Towards an Open Source IEEE 802.11p Stack: A Full SDR-based Transceiver in GNU Radio

Bastian Bloessl*, Michele Segata*†, Christoph Sommer* and Falko Dressler*
*Computer and Communication Systems, Institute of Computer Science, University of Innsbruck, Austria
†Systems and Networks, Dept. of Information Engineering and Computer Science, University of Trento, Italy
{bloessl,segata,sommer,dressler}@ccs-labs.org

***Abstract*—We present the first steps towards an Open Source simulation and experimentation framework for IEEE 802.11p networks. The framework is implemented based on GNU Radio, a real-time signal processing framework for use in Software Defined Radio (SDR) systems. The core of the framework is a modular Orthogonal Frequency Division Multiplexing (OFDM) transceiver, which has been thoroughly evaluated: First, we show that its computational demands are so low that it can be run on low-end desktop PCs or laptops and thus, the transceiver is also feasible to use in field operational tests. Secondly, we present simulation results to highlight the transceiver's capability to study and debug PHY and MAC variants in a reproducible manner. We show that the simulations match very well to a widely accepted error model for IEEE 802.11p networks. Finally, we discuss results from an extensive set of measurements that compare our SDR-based transceiver with commercial grade IEEE 802.11p cards. We made the framework available as Open Source to make the system accessible for other researchers and to allow reproduction of the results. This might also pave the way for future proofing cars by means of fully reconfigurable radios.**

I. Introduction

Inter-Vehicle Communication (IVC) is the basis for Vehicular Ad Hoc Networks (VANETs), networks of cars communicating directly with each other and possibly with infrastructure nodes [1]. Once successfully deployed, IVC is the basis for a multitude of applications ranging from safety, e.g., intersection collision warning systems [2], over efficiency, e.g., traffic information systems [3], [4], to entertainment applications [5]. Besides the use of cellular networks such as UMTS or LTE for IVC, Dedicated Short Range Communications (DSRC) based on IEEE 802.11p are considered a key technology [6].

Two important steps towards the realization of these networks were made: First, in 1999, the regulatory bodies ECC and FCC reserved five and seven 10 MHz channels in the 5 GHz band exclusively for VANETs in Europe and the U.S., respectively. Secondly, the standardization of the IEEE 802.11p [7] PHY and MAC layer has been completed. Higher layer protocols have been standardized in the context of ETSI ITS G5, including Decentralized Congestion Control (DCC) [8], and the Wireless Access in Vehicular Environment (WAVE) communication stack, including multi channel operation [9].

IEEE 802.11p combines the quality of service extensions defined in IEEE 802.11e and the OFDM PHY of IEEE 802.11a. To account for the high mobility and the challenging environment in VANETs, all timings of the IEEE 802.11a were doubled, resulting in a channel bandwidth of 10 MHz. Sticking very close to the successful IEEE 802.11a standard has the obvious advantage that the same chips can be used for applications in VANETs and, thus, the costs can be, at least in theory, reduced considerably.

Now experimental validation is needed and field tests are currently ongoing in Europe and the U.S.

The current situation is that, on the one hand, DSRC radios for practical experiments are either commercial grade solutions developed for field tests, which are rather expensive and, most importantly, do not allow to change parts of the PHY or MAC of the underlying IEEE 802.11p implementation. Examples are the Cohda Wireless MK2 or the Denso WSU. On the other side of the spectrum, adapted stacks for the popular Atheros chipset AR5414A-B2B (e.g., used in the Unex DCMA-86P) have been developed in several projects. This allows to use very cheap hardware solutions, but they are bound to the capabilities of the chipset, and not fully validated in field tests. Finally, one conclusion of the last Dagstuhl Seminar on IVC was that the reproducibility of measurements with different hardware platforms is very limited. All present researchers agreed that a fully Open Source stack would be of huge benefit.

We close this gap by providing[1] an Open Source simulation and experimentation framework for IEEE 802.11p for SDR systems. We implemented the system for GNU Radio, a real-time signal processing framework for SDR. In our experiments, we relied on the widely used USRP N210 from Ettus Research. The core of our framework is a modular OFDM transceiver, which extends our previous work on an OFDM receiver [10], by now supporting all four modulation schemes BPSK, QPSK, QAM-16, and even QAM-64. The system is implemented fully in software and runs on a typical laptop computer.

Given that the design goals for IEEE 802.11a were set to support low mobility indoor environments, it is unclear if and to what extent IEEE 802.11p allows robust and reliable communication in vehicular networks. As the properties of the PHY and MAC implementation are easy to change in an SDR solution, our system provides all necessary features to experiment with protocol changes in a flexible lab environment. In particular, our implementation supports to connect the transmitter and receiver via a simulated channel within GNU Radio to study the system behavior before performing real measurements.

[1] http://www.ccs-labs.org/projects/wime/

In addition, our SDR based IEEE 802.11p transceiver can be regarded as a first step towards SDR based systems to be deployed in real cars. We believe this will be a necessary step as the product cycle of typical cars is very long and installed systems for IVC need to be able to be changed by means of software updates only. With an SDR solution, even PHY and MAC protocol updates become feasible.

Our main contributions can be summarized as follows:

- We present the first SDR based transceiver for IEEE 802.11p, which has been implemented fully in software in GNU Radio (Section III). Performance measurements confirm that our system runs even on simple laptop computers without problems.
- Our system can be used for simulation of new PHY or MAC versions by connecting transmitter and receiver within GNU Radio (Section IV). We show that the simulations match very well to a widely accepted error model for IEEE 802.11p networks.
- We validated the SDR transceiver by comparing it with a commercial grade Unex DCMA-86P2 radio (Section V). Our measurement results indicate that the SDR solution performs exactly as expected.

II. Related Work

Experimental research in VANETs can be done on many different layers depending on which aspect of the transceiver system is to be studied.

On the highest layers one can utilize hardware prototypes like the Cohda Wireless MK2 or the NEC Linkbird IEEE 802.11p system that was used in the CVIS project [11]. These solutions implement the complete communication stack and, thus, lend themselves well to study actual applications like safety and efficiency systems, or information dissemination strategies like adaptive beaconing. The MK2, for example, implements the whole IEEE 802.11p stack, including the IEEE 802.11e based Quality of Service (QoS) extensions and provides an SDK that allows implementation of WAVE based applications. There are also a wide variety of custom stacks, e.g., based on Atheros AR5414A-B2B chipsets that are, however, not validated. The drawback of these solutions is that the PHY and (at least parts of) the MAC layer are implemented in hardware and thus fixed, i.e., it is not possible to investigate alternative PHY and MAC algorithms.

On the MAC layer, the impact of different channel access schemes, i.e., the Distributed Coordination Function (DCF) can be studied. This topic received great attention in the WiFi research community and, since IEEE 802.11p is very similar to IEEE 802.11a, the concepts can also be used in VANETs. To experiment with MAC layer protocols, the community came up with several approaches. The spectrum ranges from systems that provide access to the timings, medium access, and acknowledgement functionality [12], [13], up to completely replacing the MAC with a fully programmable state machine [14].

The physical layer represents the bottom of the communication stack, and is typically the most difficult to study with commercial devices. Experimentation and testing of new ideas and solutions is not possible, since algorithms are implemented in hardware and thus fixed. SDR systems provide the possibility to investigate the physical layer, since they replace the transceiver chip with general purpose hardware and therefore can encode, modulate, and transmit arbitrary electromagnetic signals.

In terms of hardware, there are basically two architectures for SDR systems that differ in where the signal processing is implemented. First, everything can be implemented on a Field-Programmable Gate Array (FPGA), like the well-known WARP platform that was developed at Rice University [15] or the OpenAirInterface Express MIMO board of EURECOM [16]. Second, one can implement the signal processing on General Purpose Processors (GPPs) of a normal PC. This approach is used by Microsoft's Sora platform [17] as well as the Universal Software Radio Peripheral (USRP) of Ettus Research, which is typically used with the GNU Radio signal processing framework.

Since FPGAs are often the only chance to meet tough timing constraints of today's wireless standards they are often the only option for experimentation with MAC protocols. Yet, the strict deterministic timing of FPGAs comes at the price of increased complexity of implementation. However, there is also an approach between pure GPP and FPGA based realizations: implementing only the channel access functionality on the FPGA. This allows for standard compliant broadcast transmissions while doing all signal processing on the CPU, combining the advantages of both worlds.

In terms of software, there are several physical layer implementations for IEEE 802.11p. Mango Communications recently released an IEEE 802.11 reference design for their WARP boards. Microsoft's Sora also features an implementation of IEEE 802.11, which is, however, not Open Source. The receiver we presented in [10] provides another implementation that is implemented completely on the PC and is thus well suited for rapid prototyping.

III. Transceiver Structure

To enable experimental wireless research in VANETs, we implemented an SDR-based IEEE 802.11p transceiver. An SDR system consists of a software part, where the signal processing algorithms are implemented, and a hardware part that is responsible for up and down conversion of the analog wave form. On the software side, we implemented the transceiver based on GNU Radio, a GPP based real-time signal processing framework. On the hardware side, we used the Ettus Research USRP N210 with a CBX daughter board, allowing us to operate on the Intelligent Transportation Systems (ITS) frequency band around 5.9 GHz. The FPGA image of the N210 takes care of down sampling, filtering, and removing of the DC offset.

The transceiver software consists basically of send and receive chains and code that switches between the two states.

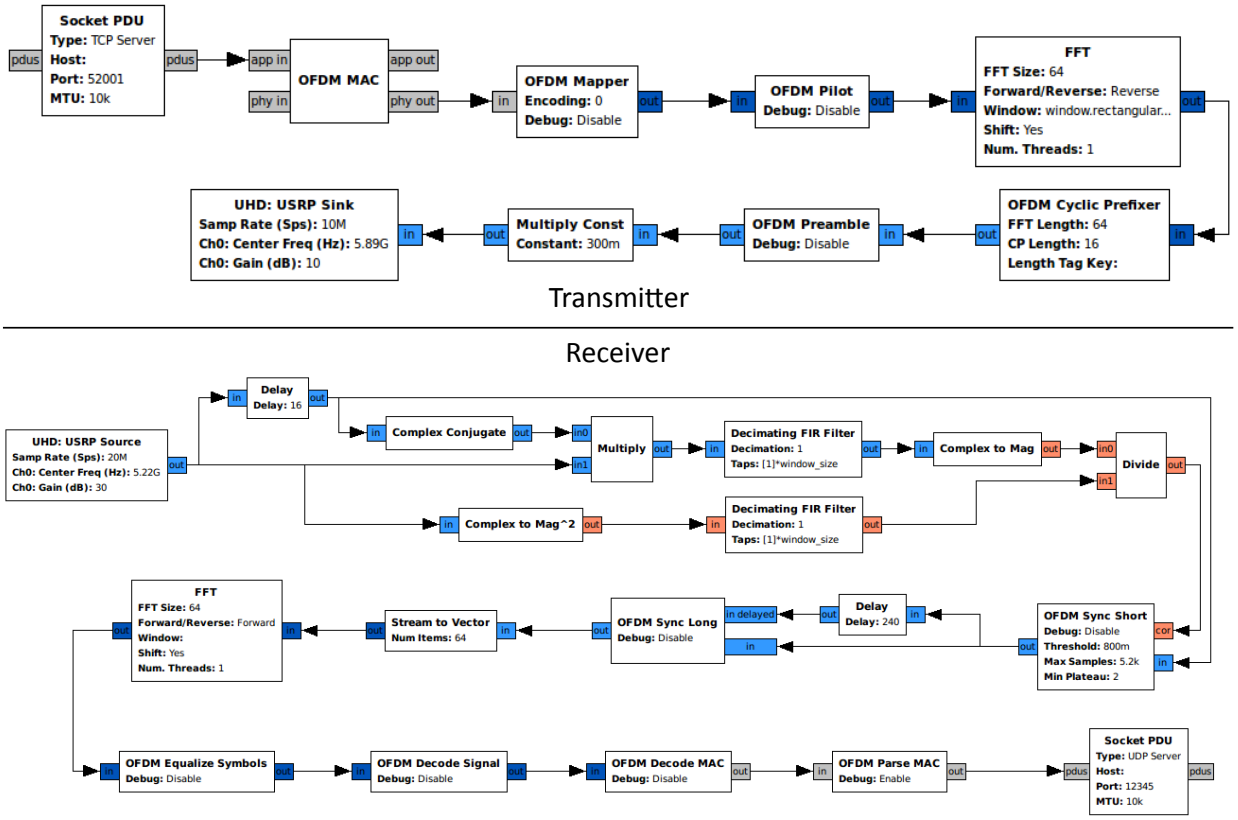

Figure 1. Overview of the transceiver structure in GNU Radio Companion.

A. Transmitter

An IEEE 802.11p transmitter for the N210 and GNU Radio has already been presented by Fuxjäger [18]. Since the transmitter was implemented partly in Python and based on an incompatible version of GNU Radio that lacked many of the recently introduced features that enable seamless packet-based operation, we reimplemented it from scratch. Regarding its use in a transceiver, our transmitter implementation also has the important advantage that it supports variable packet sizes and allows to specify the encoding on a per packet basis. In contrast to the receive chain, there are nearly no design decisions to make on transmitter side since modulation and encoding are fully specified in the standard [19, Chapter 18]. The only parameter that we chose is the transition width of the window function, which we set to 1. This window mainly asserts that the output signal honors the spectral mask defined in the standard and, thus, the signal decays fast in the frequency domain, limiting adjacent channel interference.

An overview of the transmitter structure as exposed to the GNU Radio Companion is depicted in the top half of Figure 1. The Companion provides a graphical user interface to setup and configure signal processing flow graphs. The implementation of the transmitter is straightforward and includes mainly encoding, Fast Fourier Transformation and addition of the cyclic prefix.

A major concern regarding the transmitter might be the carrier sensing mechanism, which is fundamental for both unicast and broadcast transmissions. IEEE 802.11p uses the extended DCF defined in the IEEE 802.11e amendment, which adds support for QoS. Due to latencies incurred by the communication between PC and the USRP it is not possible to implement the carrier sensing logic in software, since this introduces a large blind spot between the time the medium is sensed and when it is finally accessed. This issue can, however, be circumvented by implementing the CSMA/CA on the FPGA, which is possible since the SDR has an on-board flash where it can store a frame and handle the channel access in hardware. A basic CSMA implementation for the N210 with a non standard PHY that demonstrates this possibility has been presented in [20].

B. Receiver

On the receiver side we build on our implementation presented in [10]. In contrast to the transmitter, the performance of the receiver is crucial since it has to catch up with the incoming sample stream. The required computational speed is achieved with the help of Single Instruction Multiple Data (SIMD) instructions as provided by GNU Radio's Vectorized Library of Kernels (VOLK) [21]. Instead of iterative calculation, SIMD instructions act on vectors of data and thus, provide a significant speed up.

Our receiver has a modular and easy to use structure as depicted in the bottom half of Figure 1. At first, the

autocorrelation coefficient is calculated in order to detect the cyclic pattern of the short preamble of OFDM frames, which is used for frame detection by the *OFDM Sync Short* block. The following blocks are responsible for frame alignment, frequency offset correction, channel estimation and actually decoding the payload. On the receiver side the modularity is important since we expect a main application of the framework to be the investigation and comparison of different receive algorithms, for instance different channel estimation strategies. With a modular concept it is easy to exchange algorithms and study the impact on the performance. For the evaluations presented in this paper, we also implemented support for QAM-16 and QAM-64 encodings. This means that meanwhile all modulation and coding schemes defined in the standard are supported.

As with the transmitter, the receiver experiences communications latency between PC and USRP. Unfortunately, this makes it impossible to comply with the tough timing constraints of RTS/CTS and acknowledgement frames that are used for unicast transmissions. If unicast transmissions have to be investigated it is possible to work around this limitation by disabling retries due to missing acknowledgements and by setting the RTS/CTS threshold to infinity. However, such investigations would most likely be better served with full FPGA based solutions like WARP. Nevertheless, a vast array of VANET applications do not use acknowledged unicast transmission. Rather, the broadcast characteristic of the medium is exploited to effectively disseminate information. These broadcast transmissions are fully supported by our framework.

C. Merge Chains

To create a transceiver out of separate receive and transmit chains we need to address RX/TX switching. When the N210 is operating in half-duplex mode, the device is by default in receive mode and switches on demand when samples are streamed to the device. After a burst of samples is streamed to the device, the SDR does not switch back to receive mode immediately, but waits for a timeout since the sample stream might just be stalled. Due to this timeout, immediate responses like acknowledgements might be missed. We circumvented this issue by implementing a means for GNU Radio to signal the end of a frame, forcing the device back to receive mode.

To underline the functionality and the applicability of the transceiver as a WiFi system, we connected it to a TAP interface, a virtual Ethernet device. This way, the transceiver is seamlessly connected to the Linux TCP/IP stack (or a custom ITS G5 or WAVE stack) and can be used like a standard network interface card. Furthermore, we record sent and received traffic in PCAP format. PCAP is the de-facto standard for packet capturing and allows to investigate the traffic with network monitoring software like Wireshark.

D. Computational Performance

As a first performance metric of the transceiver we study its computational complexity and real-time capabilities. The point we want to make here is that an average PC can easily process the incoming sample stream and still has lots of spare resources.

Component	Type
CPU	Intel Core i7-2600 CPU 3.40GHz
RAM	16 GB
NIC	RTL-8169 Gigabit Ethernet
Operating System	Ubuntu 12.04 LTS, 64 bit
GNU Radio	Version 3.7
SDR	Ettus Research N210 revision 4
Daughterboard	CBX

Table I
OVERVIEW OF THE MOST IMPORTANT COMPONENTS OF OUR TEST SYSTEM.

This fact is important since it shows that the transceiver in its current state does not hit performance limits of typical computer systems and can thus, decode packets without systematic errors that occurred if the transceiver would not be able to cope with the incoming sample stream in real-time.

By default GNU Radio starts one thread for every block in the flow graph. Each of these blocks monitors performance metrics while the transceiver is running and exposes them so that they can be accessed live from other applications [22]. The monitored performance metrics include average utilization of input and output queues and consumed CPU time. We implemented a simple application that connects to the transceiver and logs all performance metrics in CSV format.

Note that the computational demands are only interesting for the receiving part of the transceiver. The transmitter is occasionally generating a sample stream that is sent to the SDR. Here, it is only important that the stream does not stall, which would corrupt the physical wave form. This is, however, no problem in practice.

To show the low computational demands, we connect a PC with a Unex DCMA-86P card via cable and attenuators to the USRP and send BPSK 1/2 encoded, 133 B packets with a rate of 30 packets per second. We picked this packet size since it corresponds to the size of an unsigned Cooperative Awareness Message (CAM) [8]. For the sake of brevity, in this paper we only show the results regarding this particular packet size. We also performed the experiments with different sizes to make sure that none of the statements in this work depends on such parameter. We configure the transmission power and attenuators so that the Signal to Noise Ratio (SNR) is around 40 dB so that packet loss is negligible. The main components of the SDR system as well as the PC are listed in Table I.

Figure 2 shows the average utilization of the input queues. Some of the blocks are listed twice in the graph since they have more that one input queue. At first, we see that none of the blocks backlogs an excessive amount of samples over time. Furthermore, we also investigated the variance of utilization: it is minimal, resulting in no overruns, i.e., drop of samples caused by too slow processing of the incoming samples.

Figure 3 provides a slightly different view on the system and shows the distribution of the aggregated work time over all blocks. While the first plot highlights bottlenecks where samples might queue up, the second blocks identifies heavy hitters in terms of required computation. In this plot, we clearly

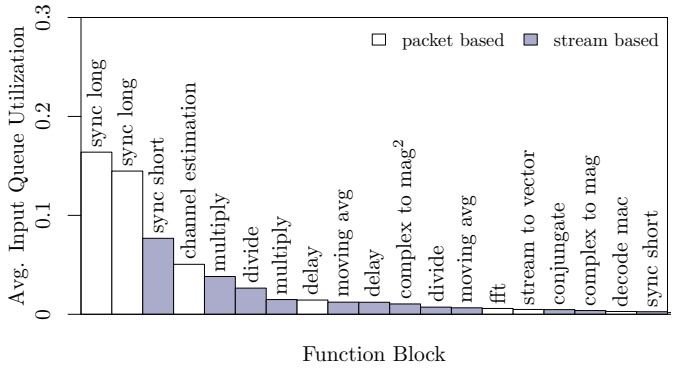

Figure 2. Average queue size of individual blocks of the receiver.

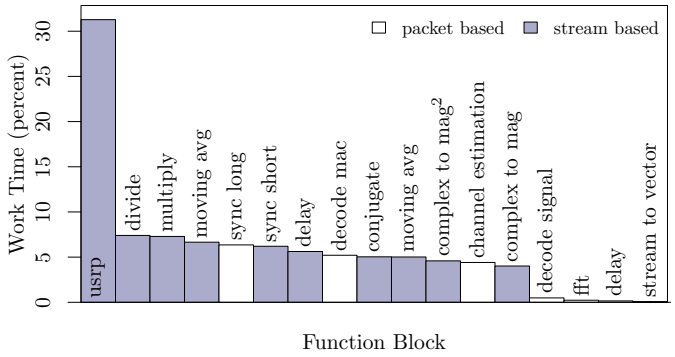

Figure 3. Total work time consumed by individual blocks of the receiver.

see that most of the time is spent in the USRP block, which interfaces the with the actual hardware.

To further reason about the performance and the limits of the system, we need to consider its structure. The first part of the receiver is responsible for frame detection and works on all incoming samples. Since this part operates on all samples it depends only on the sample rate and thus, on the bandwidth of the signal. These blocks are shaded in Figure 3 and scale linearly with the bandwidth of the signal, i.e., the computational demand will double when investigating 20 MHz signals as opposed to 10 MHz frames as in IEEE 802.11p.

Once a frame is detected, the second part of the transceiver is responsible for decoding the frame. The most computationally complex tasks in this part are frame alignment (we employ matched filtering), channel estimation, and Viterbi decoding. The complexity of the decoding process increases with packet size, and grows linearly with the number of frames per second.

The results show that there is a trade off between the number of packets per second and the bandwidth of the signal. Furthermore, in the given setup we see that there are resources left on a typical PC system, so that more advanced and complex receive algorithms can be implemented. To verify this, we also ran the transceiver on a Dell Latitude E6220 laptop with an Intel i5 processor and 8 GB RAM without any performance problems, thus, assuring that the system is also suitable for field tests.

IV. Simulation

Since we now have a complete transceiver system in GNU Radio, we can use it to study the performance of the employed algorithms by means of simulation. A simulative performance evaluation is desirable since it allows to reproduce the experiments and also to compare the systems with independent research results. To show the correctness of the implementation and that we achieve reasonable performance we conducted simulations over an Additive White Gaussian Noise (AWGN) channel. Again, we set the packet size to 133 B and send 10000 packets each, for different encodings and SNRs.

The simulative determined packet delivery ratios are shown in Figure 4. The error bars depict the confidence intervals with a confidence level of 0.95. We can see that the curves corresponding to the different encodings are reasonable in the sense that higher order modulations require a higher SNR.

More interesting is, however, the comparison with independent measurements and simulations. In [23], Mittag et al. implement a similar transceiver system in the discrete event simulator ns3. This implementation was used for detailed simulations of physical layer effects in WiFi networks. This transceiver was not real-time capable and not intended for use in an SDR environment. However, to show the correctness of the implementation and prove that their implementation produces meaningful results, the authors also present packet error curves, which match very well with the results presented in Figure 4.

In [24], Pei and Henderson derive a higher level packet error rate model for WiFi networks, which is well accepted an heavily used in ns3. The model is justified theoretically and compared to real measurements in the testbed of Carnegie Mellon University. Again, the model as well as the measurements match very well with our simulation results.

We think that simulations in an SDR environment that can be backed up with real over the air measurements have a high potential. There are two major use cases where they are particularly beneficial. First, since one important application for the transceiver is the investigation and comparison of different receive algorithms, the simulation can be used to validate implementation of these algorithms under controlled circumstances. Once the implementations of the algorithms are validated by simulations, the very same systems and implementations can be used for over the air measurements.

The second major use case for simulation is the investigation of wireless channels with different characteristics. There is a large body of channel measurements and characterizations in the VANET community. Most of the measurements are made with sophisticated channel sounders that allow to measure the characteristics in great detail. That way, features like delay spread and coherence time are measured and different channel models that capture the characteristics of different environments are proposed [25]. These channels can be easily implemented in GNU Radio and used to study the performance of different receive algorithms under reproducible conditions.

Finally, simulations can be used to derive error curves that might in turn serve as input for network simulators, albeit under the restrictions of the applicability of error curves discussed in [26].

V. Performance Comparison with Commercial IEEE 802.11p Devices

Even though we showed in the above mentioned simulations that the implemented algorithm works nicely in theory, we still owe a proof that the SDR system also works well with real hardware and produces results that are similar to commercial IEEE 802.11p devices. This is a crucial part for the evaluations since it shows that the implementation is indeed usable for research: producing reasonable results not only by means of simulations but also over the air with all the hardware impairments like frequency offsets, clock drifts and imperfect channel filters.

In [10], we already presented these measurement results for the receiver. To also provide these error curves for the transmitter, we connected the USRP via cable and attenuators to a Unex DCMA-86P2, a commercial IEEE 802.11p capable WiFi card. These cards are based on an Atheros transceiver chip and can be operated in IEEE 802.11p mode on the frequency band reserved for ITS applications (with minor modifications to the Linux kernel like removal of regulatory restrictions and the implementation of an interface to change the signal bandwidth to 10 MHz).

All measurements are performed on channel 172 with a center frequency of 5.86 GHz, again a packet size of 133 B, and a rate of 30 packets per second. Like in the case of the simulations, we send 10000 packets per configuration. At first, we investigate the transmit side and send frames with the SDR and receive them with the commercial WiFi card. The resulting error curves are depicted in Figure 5. The SNR is measured on the receiving side by the Unex devices. When put in monitor mode, the cards annotate each received frame with metadata including signal and noise levels.

To directly compare the SDR with a commercial device, we repeated the same measurements with another Unex device as sender. Since we experienced deviating results with different WiFi cards, we used the same receiver for both measurements. The error curves of these measurements are shown in Figure 6. We can see that the results for both devices match very closely except for the QAM-64 3/4 encoding where we experience worse performance with the SDR. Since we do not see such an effect in simulations and since the systems works well for the other cases, we are reasonably sure that the sample stream we generate is correct. Furthermore, we experienced no underruns, i.e., we were able to stream the samples to the device so that the device did not stall which would destroy the physical wave form. With these observations, we expect the deviation in the results to be caused by hardware imperfections. Candidates that might cause such a behavior are oscillator drift and, more likely, non linearities in the amplifier that might slightly disturb the signal, which might lead to packet errors especially in higher order modulations.

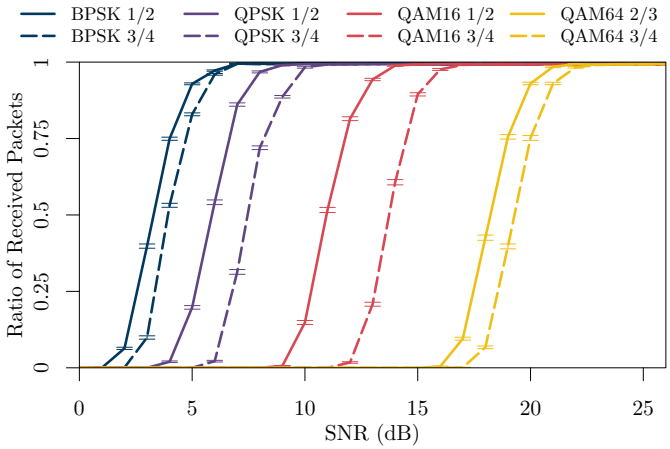

Figure 4. Simulative determined packet delivery ratio of 133 B sized packets over an AWGN channel.

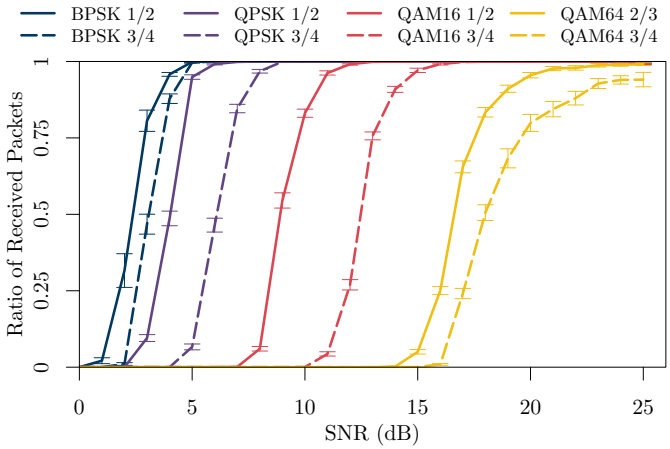

Figure 5. Packet delivery rate of frames sent from the SDR and received with a commercial device. The devices are connected via cable and the packet size is 133 B.

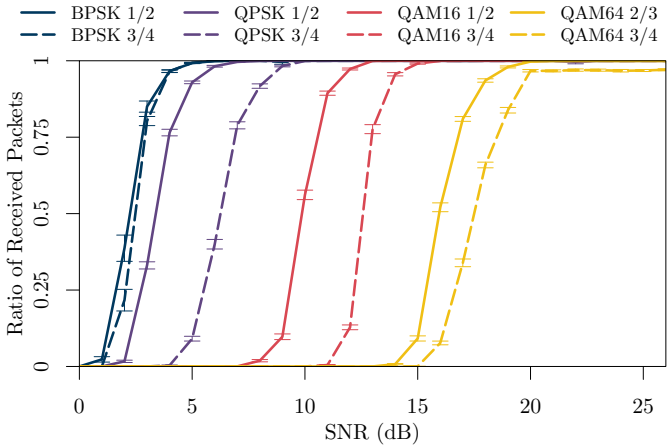

Figure 6. Packet delivery rate for two commercial grade IEEE 802.11p devices. The devices are connected via cable and the packet size is 133 B.

VI. Conclusion

We presented an IEEE 802.11p simulation and experimentation framework. We made all code available as Open Source software to allow reproduction of our results and to make the system accessible for use and for study by fellow researchers. We think that the system is accessible since it is based on GNU Radio, which has a big community and is furthermore known to work with affordable, well-known, and wide spread hardware from Ettus Research. The system can be used to investigate and compare different physical layer algorithms without limitations and the full IEEE 802.11p stack for broadcast transmissions. This system can also serve as a first step towards an Open Source WAVE or ITS G5 stack, which was identified by the community as a requirement for conducting reproducible field trials.

Another main benefit of our system is that it is implemented in an SDR environment and thus, the same code can be used to simulate as well as actually transmit and receive frames over the air. The core of the framework is an OFDM IEEE 802.11a/g/p transceiver that we evaluated with an extensive set of performance measurements. We studied its runtime performance, which is crucial for real time systems and we investigated packet error rate curves and validated it with commercial devices. The results show that the performance of the transceiver matches commercial cards.

References

[1] T. L. Willke, P. Tientrakool, and N. F. Maxemchuk, "A Survey of Inter-Vehicle Communication Protocols and Their Applications," *IEEE Communications Surveys and Tutorials*, vol. 11, no. 2, pp. 3–20, 2009.

[2] S. Joerer, M. Segata, B. Bloessl, R. Lo Cigno, C. Sommer, and F. Dressler, "To Crash or Not to Crash: Estimating its Likelihood and Potentials of Beacon-based IVC Systems," in *4th IEEE Vehicular Networking Conference (VNC 2012)*. Seoul, Korea: IEEE, November 2012, pp. 25–32.

[3] C. Lochert, B. Scheuermann, C. Wewetzer, A. Luebke, and M. Mauve, "Data Aggregation and Roadside Unit Placement for a VANET Traffic Information System," in *5th ACM International Workshop on Vehicular Inter-Networking (VANET 2008)*. San Francisco, CA: ACM, September 2008, pp. 58–65.

[4] C. Sommer, O. K. Tonguz, and F. Dressler, "Traffic Information Systems: Efficient Message Dissemination via Adaptive Beaconing," *IEEE Communications Magazine*, vol. 49, no. 5, pp. 173–179, May 2011.

[5] J.-S. Park, U. Lee, S. Y. Oh, M. Gerla, and D. S. Lun, "Emergency related video streaming in VANET using network coding," in *3rd ACM International Workshop on Vehicular Ad Hoc Networks (VANET 2006)*. Los Angeles, CA: ACM, September 2006, pp. 102–103.

[6] F. Dressler, F. Kargl, J. Ott, O. K. Tonguz, and L. Wischhof, "Research Challenges in Inter-Vehicular Communication - Lessons of the 2010 Dagstuhl Seminar," *IEEE Communications Magazine*, vol. 49, no. 5, pp. 158–164, May 2011.

[7] "Wireless Access in Vehicular Environments," IEEE, Std 802.11p-2010, July 2010.

[8] European Telecommunications Standards Institute, "Intelligent Transport Systems (ITS); Decentralized Congestion Control Mechanisms for Intelligent Transport Systems operating in the 5 GHz range; Access layer part," ETSI, TS 102 687 V1.1.1, July 2011.

[9] "IEEE Trial-Use Standard for Wireless Access in Vehicular Environments (WAVE) - Multi-channel Operation," IEEE, Std 1609.4, November 2006.

[10] B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "An IEEE 802.11a/g/p OFDM Receiver for GNU Radio," in *ACM SIGCOMM 2013, 2nd ACM SIGCOMM Workshop of Software Radio Implementation Forum (SRIF 2013)*. Hong Kong, China: ACM, August 2013, pp. 9–16.

[11] A. Festag, R. Baldessari, W. Zhang, and L. Le, "CAR-2-X Communication SDK - A Software Toolkit for Rapid Application Development and Experimentations," in *IEEE Vehicular Networking and Applications Workshop (ICC Workshops 2009)*. Dresden, Germany: IEEE, June 2009, pp. 1–5.

[12] M. Neufeld, J. Fifield, C. Doerr, A. Sheth, and D. Grunwald, "SoftMAC - Flexible Wireless Research Platform," in *4th Workshop on Hot Topics in Networks (HOTNETS 2005)*. College Park, ML: ACM, November 2005.

[13] M.-H. Lu, P. Steenkiste, and T. Chen, "Using Commodity Hardware Platform to Develop and Evaluate CSMA Protocols," in *3rd ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH 2008)*. San Francisco, CA: ACM, September 2008, pp. 73–80.

[14] G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, F. Gringoli, and I. Tinnirello, "MAClets: Active MAC Protocols over Hard-Coded Devices," in *8th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT 2012)*. Nice, France: ACM, December 2012, pp. 229–240.

[15] A. Khattab, J. Camp, C. Hunter, P. Murphy, A. Sabharwal, and E. W. Knightly, "WARP: A Flexible Platform for Clean-Slate Wireless Medium Access Protocol Design," *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R)*, vol. 12, no. 1, pp. 56–58, January 2008.

[16] P. Agostini, R. Knopp, J. Härri, and N. Haziza, "Implementation and Test of a DSRC Prototype on OpenAirInterface SDR Platform," in *IEEE ICC 2013, Workshop on Emerging Vehicular Networks*. Budapest, HUNGARY: IEEE, June 2013.

[17] K. Tan, J. Zhang, J. Fang, H. Liu, Y. Ye, S. Wang, Y. Zhang, H. Wu, W. Wang, and G. M. Voelker, "Sora: High Performance Software Radio Using General Purpose Multi-core Processors," *Communications of the ACM*, vol. 54, no. 1, pp. 99–107, January 2011.

[18] P. Fuxjäger, A. Costantini, D. Valerio, P. Castiglione, G. Zacheo, T. Zemen, and F. Ricciato, "IEEE 802.11p Transmission Using GNURadio," in *6th Karlsruhe Workshop on Software Radios (WSR)*, Karlsruhe, Germany, March 2010, pp. 1–4.

[19] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE, Std 802.11-2012, 2012.

[20] A. Puschmann, P. Di Francesco, M. A. Kalil, L. A. DaSilva, and A. Mitschele-Thiel, "Enhancing the Performance of Random Access MAC Protocols for Low-cost SDRs," in *8th International Workshop on Wireless Network Testbeds Experimental Evaluation and Characterization (WiNTECH 2013)*. Miami, FL: ACM, September 2013.

[21] T. Rondeau, N. McCarthy, and T. O'Shea, "SIMD Programming in GNU Radio: Maintainable und User-Friendly Algorithm Optimization with VOLK," in *Conference on Communications Technologies and Software Defined Radio (SDR'12)*. Brussels, Belgium: Wireless Innovation Forum Europe, June 2012.

[22] T. W. Rondeau, T. O'Shea, and N. Goergen, "Inspecting GNU Radio Applications with ControlPort and Performance Counters," in *ACM SIGCOMM 2013, 2nd ACM SIGCOMM Workshop of Software Radio Implementation Forum (SRIF 2013)*. Hong Kong, China: ACM, August 2013, pp. 65–70.

[23] J. Mittag, S. Papanastasiou, H. Hartenstein, and E. G. Ström, "Enabling Accurate Cross-Layer PHY/MAC/NET Simulation Studies of Vehicular Communication Networks," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1311–1326, July 2011.

[24] G. Pei and T. R. Henderson, "Validation of OFDM Error Rate Model in ns-3," Boeing Research & Technology, Tech. Rep., 2010.

[25] G. Acosta-Marum and M. Ingram, "Six Time- and Frequency-Selective Empirical Channel Models for Vehicular Wireless LANs," *IEEE Vehicular Technology Magazine*, vol. 2, no. 4, pp. 4–11, December 2007.

[26] M. Segata and R. Lo Cigno, "Simulation of 802.11 PHY/MAC: The quest for accuracy and efficiency," in *9th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2012)*. Courmayeur, Italy: IEEE, January 2012.