show some interesting features of our “poor man’s channel sounder”. The main idea is to use the OFDM symbols sent (data + pilots) to extract the channel frequency response H(f,t) (after IFFT) and to obtain the channel impulse response h(τ,t). Figures 7 and 8 show an example of these data extracted from a long packet transmission. One can observe that we are clearly in a strong LOS condition since the fading level observed is quite low.

V. Conclusion and Future Work

We have presented a low cost channel sounder for VANET which has been used in a one week measurement campaign. From the primary idea to realization we faced quite a lot of problems most of which we solved sometimes the hard way. The things we learned in this experience can be summarized by the following points:

- SDR technology is a mixture of digital electronics, signal processing for communication and software engineering. It is difficult to master all these domains.
- Writing software for real time applications requires specific skills. Implementing real time algorithms in fixed point arithmetic in an FPGA is also not trivial.

- SDR power is limited by the PC used. Although today's PCs are quite powerful, it is difficult to implement in real time all the algorithms needed by a high rate IEEE 802.11 standard (AGC, synchronization, channel estimation and decoding).
- One way to overcome the processing power problem is to implement some of the algorithms in the FPGA (i.e. AGC and synchronization). So be careful when you choose your SDR platform! If you plan to use Ettus Research products go for the USRP N210 product which has a larger FPGA than the USRP2.
- Ettus Research SDR solution (USRP + Gnuradio) Achilles' heel is definitely the poor quality of the documentation. As is usual with free software, you get regular updates of the FPGA firmware and the Gnuradio software which causes annoying compatibility problems. Their so called Universal Hardware Driver (UHD) which is supposed to work with all USRP platforms can sometimes have peculiar behaviors. The fact that National Instruments took over Ettus Research lets us expect that there is going to be an improvement on these points in the near future.
- Do not expect to transfer data between USRP2 and the PC at a sustainable rate over 10MS/s. Although data are transferred via a GiE interface, IQ data is coded on 32 bits (16bits for each I and Q channel). A higher rate would have been much better for channel sampling.

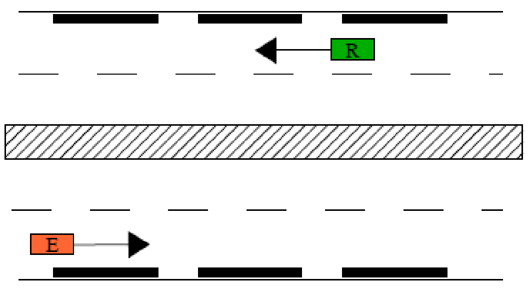

Figure 6. A2 Highway scenario

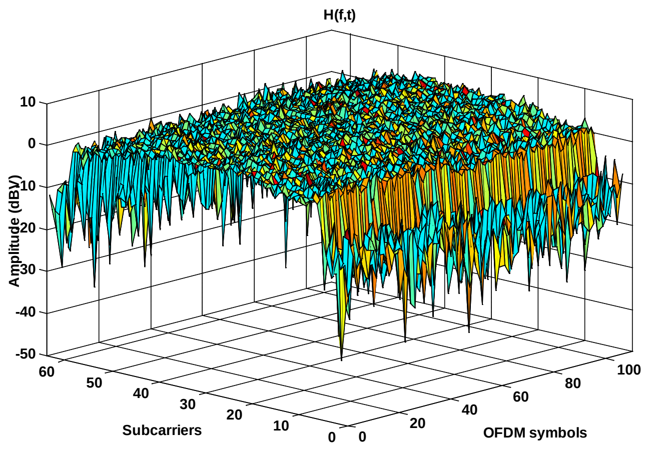

Figure 7. Evolution of H(f,t) over the transmission of a long packet (A2 Highway scenario).

We expect that these remarks are going to be useful to people wanting to follow the same approach. We are now going to concentrate our work on the measurement data. We expect to extract enough information (using a statistical approach) to be able to set up some typical VANET channel models to be integrated in our ns-2 software platform. These will be, of course, made available to the community.

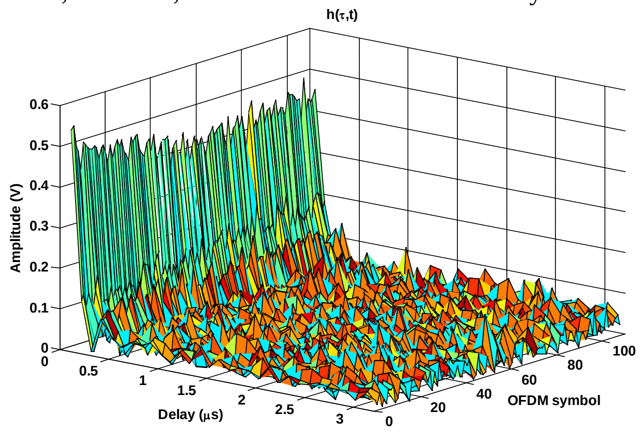

Figure 8. Evolution h(τ,t) over the transmission of a long packet (A2 Highway scenario).

ACKNOWLEDGMENT

The authors acknowledge the support of the following persons of the MIAM team from the MIPS laboratory: Michel Basset, Jérémie Daniel and Joël Lambert.

REFERENCES

[1] 802.11p-2010 - IEEE Standard for Information technology-- Amendment 6: Wireless Access in Vehicular Environments

[2] A. Molisch, F. Tufvesson, J. Karedal, C. Mecklenbrauker, "A survey on vehicle-to-vehicle propagation channels", IEEE Wireless Communications, vol.16, no.6, pp.12-22, Dec. 2009.

[3] J.Ledy, H.Boeglen, AM.Poussard, B.Hilt, R.Vauzelle, "A semi-deterministic channel model for VANETs simulations ", International Journal On Vehicular Technology, Volume 2012, Article ID 492105

[4] http://www.medav.de/rusk_mimo.html?&L=2

[5] J. Mitola, "The Software Radio", in IEEE National Telesys Conference, May 1992.

[6] http://www.ettus.com/

[7] http://www.swig.org/

[8] P. Fuxjäger, A. Costantini, D. Valerio, P. Castiglione, G. Zacheo, T. Zemen, F. Ricciato, "IEEE 802.11p Transmission Using GNURadio", 6th Karlsruhe Workshop on Software Radios, March 2010.

[9] 802.11-2007 - IEEE Standard for Information technology -- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Laye (PHY) Specifications.

[10] http://sourceforge.net/apps/wordpress/itpp/

[11] A. L. Troya Chinchilla, "Synchronization and Channel Estimation in OFDM: Algorithms for Efficient Implementation of WLAN Systems", Brandenburgischen Technischen Universität Cottbus, PhD Thesis, 2004.

[12] M-J Canet et al. , "FPGA implementation of an OFDM-based WLAN receiver", Microprocessors and Microsystems, Elsevier, Volume 36, Issue 3, May 2012.

[13] http://oprofile.sourceforge.net/news/

[14] http://www.dspace.com

[15] D. Stencil et al., "Performance of 802.11p Waveforms over the Vehicle-to-Vehicle Channel at 5.9 GHz", IEEE 802.11p working group, Sept. 2007.