Performance Assessment of IEEE 802.11p with an Open Source SDR-based Prototype

Bastian Bloessl *Student Member, IEEE*, Michele Segata *Member, IEEE*,
Christoph Sommer *Member, IEEE*, Falko Dressler *Fellow, IEEE*

Abstract—We present a complete simulation and experimentation framework for IEEE 802.11p. The core of the framework is an SDR-based OFDM transceiver that we validated extensively by means of simulations, interoperability tests, and, ultimately, by conducting a field test. Being SDR-based, the transceiver offers important benefits: It provides access to all data down to and including the physical layer, allowing for a better understanding of the system. Based on open and programmable hardware and software, the transceiver is completely transparent and all implementation details can be studied and, if needed, modified. Finally, it enables a seamless switch between simulations and experiments and, thus, helps to bridge the gap between theory and practice. Comparing the transceiver's performance with independent results from simulations and experiments, we underline its potential to be used as a tool for further studies of IEEE 802.11p networks both in field operational tests as well as for simulation-based development of novel physical layer solutions. To make the framework accessible to fellow researchers and to allow reproduction of the results, we released it under an Open Source license.

Index Terms—Vehicular Ad Hoc Network, Software Defined Radio, GNU Radio, Wireless LAN

1 Introduction

VANETs describe the vision of cars communicating directly with each other and potentially with infrastructure nodes, creating a network of connected cars. Communication allows vehicles to cooperate and form what is commonly referred to as an Intelligent Transportation System (ITS). Once established, these networks can be the enabler for many applications that promise to increase road traffic safety, efficiency, and comfort, while decreasing fuel consumption and, thus, CO_2 emissions [5]. Today, VANETs are close to become reality, as engineers in Europe, as well as in the US and Japan, are finalizing relevant standards. Well-known examples are ETSI ITS-G5, IEEE DSRC/WAVE, and ARIB T109 which are defined for Europe, the US, and Japan, respectively. To foster the introduction of VANETs, the regulatory bodies of Europe and the US already reserved frequencies in the 5.9 GHz band for exclusive use.

While Japan took a different approach, ETSI ITS-G5 and IEEE DSRC/WAVE build on the same access technology. Both rely on IEEE 802.11p [6], a physical and MAC layer for use in vehicular environments. Based on OFDM, the physical layer of IEEE 802.11p is similar to IEEE 802.11a/g, except that all timings are doubled. This reduces the bandwidth from 20 MHz to 10 MHz and doubles the guard interval, which makes the signal more robust against delay spread. On MAC layer, IEEE 802.11p introduces the Outside the Context of a BSS (OCB) mode (formerly named WAVE mode to reflect its origins in the IEEE DSRC/WAVE standard), which allows cars to communication directly without prior connection setup. Finally, IEEE 802.11p mandates the use of the Quality of Service (QoS) extensions, first defined in IEEE 802.11e.

The decision to rely on Wireless LAN (WLAN) as a base and, thus, to use readily available technology can ease market introduction, but can also lead to doubts whether a physical layer that was designed for relatively static indoor environments is able to cope with the dynamics of VANETs. The relevance of the question regarding the performance of IEEE 802.11p in fast-fading channels is reflected by a large body of literature [7]–[10]. To study the physical layer, researchers often rely on simulators that model IEEE 802.11p at signal level, considering channel effects on the electromagnetic waveform [8], [11]–[13], or on hardware prototypes for experiments in the lab and on the road [14]–[16]. Both approaches are, however, limited to their domain only, i.e., it is not possible to switch from simulations to experiments. As a consequence, there is no easy way to validate and test findings from simulation studies through experiments or vice versa.

To overcome this limitation, we propose an SDR-based approach. SDRs are programmable radios that provide access to all data down to the physical waveform, allowing to implement all signal processing in software. This aspect makes them particularly well suited to build early prototype transceivers and to experiment with novel signal processing algorithms. The general concept of SDR is also interesting for car manufacturers, as the relatively long life-cycle of a vehicle asks for hardware that can be adapted to future changes to the standards. To show the feasibility of the idea and to foster the use of SDRs for VANET research, we developed a complete IEEE 802.11p transceiver and released it under an Open Source license. In this paper, we detail its implementation, show how we validated it

- *B. Bloessl, C. Sommer, and F. Dressler are with the Heinz Nixdorf Institute and the Dept. of Computer Science, Paderborn University, Germany, E-mail: {bloessl,sommer,dressler}@ccs-labs.org*
- *M. Segata is with the University of Trento, Italy, E-Mail: msegata@disi.unitn.it*

The manuscript is based on earlier work on the implementation and evaluation of an SDR-based IEEE 802.11a/g/p transceiver that was presented in [1]–[4].

through simulations and interoperability test with IEEE 802.11p prototypes, and finally show its applicability for experiments on the road by conducting a field test.

With our SDR transceiver, signal processing is implemented in software on a normal PC, making it particularly easy to use, modify, and debug. While we also investigate standard compliant channel access for broadcast transmissions, the main application domain of the transceiver is clearly the physical layer. In contrast to existing approaches, the major advantage is that the software implementation is decoupled from the radio frontend. This and the fact that the transceiver works with cheap, open, and widely used radio hardware makes the system accessible to fellow researchers. Since the physical layer of our SDR transceiver is implemented completely in software, new physical layer concepts can easily be implemented and tested. Furthermore, this also allows to use our transceiver for simulations too, enabling studies of IEEE 802.11p and novel variants in reproducible configurations. The ability to use the same tool for both simulations and experiments allows for a seamless switch between theory and practice and presents a big advantage of the proposed approach.

2 Related Work

Given their decentralized and dynamic nature, VANETs are particularly challenging to design. Even when limiting the focus to networking aspects only, there are many open questions ranging from application layer performance, down to propagation characteristics of the physical signal. Besides analytical models, which are beyond the scope of this work, researchers rely on simulations and increasingly on experiments to study those networks [17].

Simulations are often the first choice as they are easy to conduct and allow investigating VANETs in a reproducible manner. As a first type of simulator, dedicated physical layer simulators are tuned towards studying channel effects, interference, and signal processing algorithms. Here, most researchers rely on custom, unpublished implementations using a scripting language like MATLAB [8], [12], [13]. While the level of detail is the same as with an SDR, these simulators do not allow for real-time operation. Furthermore, simplifying assumptions have to be made in these models that may hide certain aspects in practice. The second type focuses on macro-scale network simulations. To capture unique characteristics of vehicular networks and to produce realistic results, researchers couple traffic simulators with network simulators bidirectionally. Well-known examples for such simulators are Veins, iTetris, and VSimRTI [18]. Focusing on larger scenarios, they lend themselves for investigating the MAC layer and the layers above. For performance reasons, these simulators usually employ simple channel and error rate models. There is, however, also the idea to model the physical layer on signal level [11] and, thus, to combine physical layer and network layer simulators. In fact, our initial SDR implementation has already been integrated into Veins by an independent group [19]. Finally, trace-based input modeling can be used. Such an approach to realistic VANET simulations was presented in [9], where the authors recorded raw signal samples in a field tests, which were later used for offline, trace-driven simulations. The downside of this method is, however, that it produces large amounts of data. Following Nyquist's sampling theorem, a 10 MHz channel leads to, at least, 10×10^6 complex base band samples per second. Furthermore, as the receiver does not decode the data live, it cannot be part of the VANET, but is limited to passive measurements only. Our approach has the prime advantage that the same code can be used for simulations and experiments, thus, offering a seamless switch between theory and practice.

Apart from simulations, experiments can provide valuable insights. Field tests are particularly important as they show the performance of a real system and reveal potential weaknesses in system design. Today, many experiments are conducted with dedicated IEEE 802.11p prototypes from companies like Cohda Wireless [20], NEC [21], [22], and Denso [23]. These devices provide complete communication stacks for IEEE DSRC/WAVE or ETSI ITS-G5 and are, therefore, well suited for testing VANET applications. Apart from dedicated prototypes, it is possible to modify certain off-the-shelf WLAN cards to operate in IEEE 802.11p mode. UNEX DCMA-86P2 cards, for example, are based on an Atheros chipset and were used in many experiments [14]–[16], [24]. Recently, the Linux kernel added IEEE 802.11p support to the Atheros *ath9k* driver. Together with Open Source implementations of VANET communication stacks, like *OpenC2X* [25], these WLAN cards can be used to build cheap and nearly feature-complete prototypes.

While both custom and commercial prototypes are well suited to test VANET applications, they share a common limitation in that they provide very limited access to the physical layer. This limitation can be overcome using SDR. Implementing the whole signal processing in software, SDRs allow us to study, and if needed modify, all implementation details. Furthermore, they operate on the physical signal and, thus, provide access to all data, allowing for a better understanding of the system.

There are several SDR platforms available, which can be differentiated based on how the physical layer is implemented. The two most popular variants are Field-Programmable Gate Array (FPGA)-based and General Purpose Processor (GPP)-based SDRs. The WARP board is a well-known example of the first type [26]. Mango Communications, the vendors of WARP, provide a free-to-use IEEE 802.11 reference design that seems well suited for VANET research. Another FPGA prototype was recently introduced in [27], where the authors demonstrate IEEE 802.11p transmission using National Instruments USRP-RIOs and LabView. While LabView offers a workflow that allows configuration and modification of the FPGA design using a graphical interface, it also comes with large licensing fees.

The main advantages of FPGA-based SDRs are their deterministic timing and low latency, which allows them to meet the tough timing constraints of today's communication standards. Therefore, if time-critical MAC layer algorithms or higher layer protocols are subject of the study, FPGAs are often the first choice. The drawback of these architectures is their price and their limited flexibility, especially with regard to the physical layer. While, they are fully reprogrammable in theory, it is, in practice, often challenging to implement complex signal processing algorithms on FPGAs.

Rapid-prototyping and physical layer experimentation

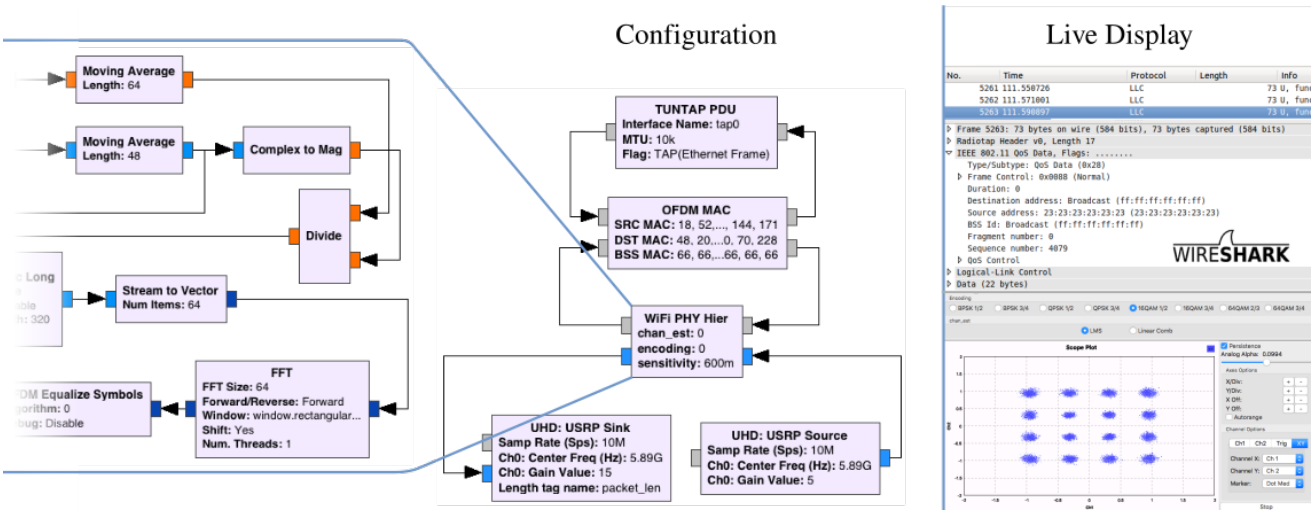


Figure 1. Overview of the transceiver's structure (left and center) and its graphical user interface while receiving 16-QAM frames (right).

are application areas where GPP-based SDRs are particularly well suited. Using GPP-based SDRs, the physical layer is implemented on a normal PC, often in a high-level programming language like C++ or Python. Compared to hardware description languages like VHDL or Verilog, which are often used with FPGAs, these high-level languages are more accessible as they are easier to write, compile, and test [28]. An additional benefit of PC implementations is that they are not limited to experiments only. By connecting sender and receiver in software, they can be used for simulations, allowing for an easy switch between theory and practice.

Popular GPP-based platforms are Microsoft's Sora and GNU Radio together with Universal Software Radio Peripherals (USRPs) from Ettus Research. For GNU Radio, there was no IEEE 802.11a/g/p transceiver available prior to this work (only an IEEE 802.11b transceiver [29], which is based on the very different Direct-Sequence Spread Spectrum (DSSS) physical layer). Microsoft, in turn, presented Sora together with *SoftWiFi*, an IEEE 802.11a/b/g implementation [30]. While both our transceiver and Sora rely on a GPP-based architecture, they focus on very different aspects.

Sora uses sophisticated hardware-software co-design that demonstrates what a highly optimized GPP architecture is able to achieve. In particular, Sora manages to reduce latency enough to extend standard compliance from the physical layer up to the MAC, allowing for unicast communication with commercial WLAN cards. These impressive results are possible by using a custom hardware control board that connects to the PC via PCIe, a high throughput and low-latency bus. On software side, Sora uses operating system specific optimizations to boost performance. Among others, Sora makes heavy use of look-up tables and employs a static scheduling scheme that distributes signal processing tasks manually between cores. To optimize data exchange between CPU cores, Sora uses special FIFO buffers that minimize synchronization overhead.

While Sora shows the potential of hardware-software co-design, it was a deliberate decision to use a different approach for our transceiver. We go without hardware and operating system specific optimizations and, therefore, trade off latency for a more accessible implementation that is not strictly tight to a particular platform. While we also explore the possibility to extend our implementation with standard compliant channel access for broadcast transmissions (i.e., the case where we do not have to meet the timing constraints of acknowledgment frames), our focus is clearly to provide a modular and easy to adapt physical layer implementation that lends itself well to rapid prototyping. Using GNU Radio, we can benefit from an active community and a large ecosystem, which provides us with a graphical editor to configure our transceiver, graphical outputs that visualize the signal live, and a wide variety of supported SDR radio frontends. Furthermore, our transceiver can be used on macOS and Linux and runs on many hardware platforms, including even ARM-based embedded devices. In contrast to Sora, it is, therefore, not limited to desktop PCs, making it a particularly interesting option for field tests.

To summarize, its biggest strengths are clearly the use of accessible software and hardware as well as the ability to run in simulation mode to experiment with new physical layer concepts in well-defined and repeatable scenarios. In the academic context, the most notable SDRs platforms that have been successfully tested with our transceiver are USRPs from Ettus Research, the HackRF, and the BladeRF. Overall, our transceiver is not tightly bound to a particular platform and focuses on providing a modular and easy to adapt implementation that is compatible with many radio frontends.

3 Concept and Implementation

Given the advantages of GPP-based architectures, we developed an IEEE 802.11p transceiver using GNU Radio [31], an Open Source real-time signal processing framework. For our implementation, we did not invent new signal processing algorithms, but used readily available and widely used algorithms to create a modular, easy to use, and easy to adapt transceiver that forms a solid base for research on VANETs. Even preliminary versions of our transceiver proved to be useful in many different contexts. For example, in studies on channel estimation algorithms for VANETs [32], [33], studies on location privacy in VANETs [34], and studies on the computational complexity of the transceiver [35].

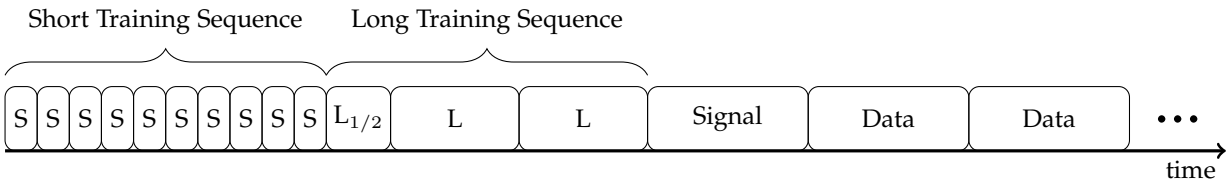


Figure 2. Overview of the WLAN frame structure. Each frame starts with a short and a long training sequence for synchronization, followed by the signal field, containing information about the length and encoding of the frame, and the data symbols, carrying the actual payload.

3.1 Overview

Similar to LabView and Simulink, GNU Radio is, at its core, a signal processing system for data streams. This architecture is very natural for SDRs, where the hardware produces a constant stream of complex baseband samples. To process the data, the sample stream is piped through signal processing blocks that implement actual functionality. With GNU Radio, such signal processing system is described by a *flow graph*, a data structure that defines how blocks are parameterized and connected.

A small part of our transceiver flow graph is depicted on the left hand side of Figure 1. The screenshot shows the transceiver in *GNU Radio Companion*, a graphical user interface that eases the creation and configuration of flow graphs. While the detailed functionality is not important at this point, the figure gives an idea of GNU Radio's stream paradigm; the signal processing blocks are visualized by the boxes, while the sample streams are depicted by the arrows that connect them. As flow graphs can get rather complex, GNU Radio allows encapsulating functionality in *hierarchical blocks* that can be used as building blocks in other flow graphs. An example is shown in the center of Figure 1, where we encapsulated the physical layer in a hierarchical block, allowing us to create a clearly laid out transceiver flow graph.

A great advantage of GNU Radio is that apart from real-time signal processing of samples from an SDR, it can be used for simulations, too. This is easily possible by looping back the generated sample stream into the receiver without interfacing actual radio hardware. GNU Radio aids simulation setup by providing models for hardware impairments like phase noise and clock drift, as well as propagation environments like Additive White Gaussian Noise (AWGN), Rayleigh, and multi-path fading.

With increased bandwidth demands of state of the art technologies, SDRs have to process a large number of samples per second. To cope with the data in real-time, GNU Radio exploits multi-core CPUs by starting each signal processing block in its own thread to distribute the load between the cores. To further speed-up computations, the individual blocks exploit vectorised instructions that today's CPUs provide through SIMD extensions like MMX, SSE, and AVX. With GNU Radio, these instructions are accessed through the Vector-Optimized Library of Kernels (VOLK) [36], which provides optimized implementations of common signal processing functions.

Improved computational performance through threaded operation and optimized instructions are essential to the implementation of our IEEE 802.11p transceiver. The critical component with regard to computational performance is the receiver, as it has to process a large number of samples per second. Before we started this project, there was no OFDM receiver available for GNU Radio supporting the required bandwidth [1]. It was, therefore, unclear whether it is possible to realize a receiver that can run in real-time on a normal laptop or desktop PCs.

3.2 Transmitter

Compared to the receiver [1], the transmitter [2] is rather straightforward to implement as the signal is fully specified in the standard and has to be generated accordingly. This is in contrast to the receiver, where the implementation is a design decision and generally a trade-off between performance and complexity. Furthermore, the transmitter does not pose high computational demands, as the whole frame can be pre-computed before it is streamed to the radio. The sole requirement is that the stream does not stall during transmissions, which is, however, no problem in practice.

We implemented the transmitter in C++, translating the frame format specification of the IEEE 802.11 standard to the stream paradigm of GNU Radio. Our transceiver is functional complete in the sense that it supports all packet sizes and modulation and coding schemes. The input to the transmitter is the payload of the frame, which is subsequently

- prefixed it with a MAC header that, for example, contains source and destination MAC addresses and the frame type;
- appended with a 32 Bit Cyclic Redundancy Check (CRC) for error detection;
- encoded with a convolutional code and punctured according to the coding rate;
- mapped to complex constellation points using BPSK, QPSK, 16-QAM, or 64-QAM;
- interleaved with pilot symbols;
- transformed to time domain with a 64-bin Fast Fourier Transformation (FFT);
- prepended with a cyclic prefix to cope with inter-symbol interference; and
- filtered to improve the spectral shape.

The output of this process are OFDM data symbols (cf. Figure 2). Finally, the whole frame is prefixed with a preamble sequence and a BPSK ½ encoded *signal field* that informs the receiver about the length and encoding of the following data symbols.

It should be mentioned that an IEEE 802.11p transmitter has already been presented by Fuxjäger et al. [37]. This implementation, however, was based on an older version of GNU Radio that lacked important features that allow for seamless packet-based operation. Moreover, it used a hardcoded fixed packet size and did not support specifying

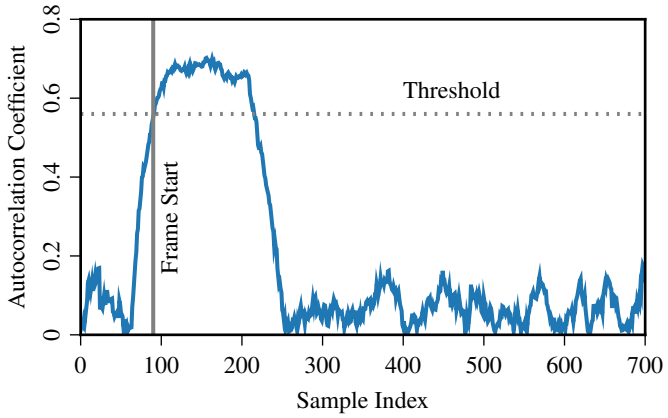


Figure 3. Characteristic behavior of the autocorrelation function as calculated in the frame detection part of the receiver.

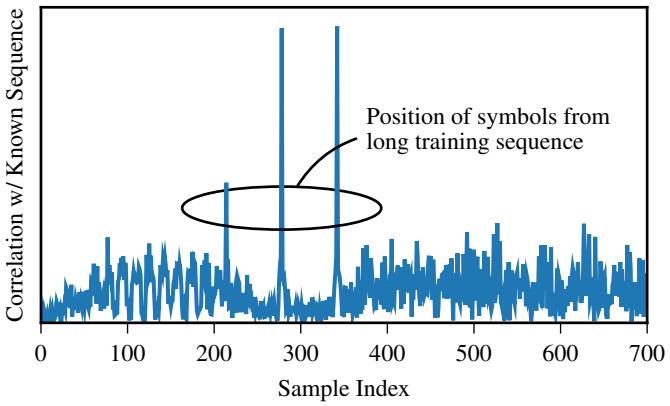


Figure 4. Characteristic output of the matched filter that correlates the sample stream with the known pattern of the long training sequence.

the modulation and coding scheme on a per packet basis. Given these limitations, we decided to reimplement the transmit side from scratch.

3.3 Receiver

The realization of a modular IEEE 802.11a/g/p receiver is an important part of our work and will be covered in greater detail. Its design is mainly motivated by the WLAN frame structure, depicted in Figure 2. Each frame starts with a short and a long training sequence that, despite their names, have the same length and span over two OFDM symbols. The qualifier rather refers to the structure of the sequence: The short training sequence uses a short pattern that repeats ten times during two OFDM symbols, while the long training sequence uses a longer pattern that repeats 2.5 times.

3.3.1 Frame Detection

To detect a frame, the receiver exploits the periodicity of the short training sequence by calculating the autocorrelation coefficient of the sample stream. Once this coefficient exceeds a configurable threshold, subsequent signal processing steps are triggered. Denoting the sample stream, i.e., the sequence of complex baseband samples received from the SDR, as s and its complex conjugate as $\bar{\mathrm{s}}$, the autocorrelation a is calculated as

$$\mathrm{a}[n] = \sum_{k=0}^{N_{\text{win}}+15} \mathrm{s}[n+k]\,\bar{\mathrm{s}}[n+k+16]\ . \tag{1}$$

The gap of 16 samples corresponds to the length of the short training sequence. To be independent from the input power level, we normalize a by the signal power p to calculate the autocorrelation coefficient c as

$$\mathrm{p}[n] = \sum_{k=0}^{N_{\text{win}}-1} \mathrm{s}[n+k]\,\bar{\mathrm{s}}[n+k]\ ; \tag{2}$$

$$\mathrm{c}[n] = \frac{\left|\mathrm{a}[n]\right|}{\mathrm{p}[n]}\ . \tag{3}$$

The typical course of the autocorrelation coefficient at the start of a frame is depicted in Figure 3. The plot shows the characteristic plateau during the short training sequence at a Signal to Noise Ratio (SNR) of 10 dB.

With two complex multiplications per sample (one to compute the autocorrelation and one to compute the power), this is a very efficient algorithm [38]. Another option would be to cross-correlate s with the known pattern of the short training sequence, which would, however, require 16 times the multiplications, as this method scales with the length of the sequence. Since the frame detector has to process all samples, its computational efficiency is important, which led us to use the autocorrelation-based approach. To find a good parameter set for the employed algorithms, we conducted simulations over an AWGN channel and determined parameters empirically. The results of these simple simulations, which we left out for the sake brevity, showed that a window size N_{win} of 48 and an autocorrelation coefficient threshold of 0.56 yield good results.

3.3.2 Symbol Alignment

Symbol alignment is done in a separate step, since the plateau of the autocorrelation cannot accurately determine the start of the frame. As this part of the receiver operates only on a subset of the samples, i.e., it is triggered when a frame was detected, we are able to use more complex algorithms at this stage. Here, we employ a matched filter to cross-correlate the sample stream with the known pattern of the long training sequence. The typical course of the cross-correlation function is depicted in Figure 4, which, again, is plotted at an SNR of 10 dB. Note that the plot uses a linear y-axis scale, but we omitted the axis labels as the signal is not normalized, i.e., only its relative course is of interest at this stage.

As shown in Figure 2, the long training sequence consists of a pattern that repeats 2.5 times. The spikes in Figure 4 indicate the position where this pattern matches with the input signal. Since the peaks are very narrow, this method allows us to determine the OFDM symbol boundaries accurately. In our implementation, we filter the number of samples corresponding to the length of the training sequence and search for peaks with the appropriate spacing.

3.3.3 Frequency Offset Correction

With moving devices and imperfect oscillators, the received spectrum deviates from the ideal carrier frequency, introducing a frequency offset that the receiver has to account for. Like in [39], we exploit the repetitive nature of the

short training sequence to estimate the frequency offset by calculating

$$\Delta f = \frac{1}{16} \arg \left(\sum_{n=0}^{N_{\text{short}}-1-16} s[n]\, \overline{s}[n+16] \right) . \quad (4)$$

We average over N_{short} samples, the complete length of the short training sequence to get a good estimate of the offset. This is possible, as the frame's alignment and, thus, its start is known very accurately in this stage. The rationale behind the formula becomes clear when looking at the perfect case. Given the periodicity of the short training sequence, a sample corresponds to the time shifted sample ($s[n] = s[n+16]$). Therefore, multiplication with its complex conjugate yields a real number, and Δf is zero. A frequency shift, in turn, introduces a slight phase rotation between $s[n]$ and $s[n+1]$, which will be reflected in Δf, the average phase rotation between successive symbols, which we use to correct the signal by calculating

$$s[n] \leftarrow s[n]\, e^{\,\mathrm{i}\,(n\,\Delta f)} . \quad (5)$$

3.3.4 Channel Estimation

Using a 64-bin IFFT to switch to frequency domain, we employ the Least Squares (LS) algorithm to compensate for frequency selective fading, introduced by multi-path propagation. The LS equalizer, which is often used as a baseline [10], [40], is a simple algorithm with low computational complexity that uses the long training sequence as block pilots to compute an estimate of the channel and uses it to correct the rest of the frame. Since we expect the study of channel estimation algorithms to be one interesting application of our transceiver, we implemented an interface to plug in different algorithms.

3.3.5 Decoding

With the corrected symbols, the final step is to decode the information using the reverse process implemented in the transmitter. As the signal field contains information regarding the modulation and encoding of the payload, it is decoded first and its data is used to configure the decoder. Finally, the received frames can be exported in the PCAP format, which allows analyzing them with network monitoring software like *Wireshark*. A screenshot of the receiver's graphical user interface is depicted on the right hand side of Figure 2. At the top, it shows a log of the received frames in Wireshark, including meta data like the modulation and encoding scheme and a constellation plot on the bottom (showing 16-QAM symbols in this case). Apart from logging frames, it is possible to pipe them in a TUN/TAP interface, a virtual network device, and, thus, to connect the SDR to the Linux TCP/IP stack. This way the SDR can be used like a normal network interface.

4 Simulations

In a first step, we validate our implementation of IEEE 802.11p in a simulation study by measuring the frame error rate over an AWGN channel. Apart from highlighting its applicability for simulations in the first place, this provides an initial performance evaluation and allows to compare our implementation with independent results. In this

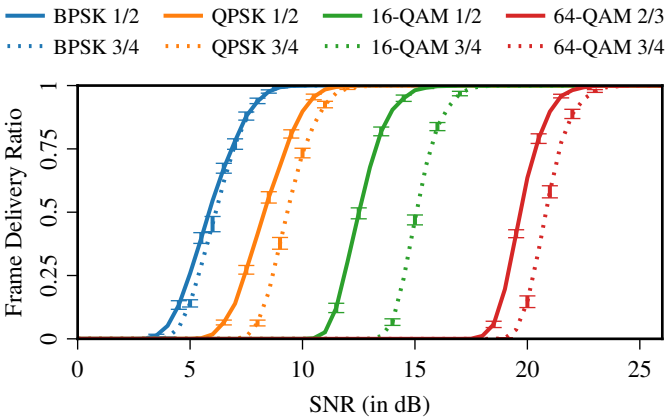


Figure 5. Frame delivery ratio of 435 Byte frames simulated over an AWGN channel.

simulations and all further experiments we use a frame size of 435 Byte, which we selected to show the performance with an intermediate sized frame as opposed to very small or very large ones. Furthermore, we consider 435 Byte to be about the size of an ETSI ITS-G5 Cooperative Awareness Message (CAM) [41], including all certificate information. The actual size of a CAM can, however, vary significantly as

- its content depends on the type of the sender, i.e., the format is different for a vehicle and a construction site;
- it may contain low-frequency information that is not sent with every frame; and
- the certificate information, which is used to validate the signature of the message, can be a hash of a previously transmitted certificate or a complete certificate chain with multiple intermediate Certificate Authorities (CAs).

Figure 5 plots the frame delivery ratio of all modulation and coding schemes for different SNRs. The error bars show the 95 % confidence intervals. At a very basic level, the graph shows that the performance is reasonable in the sense that higher order schemes need a higher SNR to work reliably. The graph is, however, more interesting if we compare the absolute positions of the curves with independent results. Since these results are from experiments with different frames sizes, we reran our simulations with corresponding parameters to allow direct comparison. For the sake of brevity, we omit additional graphs for these very similar simulations.

In [11], Mittag et al. present an IEEE 802.11a/g/p physical layer simulation model for the ns-3 network simulator. The authors model the physical layer at signal level and, thus, use the same level of abstraction as an SDR. The difference is that their implementation is not real-time capable and, thus, limited to simulations only. The authors present error curves from experiments in a testbed and from simulations using their model, both matching very well our simulation results.

In [42], Pei and Henderson validate the widely used WLAN frame error rate model of the ns-3 simulator. The paper presents an analytical model for calculating the frame error rate that is validated by experiments in a testbed. Again, their results match very well with ours, increasing

Table 1
Overview of the most important components of our SDR setup.

Component	Type
Operating System	Ubuntu 15.04 and OSX 10.11
GNU Radio	Version 3.7.7
UHD	Version 3.8.4
SDR	Ettus Research N210 (revision 4)
Daughterboard	XCVR 2450
Frequency	Channel 178 (5.89 GHz)
Bandwidth	10 MHz and 20 MHz

Table 2
WLAN cards and IEEE 802.11p prototypes used in our interoperability tests.

NIC	Standard	Bandwidth	
MacBook Pro/Air	802.11a/g	20 MHz	✓
Intel Ultimate-N 6300	802.11a/g	20 MHz	✓
Air Live X.USB	802.11a/g	20 MHz	✓
Cohda MK2/MK5	802.11p	10 MHz	✓
UNEX DCMA-86P2	802.11a/p	10/20 MHz	✓

the confidence in the correctness of our implementation and showing that we achieve reasonable performance.

5 Interoperability

In simulations, we fed back the generated sample stream directly into our receiver. While we have seen that this provides reasonable results, we did not prove that our implementation is standard compliant and works with other devices. To show that this is indeed the case, we conduct extensive interoperability test with both commercial WLAN cards and IEEE 802.11p prototypes. Table 1 summarizes the SDR setup that we use in our interoperability tests as well as all following experiments. The setup is based on GNU Radio 3.7 and an Ettus Research N210 equipped with an XCVR 2450 daughterboard, which supports half-duplex operation in the 2.4 GHz and 5 GHz band. In these experiments, we do not rely on any hardware-specific features. Therefore, it is possible to replace the N210 with another SDR platform that offers the required bandwidth and covers the frequency band of interest. In fact, our implementation also works with the B210 from Ettus Research and the HackRF from Great Scott Gadgets, for example.

IEEE 802.11a/g/p are very similar standards that differ only in bandwidth and frequency. Given its ubiquity, we test WLAN first using off-the-shelf cards that are available in our lab. Table 2 list some of them, including widely used chipsets from Intel as well as the chips from Apple products used, e.g., in the MacBook Air. We test the cards in IEEE 802.11g mode in the 2.4 GHz band as well as in IEEE 802.11a mode in the 5 GHz band. Managing to establish bidirectional communication with these devices using either modulation and coding scheme shows the correctness of the implementation and proves that the SDR produces standard compliant frames.

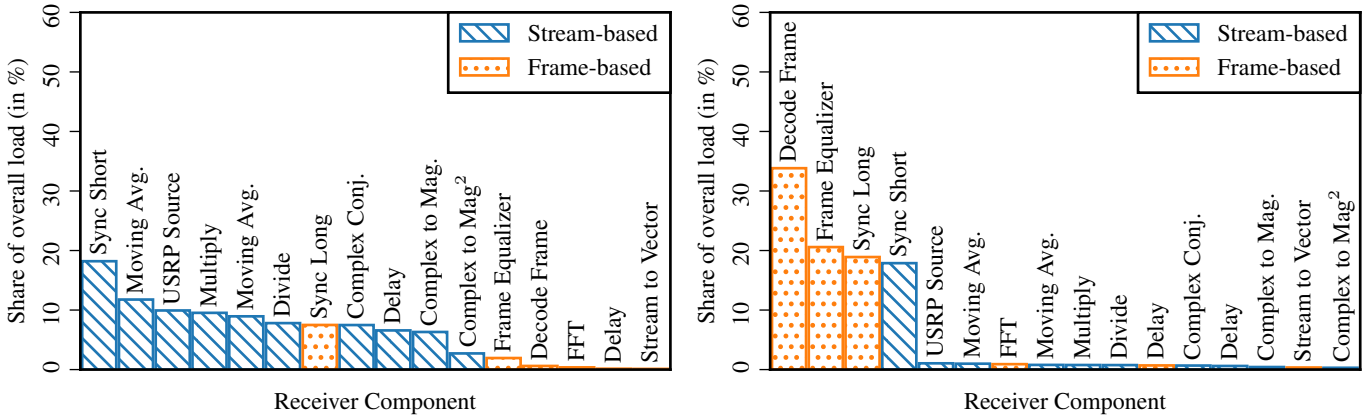
As we expect VANETs to be one of the main application areas of our transceiver, we also conduct interoperability test with different IEEE 802.11p prototypes. The first prototype uses a UNEX DCMA-86P2, a commercial IEEE 802.11p-capable MiniPCI WLAN card that has been successfully used by many of the Grand Cooperative Driving Challenge (GCDC) participants. This card is based on an Atheros chip that is supported by the *ath5k* Linux driver. To enable IEEE 802.11p operation, we adapt the Linux kernel driver to switch to 10 MHz mode and remove regulatory restrictions to tune to the ITS channels in the 5.9 GHz band. Apart from the UNEX DCMA-86P2, we test an *ath9k*-based Atheros card, which is supported by the official Linux IEEE 802.11p implementation and, therefore, does not need manual patching. The third device is a Cohda Wireless MK5, an integrated IEEE 802.11p prototype that features communication stacks for both IEEE DSRC/WAVE and ETSI ITS-G5. The MK5 and its predecessors are well-known in the research community and were used in major field tests in USA, Australia, Germany, France, and Korea. Designed for ITS, the MK5 is a complete On Board Unit (OBU) that features, for example, a CAN bus interface and a sophisticated IEEE 802.11p transceiver that supports multi-antenna configurations. Again, we manage to set up bidirectional communication, with all prototypes, supporting our claim of a standard compliant physical layer implementation. To make sure that our physical layer implementation is standard compliant, we tested transmission and reception with all modulation and coding schemes defined by the IEEE 802.11 a/g/p standard (see Table 3). Furthermore, with going beyond simulations, these test show that we implemented a receiver that is able to deal with impairments of real hardware and unsynchronized clocks.

Table 3
Our transceiver supports all modulation and coding schemes defined in the IEEE 802.11a/g/p standard.

Modulation	Code Rate	Transmission	Reception
BPSK	1/2, 3/4	✓	✓
QPSK	1/2, 3/4	✓	✓
16-QAM	1/2, 3/4	✓	✓
64-QAM	2/3, 3/4	✓	✓

6 Computational Performance

When it comes to GPP-based SDR systems, the computational performance and the ability to process samples in real-time are critical factors. In this context, we refer to the term *real-time* signal processing to contrast offline signal processing. It implies that the PC is able to keep up with the incoming sample stream without dropping samples. In other words, the average processing time per sample is smaller than the sample duration. This property is crucial, since, otherwise, the transceiver would have to drop samples, which causes packet loss. Ultimately, this could lead to wrong interpretations of measurement results if lost frames are regarded as effects of the wireless channel or shortcomings of receive algorithms. Given the importance, we have an in-depth look at the computational complexity and real-time capabilities of our system.



(a) 435 Byte QPSK $^1/_2$ frames at a rate of 10 Hz.

(b) Fully saturated channel with 1500 Byte 64-QAM $^3/_4$ frames.

Figure 6. Computational demands of individual signal processing blocks.

In that regard, the transmit side is not critical as the whole waveform can be pre-computed and streamed to the SDR. The only requirement is that the stream does not stall, which is, however, no problem in practice. The receive side is much more challenging as it has to process a large number of samples in real-time. An IEEE 802.11p channel with a bandwidth of 10 MHz, for examples, results in 10×10^6 complex floating point numbers per second.

To cope with such high bandwidths, GNU Radio starts each signal processing block in its own thread. During runtime, each block monitors performance related metrics like CPU time and fill state of the input and output queues [43]. Depending on the hardware platform and the operating system, there are several methods available to log the CPU time. We use the accurate *thread clock*, which takes into account only the time when the thread was actually scheduled by the operating system.

To capture the data, we implemented an application that connects to the transceiver while it is running. After it establishes the connection it resets the performance data of the transceiver, waits for a configurable amount of time, and prints the data in CSV format. We used this application to study the computational complexity of the individual receiver components. For the measurements, we ran the receiver on a desktop PC with an Intel i7-7700k CPU and 16 GByte RAM. The operating system was Ubuntu 16.04 running a Linux 4.4.0 kernel. We compiled GNU Radio and our transceiver with GCC 5.4.0 in *release mode*, which enables all standard compliant run time optimizations. While the quantitative results vary dependent on the system, we have seen similar qualitative results, for example, on other Linux laptops and a MacBook Air running macOS.

In a first test, the receiver decoded 435 Byte QPSK $^1/_2$ frames, sent at a rate of 10 frames per second and a bandwidth of 10 MHz. To have absolute accurate timing, we generated the sample stream with the proper interframe space in advance and used another SDR to replay these samples. We ran the receiver over 30 s and logged the CPU time of the individual components. The first notable observation is that the receiver did not drop samples and was able to decode all frames. This means that all samples were processed and all frames traversed the whole signal processing chain. The results are depicted in Figure 6a, where we plot the share of the overall CPU time per block.

The style of the bars corresponds to different parts of the receiver. The first part labeled "Stream-based" comprises blocks that are involved in frame detection. These blocks have to process all samples to search for the cyclic pattern of the short training sequence and decide whether subsequent decoding steps should be triggered. Their load scales, therefore, linearly with the bandwidth of the signal, i.e., a 5 MHz signal would cut their computational demands in half.

The second part labeled "Frame-based" comprises blocks that are involved in decoding frames. Depending on the signal processing task, these blocks scale with different frame parameters:

- The *Sync Long* block causes constant overhead per frame as it mainly aligns with the frame using cross-correlation with the known preamble.
- The *Equalize Symbols* block implements the equalizer, which mainly scales with the number of OFDM symbols, i.e., frame duration.
- The *Decode MAC* block works on bits and, therefore, scales with the length of the data payload.

By understanding the computational complexity of individual blocks, it is possible to adapt the experiment to the available processing power.

In our initial experiment this was not required as an average PC can easily decode IEEE 802.11p frames in real-time. We tested the same configuration also with a laptop and a MacBook Air (both with 8 GByte RAM and an Intel i5 CPU) without frame loss. This shows that even laptops are real-time-capable for low-traffic scenarios. On the desktop, this configuration leaves more than enough headroom to experiment with more complex signal processing algorithms.

Apart from this low-traffic scenario, we were curious to see if our implementation can cope with the most challenging conditions, i.e., a fully saturated channel. In this experiment, we create a sample stream with 1500 Byte 64-QAM $^3/_4$ frames. Using the modulation and coding scheme with the highest data rate, we maximize the computational load for the receiver. In addition to that, we separate the frames by only 58 µs, which corresponds to the minimum interframe space for IEEE 802.11p. We do not use any backoff slots, but

always use the minimum value to create the most challenging scenario.

Also in this experiment, we ran the receiver to monitor the load of each component. The results are depicted in Figure 6b. Again, the receiver did not drop any samples, which means that we are able to process even a saturated channel in real-time. As expected, the frame-based receiver components are responsible for a large part of the overall load. The *Decode MAC* block, which includes the Viterbi decoder, causes the highest load. With about 34 % of the overall load, it would currently present the bottle-neck when aiming for higher data rates. In the 30 s experiment, it occupied one CPU core nearly exclusively with a CPU time of 27 s.

Finally, we wanted to rule out that per-frame overhead could be a limiting factor. We, therefore, performed a similar experiment with short 64-QAM $^3/_4$ frames, containing only the MAC and LLC, but no data payload. Also these frames were sent with an interframe space of 58 µs. Like in the previous experiments, the receiver was able to keep up with the sample stream, highlighting the real-time capability of our implementation.

7 Time-Critical Functionality

Having a physical layer implementation that works well with other devices, we are interested in which level of standard compliance we can reach with a GPP-based SDR. It is well-known that the disadvantage of this architecture is the delay, introduced by streaming the samples to the PC, and the jitter, introduced by processing the signal on a non-real-time operating system [44]. It is, therefore, hardly possible to acknowledge a unicast frame or reply to a Request To Send (RTS) in only 32 µs as mandated by the standard [45].

In both cases, the main problem is that the receiver has to decode the frame before it can react by sending an Acknowledgement Frame (ACK) or a Clear To Send (CTS). Meeting those timings would require to implement the whole receiver on the FPGA. Fortunately, there are some time-critical functions, where the receiver does not have to decode the frame, but merely to detect its presence. Examples for such functions are Automatic Gain Control (AGC) and channel access for broadcast frames. In the following, we show that it is indeed possible to implement such limited functionality on the FPGA without giving up advantages of a software implementation. Using what Nychis et al. termed a *split-functionality approach* [44], we implement time-critical functionality on the FPGA while leaving the main physical layer implementation in software. Following this approach, we explore the possibilities of the architecture without sacrificing the accessibility of the physical layer implementation. This design decision differentiates our transceiver from, for example, Microsoft's Sora.

7.1 Automatic Gain Control

Since transceivers often receive frames from multiple sources, the power level can vary significantly between frames. This is especially relevant in VANETs, where the maximum transmit power is higher than in ordinary WLAN networks, resulting in larger differences between high and low power frames. Given these varying input power levels, a static gain cannot provide optimal performance, as either weak frames do not take full advantage of the dynamic range of the Analog-to-Digital Converter (ADC), resulting in quantization noise, or high power frames saturate the receiver and introduce distortions. With AGC, the receiver measures the power during frame detection and adjusts the gain to an optimal level that balances the trade-off between quantization noise and distortions.

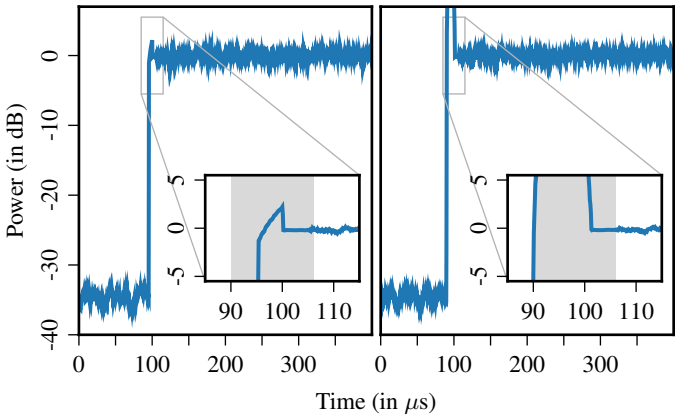


Figure 7. Power over time for a low-power frame that got amplified by AGC (left) and a high-power frame that got attenuated (right).

To find such level, we send frames from one SDR to another while varying the gain at both sender and receiver. Using the results from this experiment, we determine an optimal input power level empirically. To implement AGC, we extend the FPGA of the Ettus N210 to control the gain of the XCVR 2450 daughterboard. The daughterboard uses a MAX2829 transceiver, which allows controlling the gain directly via pins. While readjusting the gain, the transceiver introduces considerable distortions, but resettles after only 40 ns [46], which is short compared to 16 µs of the short training sequence. This is an important performance characteristic, as all gain adjustments have to happen during the short training sequence, since the following long training sequence is used for initial channel estimations.

To realize AGC, we extend the FPGA design of the N210 with VHDL modules to calculate the autocorrelation coefficient and employ it for frame detection using the same algorithm as on the host (cf. Section 3). Once a frame is detected, we use a lookup table to map its input power level to a gain setting that adjusts it as close as possible to the optimal value.

As demonstrated in [4], we use a commercial WLAN card to test the performance of our implementation by sending frames at different power levels and recording the signal after gain control on a PC. The results are depicted in Figure 7. Both plots show the power level during frame start. As the N210 is not calibrated to measure absolute power, we plot the relative power with regard to the target power level on the y-axis. The left hand side corresponds to a low-power frame that was amplified by the AGC to reach the desired level, while the right hand side shows a high-power frame that got attenuated. We can see that after initial adjustments, both frames settle at the same power level. In the inset, we zoom in at the very start of the frame and shade the short training sequence in gray. This shows

Table 4
Default EDCA parameter set [45, Table 8-106].

AC	CW_{min}	AIFS
Background	$aCW_{min} = 15$	149 µs
Best Effort	$aCW_{min} = 15$	110 µs
Video	$(aCW_{min} + 1)/2 - 1 = 7$	71 µs
Voice	$(aCW_{min} + 1)/4 - 1 = 3$	58 µs

that the receiver stabilizes during the short training sequence and does not cause distortions thereafter. All frames were perfectly receivable, showing the feasibility of the approach.

7.2 Channel Access

The second time-critical functionality that we studied [3] is channel access for broadcast frames. By limiting the scope to broadcasts, we omit dependent transmissions like ACKs, which cannot be realized with an N210. Fortunately, this limitation is not problematic in the context of VANETs, as they rely mostly on broadcasts. In fact, it was recently shown that unicast transmissions can cause considerable performance degradation [47].

IEEE 802.11p uses Enhanced Distributed Channel Access (EDCA) [45], which supports QoS through traffic classes that range from *voice*, with the highest priority, to *best effort* with the lowest priority. In a nutshell, EDCA works as follows: The traffic classes differ in their average channel access delay, where high-priority frames, for example, wait shorter on average and are, thus, likely to be sent first. When a frame is scheduled for transmission, it waits until the channel is free for a period called Arbitration Inter Frame Space (AIFS) before it starts counting down a random number of 13 µs slots. The number of slots is chosen uniformly between zero and a congestion window of CW_{min}. Both AIFS and CW_{min} depend on the traffic class of the frame. Their default parameters for IEEE 802.11p are listed in Table 4. If the channel turns busy at some point during this procedure, the sender restarts the process, i.e., it waits again for an AIFS period before it continues to count down slots. Once the slots reach zero, the frame is sent.

To realize this functionality on the FPGA, we implement modules that calculate the power level to decide whether the channel is free or busy and add a state machine that coordinates channel sensing and timings. The frames are still generated on the PC and transferred to the N210 where they are kept in memory until the state machine triggers their transmission. Per-frame parameters like AIFS and back off slots are generated on the PC and attached as metadata.

To show the feasibility of the approach and to validate our implementation, we start with an experiment to test channel sensing and timing accuracy. Using one SDR to generate noise and block the channel, we pre-load a WLAN frame on another SDR. The frame is configured to be sent with zero backoff slots and the shortest inter frame space, i.e., the AIFS of the voice traffic class, which is 58 µs. With a third SDR, we monitor the channel while switching off the noise source. The channel utilization of this experiment is depicted in Figure 8. The first 100 µs show the channel blocked by noise. After switching off the noise source, the channel remains idle for 59.8 µs before the WLAN frame is sent. Receiving the frame

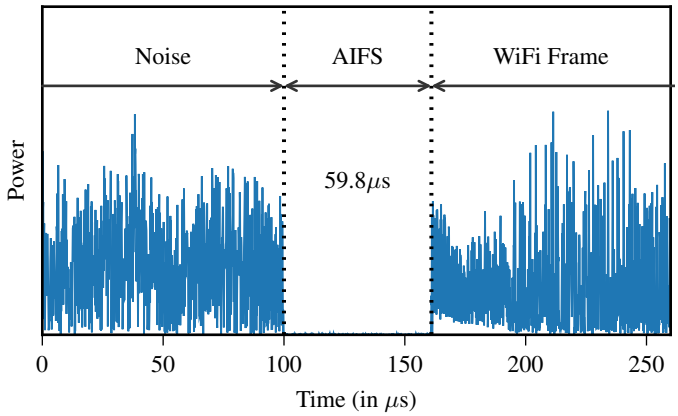


Figure 8. Channel utilization to verify channel sensing and AIFS timing.

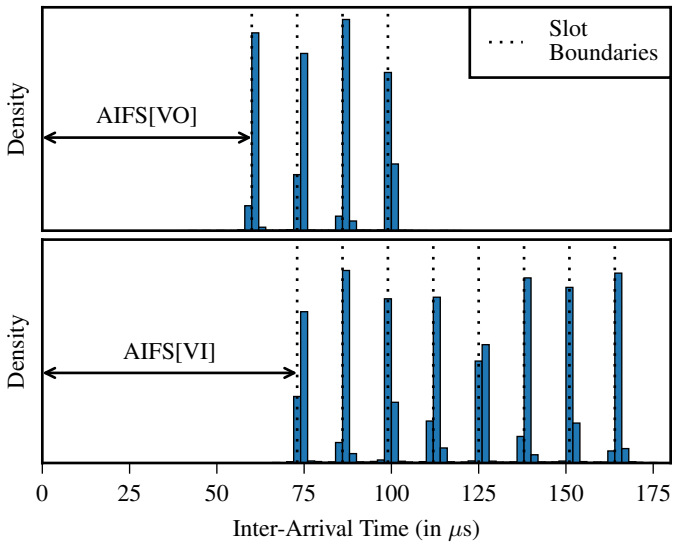


Figure 9. Distribution of inter-arrival times when saturating the channel with frames of the access category voice (top) and video (bottom).

with another receiver, we asserted that it was not corrupted by the Carrier Sense Multiple Access (CSMA) process.

The results from this experiment indicate that channel sensing and delaying work as expected. In fact, the time gap is close to the target delay of 58 µs. The additional 1.8 µs can be explained by 1 µs RX-TX turnaround time of the half-duplex transceiver [46] and a 0.8 µs time window used to average power values in the channel sensing module. As the additional delay is similar for every frame, we could subtract it from the inter frame space. This is, however, not needed since already the current implementation is below the 2 µs *aRxTxTurnaroundTime* mandated by the standard.

In a second experiment, we saturate the channel with frames of a given access category and measure their inter-arrival times. Using an idle channel, the sender never has to back off, but is able to send at maximum rate. In that scenario channel access should work as follows: After every frame, the sender enters a post-transmit backoff state, waiting for an AIFS plus a random number of backoff slots. This mechanism avoids that a single transmitter captures the channel, sending frames back-to-back. To verify the distribution of the inter-arrival times, we use a UNEX card to log the frames and record their arrival times in the receive interrupt handler.

Figure 10. Photo of the experiment setup. The receivers use 6 dBi dipole antennas.

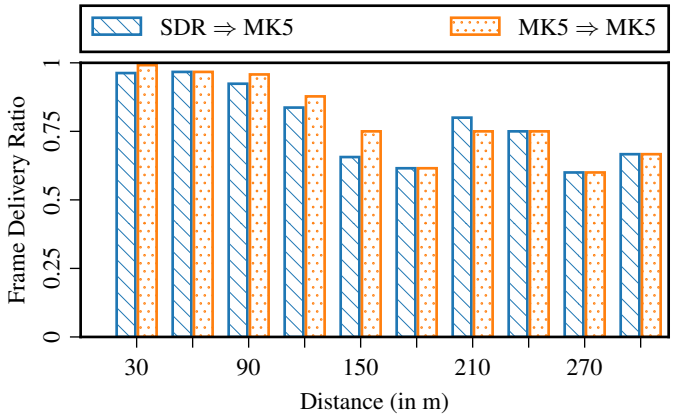


Figure 11. Packet delivery ratio of frames sent with an MK5 and an SDR, using another MK5 as a reference receiver.

We repeat the experiment with all access categories, but, for the sake of brevity, include only results for voice and video. Figure 9 shows the distribution of their inter-arrival times for these traffic classes. Each plot is based on over 30 000 frames. We can see that the histograms match very well with the ideal distributions, as all time slots are used with about the same probability and the timings match well with the inter-arrival times defined in the standard (indicated as dashed gray lines in the plot). Furthermore, as there are no larger inter-arrival times, this also shows that it is possible to fully saturate the channel.

In a final experiment, we study MAC layer interoperability with a commercial WLAN card by testing whether each device gets its fair share of the channel. We saturate the channel with a UNEX card and an SDR, both sending frames of the same access category. Using a second UNEX card, we monitor transmissions and calculate the bandwidth over time. The results (graphs omitted for the sake of brevity) attest a fair share of the channel between both devices and, thus, MAC layer interoperability between the SDR and the commercial card.

To summarize, our experiments show that, with minor modifications to the FPGA, it is possible to implement AGC and channel access for broadcast transmissions. Following the split functionality approach, we preserve the advantages of a software implementation, while extending standard compliance beyond the physical layer.

8 Field Test

To highlight the applicability of the transceiver for experiments on the road, we conduct a field test near Paderborn, Germany. For this test, we equip two cars with multiple IEEE 802.11p prototypes, as shown in Figure 10. On the transmit side, we use a Cohda Wireless MK5 and our SDR implementation with an Ettus N210 using an XCVR 2450 daughterboard. While both transmitters are in no way calibrated measurement devices, we set their output power to about 10 dBm. On the MK5, this can be configured, while on the SDR we adjust the gain and the signal amplitude, assuming a maximum output power of 20 dBm, which is listed as a typical value in the datasheet. Both devices are controlled by a laptop, which triggers frame transmissions alternating on the MK5 and the SDR, leaving enough timing margins to avoid collisions between the devices. Like in simulations, we use 435 Byte frames to resemble an average-sized ITS frame. We set the modulation and coding scheme to QPSK ½, as we do not want to use the simplest scheme, but show that the receiver is able to deal with higher order modulations. Furthermore, QPSK ½ was found to provide a good compromise between throughput and robustness, and it represents the default modulation and coding scheme for periodic awareness messages in ETSI ITS-G5 [48].

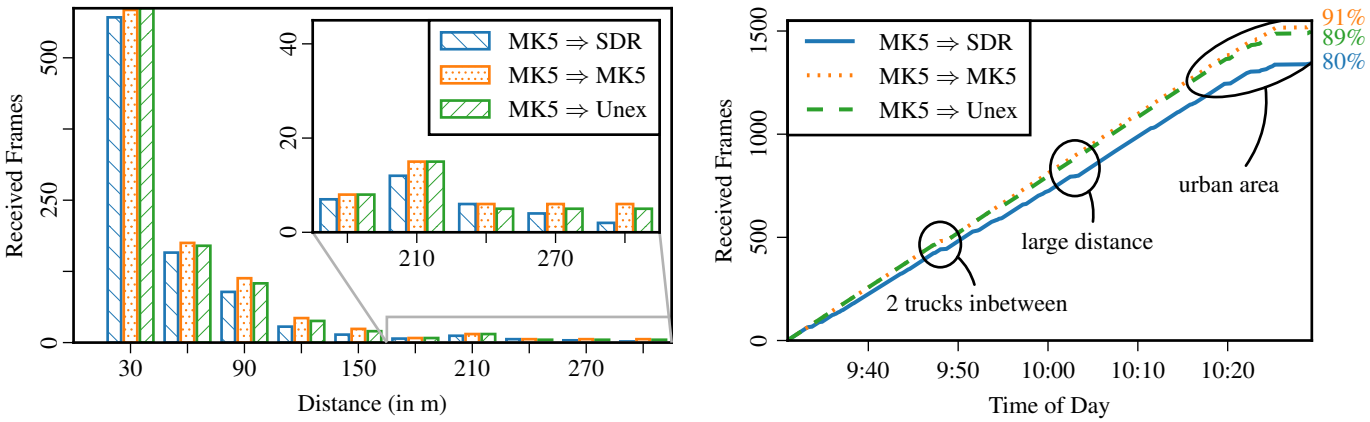
On the receive side, we use another MK5, a similar SDR setup, and, additionally, a UNEX DCMA-86P2. The receivers use 9 dBi dipole antennas mounted at the center of the roof (cf. Figure 10). Note that each receiver uses its own antenna and, thus, experiences independent fast fading effects. Thus, we compare average reception rates and not individual frames.

Both cars log their positions every 0.5 s with a NEO-7N, a high precision GPS receiver from u-blox. The positions at frame transmissions are later linearly interpolated based on the GPS time series. During the measurements, the cars drive around in diverse surroundings, ranging from open fields, to rural areas, to city environments. The speed was up to 70 km/h, but mostly around 40 km/h.

8.1 Transmit Performance

The first aspect that we investigate is the ability to transmit standard compliant IEEE 802.11p frames. Using the Cohda Wireless MK5 as the receiver, we plot the packet delivery ratio of frames generated by the SDR and the MK5 at different distances in Figure 11. Here and in the following plots, we bin the data in 10 intervals ranging from 15 m to 300 m, excluding short distance frames recorded on the parking lot. The graph in Figure 11 conveys two important messages. First, our SDR transceiver is able to generate standard compliant IEEE 802.11p frames that are received by the MK5, not only under idealized lab conditions, but also in a realistic scenario. Second, both transmitters show similar performance, highlighting the applicability of the SDR transceiver for field tests.

These measurements should not be mistaken for a performance comparison between the SDR and the MK5, as this would require more precise power calibration and the exact same antenna placement as the position on the roof can have a significant impact [10], [49]. Apart from performance



(a) Packet delivery ratio of IEEE 802.11p frames

(b) Cumulative number of packets received over time.

Figure 12. Receive performance using a Cohda Wireless MK5 as a sender.

characteristics, the plot shows that the environment naturally led to packet loss, as it is typical under realistic conditions.

8.2 Receive Performance

Having tested the ability to send standard compliant frames, we study the receive performance of our SDR prototype. Using the MK5 as sender, we plot the number of frames received by the SDR, the MK5, and the UNEX card in Figure 12a. The inset shows a zoomed version of distances between 150 m and 300 m where the sample size is relatively low. The main message of the plot is that all three transceivers show comparable performance in our field test. Based on advanced hardware and software, the MK5 provides a slightly higher reception rate. This is in accordance to with independent experiments that compared a predecessor of the MK5 with off-the-shelf cards [20].

Another view on the same data is shown in Figure 12b, where we plot the cumulative number of received packets over time. Since we sent frames at a constant rate, the total number would yield a straight line under ideal conditions. The plot, however, shows regions with a lower slope, indicating packet loss. Correlating these losses with events that we logged during the experiment shows that the losses seem reasonable in the sense that they are caused by shadowing, a large gap between sender and receiver, and challenging multi-path environments. In other words, the SDR does not suffer from systematic, random errors, but dropped frames in challenging environments. In addition, we annotate the overall ratio of received packets per receiver on the right hand side of the plot to allow a quantitative comparison. It shows that the SDR lost 20 % of all frames, while the MK5 and the UNEX card lost around 10 %. The difference can be explained by the use of more sophisticated receive algorithms in the commercial device and the prototype. The MK5, for example, is likely based on a channel estimation algorithm published by Cohda Wireless employees [7]. The lower reception rate of the SDR, in turn, is no general weakness of the approach, but reflects known limitations of the LS equalizer in dynamic channels [10], [40].

9 Use Cases and Conclusion

In summary, we presented and evaluated a simulation and experimentation framework for IEEE 802.11p. The framework is built around an SDR-based IEEE 802.11p transceiver that we verified through extensive interoperability tests with VANET prototypes and consumer grade WLAN cards. The physical layer of the transceiver is implemented completely in software, making it easy to use, modify, and debug. Despite the delay challenges of the SDR architecture, we showed that with minor modifications to the FPGA it is possible to realize standard compliant channel access and even AGC. Ultimately, the applicability of our transceiver for real-world experiments was shown in a field test. To foster its use and to allow reproduction of the results, we released the transceiver under an Open Source license.[1]

We believe that our SDR-based IEEE 802.11p transceiver will serve as a valuable tool to study many aspects of VANETs. In fact, it has already been used for privacy-related research, where it provided access to information that is not available with normal WLAN cards [34], and large-scale vehicular network simulations, where it provided a realistic physical layer simulation model [19]. We think, however, that its most interesting applications in the signal processing context have yet to be explored. Given the fact that it can be used for simulations and experiments, it allows for a workflow that other tools do not offer. The physical layer is implemented in a high-level programming language and is, therefore, well suited for rapid prototyping of signal processing algorithms. Using simulations, these implementations can be verified and algorithms can be tested in a reproducible manner. Since the performance of the transceiver can be assessed with different simulation models for the wireless channel, interference, and hardware impairments, it is possible to get an in-depth understanding of how the system behaves in diverse environments. Finally, the switch to lab measurements and field tests is seamless and merely a slight reconfiguration of the flow graph. With the possibility to conduct experiments, researchers can easily proof the feasibility of an idea or algorithm. Another benefit stems from the fact that field tests often provide new insights that in a kind of feedback loop can lead to new ideas that, again, can be studied through simulations. Such iterative workflow is barely possible with other prototypes and, in our opinion, a big advantage of our approach.

1. https://www.wime-project.net

REFERENCES

[1] 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, Aug. 2013, pp. 9–16.

[2] ——, "Towards an Open Source IEEE 802.11p Stack: A Full SDR-based Transceiver in GNURadio," in *5th IEEE Vehicular Networking Conference (VNC 2013)*. Boston, MA: IEEE, Dec. 2013, pp. 143–149.

[3] B. Bloessl, A. Puschmann, C. Sommer, and F. Dressler, "Timings Matter: Standard Compliant IEEE 802.11 Channel Access for a Fully Software-based SDR Architecture," in *20th ACM International Conference on Mobile Computing and Networking (MobiCom 2014), 9th ACM International Workshop on Wireless Network Testbeds, Experimental evaluation and Characterization (WiNTECH 2014)*. Maui, HI: ACM, Sep. 2014, pp. 57–63.

[4] B. Bloessl, C. Sommer, and F. Dressler, "Power Matters: Automatic Gain Control for a Software Defined Radio IEEE 802.11a/g/p Receiver," in *34th IEEE Conference on Computer Communications (INFOCOM 2015), Demo Session*. Hong Kong, China: IEEE, Apr. 2015, pp. 25–26.

[5] C. Sommer and F. Dressler, *Vehicular Networking*. Cambridge University Press, Nov. 2014.

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

[7] P. Alexander, D. Haley, and A. Grant, "Outdoor Mobile Broadband Access with 802.11," *IEEE Communications Magazine*, vol. 45, no. 11, pp. 108–114, Nov. 2007.

[8] K. K. Nagalapur, F. Brännström, and E. G. Ström, "On Channel Estimation for 802.11p in Highly Time-Varying Vehicular Channels," in *IEEE International Conference on Communications (ICC 2014)*. Sydney, Australia: IEEE, Jun. 2014, pp. 5659–5664.

[9] J. A. Fernandez, K. Borries, L. Cheng, B. V. K. Vijaya Kumar, D. D. Stancil, and F. Bai, "Performance of the 802.11p Physical Layer in Vehicle-to-Vehicle Environments," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 1, pp. 3–14, Jan. 2012.

[10] C. F. Mecklenbräuker, A. F. Molisch, J. Karedal, F. Tufvesson, A. Paier, L. Bernadó, T. Zemen, O. Klemp, and N. Czink, "Vehicular Channel Characterization and its Implications for Wireless System Design and Performance," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1189–1212, Jul. 2011.

[11] 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, Jul. 2011.

[12] S.-K. Lee, Y.-A. Kao, and H.-W. Chen, "Performance of A Robust Inner Receiver with Frequency Domain LMS Equalizer for DSRC Systems," in *International Wireless Communications and Mobile Computing Conference 2006 (IWCMC'06)*. Vancouver, Canada: ACM, Jul. 2006, pp. 985–990.

[13] Y. Zhang, I. L. Tan, C. Chun, K. Laberteaux, and A. Bahai, "A Differential OFDM Approach to Coherence Time Mitigation in DSRC," in *5th ACM International Workshop on Vehicular Inter-Networking (VANET 2008)*. San Francisco, CA: ACM, Sep. 2008, pp. 1–6.

[14] F. A. Teixeira, V. F. e Silva, J. L. Leoni, D. F. Macedo, and J. M. Nogueira, "Vehicular networks using the IEEE 802.11p standard: An experimental analysis," *Elsevier Vehicular Communications*, vol. 1, no. 2, pp. 91–96, Apr. 2014.

[15] J. Santa, F. Pereñíguez, A. Moragón, and A. F. Skarmeta, "Experimental evaluation of CAM and DENM messaging services in vehicular communications," *Elsevier Transportation Research Part C: Emerging Technologies*, vol. 46, pp. 98–120, Sep. 2014.

[16] A. B. Reis, S. Sargento, F. Neves, and O. K. Tonguz, "Deploying Roadside Units in Sparse Vehicular Networks: What Really Works and What Does Not," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 6, pp. 2794–2806, Jul. 2014.

[17] F. Dressler, H. Hartenstein, O. Altintas, and O. K. Tonguz, "Inter-Vehicle Communication - Quo Vadis," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 170–177, Jun. 2014.

[18] C. Sommer, J. Härri, F. Hrizi, B. Schünemann, and F. Dressler, "Simulation Tools and Techniques for Vehicular Communications and Applications," in *Vehicular ad hoc Networks - Standards, Solutions, and Research*, C. Campolo, A. Molinaro, and R. Scopigno, Eds. Springer, May 2015, pp. 365–392.

[19] D. Maier, S. Moser, and F. Slomka, "Deterministic Models of the Physical Layer Through Signal Simulation," in *8th International Conference on Simulation Tools and Techniques (SIMUTools 2015)*. Athens, Greece: ICST, Aug. 2015, pp. 175–182.

[20] H. Rakouth, P. Alexander, A. Brown Jr., W. Kosiak, M. Fukushima, L. Ghosh, C. Hedges, H. Kong, S. Kopetzki, R. Siripurapu, and J. Shen, "V2X Communication Technology: Field Experience and Comparative Analysis," in *FISITA World Automotive Congress*. Beijing, China: Springer, Nov. 2012, pp. 113–129.

[21] M. Boban, T. Vinhosa, J. Barros, M. Ferreira, and O. K. Tonguz, "Impact of Vehicles as Obstacles in Vehicular Networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 1, pp. 15–28, Jan. 2011.

[22] T. Mangel, O. Klemp, and H. Hartenstein, "A Validated 5.9 GHz Non-Line-of-Sight Path-Loss and Fading Model for Inter-Vehicle Communication," in *11th International Conference on ITS Telecommunications (ITST 2011)*. St. Petersburg, Russia: IEEE, Aug. 2011, pp. 75–80.

[23] C. Sommer, S. Joerer, and F. Dressler, "On the Applicability of Two-Ray Path Loss Models for Vehicular Network Simulation," in *4th IEEE Vehicular Networking Conference (VNC 2012)*. Seoul, Korea: IEEE, Nov. 2012, pp. 64–69.

[24] A. Geiger, M. Lauer, F. Moosmann, B. Ranft, H. Rapp, C. Stiller, and J. Ziegler, "Team AnnieWAY's entry to the 2011 Grand Cooperative Driving challenge," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1008–1017, Apr. 2012.

[25] S. Laux, G. S. Pannu, S. Schneider, J. Tiemann, F. Klingler, C. Sommer, and F. Dressler, "OpenC2X - An Open Source Experimental and Prototyping Platform Supporting ETSI ITS-G5," in *8th IEEE Vehicular Networking Conference (VNC 2016), Demo Session*. Columbus, OH: IEEE, Dec. 2016, pp. 152–153.

[26] 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*, vol. 12, no. 1, pp. 56–58, Jan. 2008.

[27] Y. Shin, G. Seo, S. Woo, K. Ko, and C. Mun, "Demo: Implementation of IEEE 802.11p transceiver Using USRP-RIO By LabVIEW Communications," in *7th IEEE Vehicular Networking Conference (VNC 2015)*. Kyoto, Japan: IEEE, Dec. 2015, pp. 177–178.

[28] G. Sklivanitis, A. Gannon, S. N. Batalama, and D. A. Pados, "Addressing Next-Generation Wireless Challenges with Commercial Software-Defined Radio Platforms," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 59–67, Jan. 2016.

[29] T. Vilches and D. Dujovne, "GNURadio and 802.11: Performance Evaluation and Limitations," *IEEE Network*, vol. 28, no. 5, pp. 27–31, Sep. 2014.

[30] K. Tan, H. Liu, J. Zhang, Y. Zhang, J. Fang, 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, Jan. 2011.

[31] T. W. Rondeau, "On the GNU Radio Ecosystem," in *Opportunistic Spectrum Sharing and White Space Access: The Practical Reality*, O. Holland, H. Bogucka, and A. Medeisis, Eds. Wiley, May 2015, pp. 25–48.

[32] B. Bloessl, M. Gerla, and F. Dressler, "IEEE 802.11p in Fast Fading Scenarios: From Traces to Comparative Studies of Receive Algorithms," in *22nd ACM International Conference on Mobile Computing and Networking (MobiCom 2016), 1st ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services (CarSys 2016)*. New York, NY: ACM, Oct. 2016.

[33] R.-A. Stoica, S. Severi, and G. T. F. de Abreu, "On Prototyping IEEE802.11p Channel Estimators in Real-World Environments Using GNURadio," in *2016 IEEE Intelligent Vehicles Symposium (IV)*. Gothenburg, Sweden: IEEE, Jun. 2016, pp. 10–15.

[34] B. Bloessl, C. Sommer, F. Dressler, and D. Eckhoff, "The Scrambler Attack: A Robust Physical Layer Attack on Location Privacy in Vehicular Networks," in *4th IEEE International Conference on Computing, Networking and Communications (ICNC 2015), CNC Workshop*. Anaheim, CA: IEEE, Feb. 2015, pp. 395–400.

[35] G. Arcos, R. Ferreri, M. Richart, P. Ezzatti, and E. Grampín, "Accelerating an IEEE 802.11 a/g/p Transceiver in GNU Radio," in *9th Latin America Networking Conference (LANC'16)*. Valparaíso, Chile: ACM, Oct. 2016, pp. 13–19.

[36] T. W. 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, Jun. 2012.
[37] 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, Mar. 2010, pp. 1–4.
[38] L. Chia-Horng, "On the design of OFDM signal detection algorithms for hardware implementation," in *IEEE Global Telecommunications Conference (GLOBECOM 2003)*. San Francisco, CA: IEEE, Dec. 2003, pp. 596–599.
[39] T. Schmidl and D. Cox, "Robust frequency and timing synchronization for OFDM," *IEEE Transactions on Communications*, vol. 45, no. 12, pp. 1613–1621, 1997.
[40] R.-A. Stoica, S. Severi, and G. T. Freitas de Abreu, "Learning the Vehicular Channel Through the Self-Organization of Frequencies," in *7th IEEE Vehicular Networking Conference (VNC 2015)*. Kyoto, Japan: IEEE, Dec. 2015, pp. 68–75.
[41] "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service," ETSI, EN 302 637-2 V1.3.2, Nov. 2014.
[42] G. Pei and T. R. Henderson, "Validation of OFDM Error Rate Model in ns-3," Boeing Research & Technology, Tech. Rep., 2010.
[43] 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, Aug. 2013, pp. 65–70.
[44] G. Nychis, T. Hottelier, Z. Yang, S. Seshan, and P. Steenkiste, "Enabling MAC Protocol Implementations on Software-Defined Radios," in *6th USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2009)*. Boston, MA: USENIX, Apr. 2009, pp. 91–105.
[45] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE, Std 802.11-2012, 2012.
[46] "MAX2828/MAX2829 Single-/Dual-Band 802.11a/b/g World-Band Transceiver ICs," Maxim Integrated, Datasheet Rev 0, Oct. 2004.
[47] F. Klingler, F. Dressler, and C. Sommer, "IEEE 802.11p Unicast Considered Harmful," in *7th IEEE Vehicular Networking Conference (VNC 2015)*. Kyoto, Japan: IEEE, Dec. 2015, pp. 76–83.
[48] "Intelligent Transport Systems (ITS); Vehicular communications; GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Sub-part 2: Media-dependent functionalities for ITS-G5," ETSI, TS 102 636-4-2 V1.1.1, Oct. 2013.
[49] A. Kwoczek, Z. Raida, J. Láčík, M. Pokorný, J. Puskely, and P. Vágner, "Influence of Car Panorama Glass Roofs on Car2car Communication," in *3rd IEEE Vehicular Networking Conference (VNC 2011), Poster Session*. Amsterdam, Netherlands: IEEE, Nov. 2011, pp. 246–251.

Bastian Bloessl [S] (bloessl@ccs-labs.org) is a researcher at the CONNECT Center, Trinity College Dublin, Ireland's Research Center for Future Networks and Communications, where he is funded through a Marie Skłodowska-Curie fellowship. He received his diploma in Computer Science from the University of Würzburg, Germany, in 2011. After his diploma, he started as a PhD student at the Computer and Communication Systems Group at the University of Innsbruck, Austria. In 2014, he moved with the group to Paderborn University, Germany, to continue his studies. In 2015, he won a FitWeltweit scholarship from the German Academic Exchange Service (DAAD), which funded a six-month stay in the research group of Prof. Mario Gerla at the Computer Science Department of the University of California, Los Angeles (UCLA). His research is focused on using software defined radio-based prototypes to assess the performance and robustness of vehicular and sensor networks.

Michele Segata [M] (msegata@disi.unitn.it) received his B.Sc. and M.Sc. in Computer Science from the University of Trento in 2009 and 2011, respectively. In 2016 he got a double PhD in Computer Science from the Universities of Innsbruck and Trento. His research mainly focuses on the development of communication protocols and highly realistic simulation models for platooning. He also worked on vehicular networking-based safety application and a software defined radio implementation of the IEEE 802.11p physical layer. He volunteered in the organization of international conferences such as VNC and WONS. Since April 2016 he is a Postdoctoral Research Fellow with the Advanced Networking Systems group in Trento, led by Prof. Renato Lo Cigno.

Christoph Sommer [M] (sommer@ccs-labs.org) is an Assistant Professor (AkadR a.Z.) at Paderborn University, joining the Distributed Embedded Systems Group in 2014. He received his Ph.D. degree in engineering (Dr.-Ing., with distinction) and his M.Sc. degree in computer science (Dipl.-Inf. Univ.) from the University of Erlangen in 2011 and 2006, respectively. In 2010, he was a visiting scholar with the research group of Ozan K. Tonguz at the Electrical and Computer Engineering Department of Carnegie Mellon University (CMU). In 2012, he was a visiting scholar with the research group of Mario Gerla at the Computer Science Department of the University of California, Los Angeles (UCLA). Until 2014, he was a Postdoctoral Research Fellow with the Computer and Communication Systems Group at the University of Innsbruck. Since 2011, he is a member of the ACM/Springer Wireless Networks (WINET) editorial board. Since 2016, he serves as area editor for Elsevier Computer Communications (COMCOM). His research is focused on questions regarding traffic efficiency, safety, and security aspects of Car-to-X communication in heterogeneous environments. He also authored the textbook Vehicular Networking, published in 2014 by Cambridge University Press.

Falko Dressler [F] (dressler@ccs-labs.org) is Full Professor for Computer Science and Chair for Distributed Embedded Systems at the Heinz Nixdorf Institute and the Dept. of Computer Science, Paderborn University, where he is also a member of the University Senate. Before moving to Paderborn, he was a Full Professor at the Institute of Computer Science, University of Innsbruck and an Assistant Professor at the Dept. of Computer Science, University of Erlangen. He received his M.Sc. and Ph.D. degrees from the Dept. of Computer Science, University of Erlangen in 1998 and 2003, respectively. Dr. Dressler is associate editor-in-chief for Elsevier Computer Communications as well as an editor for journals such as IEEE Trans. on Mobile Computing, Elsevier Ad Hoc Networks, and Elsevier Nano Communication Networks. He has been guest editor of special issues in IEEE Journal on Selected Areas in Communications, IEEE Communications Magazine, Elsevier Ad Hoc Networks, and many others. He has been chairing conferences such as IEEE INFOCOM, ACM MobiSys, ACM MobiHoc, IEEE VNC, IEEE GLOBECOM, and many others. He authored the textbooks Self-Organization in Sensor and Actor Networks published by Wiley & Sons and Vehicular Networking published by Cambridge University Press. He has been an IEEE Distinguished Lecturer as well as an ACM Distinguished Speaker. Dr. Dressler is an IEEE Fellow as well as a Senior Member of ACM, and member of GI (German Computer Science Society). He also serves in the IEEE COMSOC Conference Council. His research objectives include adaptive wireless networking, self-organization techniques, and embedded system design with applications in ad hoc and sensor networks, vehicular networks, industrial wireless networks, and nano-networking.