

Bastian Bloessl

A Physical Layer Experimentation Framework for Automotive WLAN

Dissertation

June 2018

Please cite as:

Bastian Bloessl, "A Physical Layer Experimentation Framework for Automotive WLAN," PhD Thesis (Dissertation), Heinz Nixdorf Institute, Paderborn University, Germany, June 2018.



Distributed Embedded Systems (CCS Labs)
Heinz Nixdorf Institute, Paderborn University, Germany

Fürstenallee 11 · 33102 Paderborn · Germany

<http://www.ccs-labs.org/>

A Physical Layer Experimentation Framework for Automotive WLAN

Dissertation
zur Erlangung des Grades

DOKTOR DER NATURWISSENSCHAFTEN

vorgelegt von

Bastian Bloessl

geb. am 11. Juli 1984
in Bamberg, Deutschland

angefertigt in der Fachgruppe

**Distributed Embedded Systems
(CCS Labs)**

**Heinz Nixdorf Institut
Universität Paderborn**

Betreuer: **Prof. Dr.-Ing. habil. Falko Dressler**
Gutachter: **Prof. Dr.-Ing. habil. Falko Dressler**
Prof. Dr.-Ing. Matthias Hollick
Prof. Dr. Renato Lo Cigno

Tag der Abgabe: **27. April 2018**
Tag der Promotion: **25. Juni 2018**

Abstract

Future cars will be equipped with communication modules that allow them to exchange information directly with each other and potentially infrastructure nodes, forming a Vehicular Ad Hoc Network (VANET). Through communication, cars will be able to coordinate and drive cooperatively, which will make transportation safer, more efficient, and more comfortable than ever before. One of the considered technologies for vehicular networks is IEEE 802.11p, a slightly modified version of consumer Wireless LAN (WLAN) that was adapted to better fit the characteristics of vehicular environments. While the decision to rely on readily available technology might ease market introduction, it also raises the question whether a physical layer that was designed for relatively static indoor environments can provide reasonable performance in highly dynamic VANETs.

Using Software Defined Radios (SDRs), i.e., fully programmable radios, we are able to address this question, as they allow us to closely examine and modify the physical waveform. We made SDRs accessible for research on VANETs by implementing the first IEEE 802.11p transceiver for GNU Radio, a popular real-time signal processing framework for use in SDRs. Performing all signal processing on a PC, our transceiver is well-suited for rapid prototyping and can be used for simulations as well as real-world experiments, offering a seamless switch from theory to practice.

In the first part of the thesis, we detail the design of our IEEE 802.11p transceiver, study its computational complexity, and present results from thorough validations through simulations and interoperability tests. We furthermore show that it is possible to support time-critical functionalities like channel access and automatic gain control without giving up the advantages of a PC implementation.

In the second part, we use our transceiver to address selected research questions in VANETs. Here, we conduct field tests to compare the performance of different devices and algorithms in realistic environments and study the impact of noise and intra-technology interference on IEEE 802.11p. Finally, we show a use-case for our transceiver that goes beyond signal processing: With full access to all information down to the physical layer, we develop a novel, robust attack on the location privacy of vehicles and study its impact through network simulations.

Kurzfassung

Zukünftig werden Automobile mit Kommunikationsmodulen ausgestattet sein, die einen direkten Datenaustausch zwischen den Fahrzeugen ermöglichen. Auf diese Weise können sich Verkehrsteilnehmer koordinieren, um so den Straßenverkehr sicherer, effizienter und komfortabler zu gestalten. Eine der Technologien, die dafür in Betracht gezogen wird, ist IEEE 802.11p, eine an Fahrzeugnetze angepasste Version von normalem Wireless LAN (WLAN). Die Entscheidung WLAN, und damit einen bereits vorhandenen Standard, heranzuziehen, ist im Hinblick auf die Markteinführung sicherlich nachvollziehbar. Gleichzeitig stellt sich allerdings die Frage, ob eine Technologie, die für relativ statische Anwendungen entwickelt wurde, den Herausforderungen dynamischer Fahrzeugnetze gerecht werden kann.

Software Defined Radios (SDRs), programmierbare Funksende- und -empfangseinheiten, bieten vollen Zugriff auf alle Aspekte der Kommunikation und sind damit prädestiniert, die Eignung von WLAN zu untersuchen. Um dies zu ermöglichen, haben wir IEEE 802.11p basierend auf GNU Radio implementiert. GNU Radio ist eine SDR-Entwicklungsumgebung, mit der drahtlose Kommunikation prototypisch umgesetzt werden kann. Durch Abbildung des Standards in Software können wir dieselbe Implementierung für Simulationen und Messungen benutzen.

Im ersten Teil der Arbeit gehen wir auf unsere IEEE 802.11p-Implementierung ein. Wir untersuchen die Aufwändigkeit der Berechnungen und zeigen die Korrektheit durch Simulationen und Interoperabilitätstests. Darüber hinaus erweitern wir unsere Implementierung durch zeitkritische Funktionen, wie Kanalzugriff und automatische Anpassung der Empfangsverstärkung, ohne die Komplexität merklich zu erhöhen.

Im zweiten Teil der Arbeit verwenden wir unsere Implementierung, um ausgewählten Forschungsfragen nachzugehen. In diesem Kontext führen wir zwei Feldtests durch, in denen wir die Leistungsfähigkeit des Standards in einer realistischen Umgebung untersuchen. Weiterhin untersuchen wir den Einfluss von Interferenz auf IEEE 802.11p und validieren so ein in vielen Studien genutztes Simulationsmodell. Abschließend stellen wir einen neuen Angriff auf die Privatsphäre in Fahrzeugnetzen vor, der erst durch unsere Implementierung und die Möglichkeit auf alle Daten des Empfangsprozesses zuzugreifen, realisiert werden kann.

Contents

1	Introduction	1
1.1	Automotive WLAN	2
1.2	Contribution	4
2	WLAN for Vehicular Networking	11
2.1	IEEE 802.11p	13
2.2	Medium Access	14
2.3	Orthogonal Frequency-Division Multiplexing	17
2.4	Physical Layer Challenges	19
2.5	Protocol Stacks	22
2.6	Research Methodologies	25
3	Software Defined Radio	27
3.1	Architectures	31
3.2	Frameworks	32
4	An SDR-based WLAN Transceiver	39
4.1	Physical Layer	44
4.2	Verification and Interoperability	56
4.3	Computational Complexity	64
4.4	Time-Critical Functionality	69
4.5	Conclusion	81
5	WLAN for Vehicular Networking	83
5.1	Field Tests	86
5.2	Impact of Intra-Technology Interference	97
5.3	The Scrambler Attack	106
6	Conclusion	117
	Bibliography	131

Chapter 1

Introduction

Today, we witness how autonomous driving is about to change transportation and, with it, large parts of our society. Research and development has overcome fundamental challenges and the first prototypes are allowed on public streets. These proof-of-concept implementations gave the sector a whole new drive and motivated big companies to launch visionary projects. Uber already operates a fleet of self-driving taxis in Pittsburgh¹, while Amazon considers self-driving vans or trucks to transport their goods². Without a doubt, the technology will constitute a huge step towards fully automated business processes as envisioned in the Industry 4.0 context. By leaving the research labs and being tested on the street, the topic also became prevalent in the general public and is covered in the news on a nearly daily basis. It seems that autonomous driving is a technology whose time has finally come.

Taking a more technical perspective, we can view the current generation of autonomous cars as cyber-physical systems with a multitude of sensors attached to them. Using data from stereo cameras, gyroscopes, ultrasonic distance sensors, Global Positioning System (GPS), RADAR, and LIDAR, modern cars are able to sense their immediate environment with a high precision. Based on this data, vehicles are able to maneuver reliably even in challenging environments [1], which raises the hope that the technology will help to drastically reduce the number of accidents in the future – especially since over 90 % of the crashes are accredited to human factors [2]. Yet, the current generation of autonomous vehicles shares a common limitation: They rely on locally available sensor data only. Since these sensors are, furthermore, limited by the line of sight, vehicles have to drive defensively in environments with limited visual range. Such environments are, however, very common as buildings or other cars and trucks can obstruct the field of vision.

¹<https://newsroom.uber.com/us-pennsylvania/new-wheels/>

²<http://uk.businessinsider.com/amazon-is-exploring-self-driving-technology-2017-4>

Future vehicles will be different. They will employ wireless communication to exchange information with each other and potentially infrastructure nodes, forming a vehicular network. The ability to communicate is not a mere add-on but adds a whole new dimension, enabling the transition from *autonomous driving* to *cooperative driving*. If we consider what the Internet was for PCs, we might be able to grasp the potential. Still, communication is not only about autonomous driving but the enabler for a much wider class of Cooperative Intelligent Transportation Systems (C-ITS). Already in 2009, European Telecommunications Standards Institute (ETSI) envisioned a basic set of applications [3] and the list has been growing ever since. Today, people are working, for example, on intersection collision warning systems [4], traffic information systems [5], green light speed advisories [6], and automated car following [7]. Overall, inter-vehicular communication is an important technological advancement that will make driving safer, more efficient, more comfortable, and more entertaining.

Given the wealth and diversity of applications, there is no single communication technology that is well-suited for all use-cases. Vehicular networks will, therefore, be heterogeneous, using access technologies like LTE, Wireless LAN (WLAN), Millimeter Wave (mmW), and Visible Light Communication (VLC). Apart from costs, the technologies mainly differ in terms of throughput, delay, range, and directionality of the communication.

1.1 Automotive WLAN

One of the core technologies for future vehicular networks will be automotive WLAN, a slightly modified version of normal WLAN that operates on a dedicated frequency band at 5.9 GHz and was adapted to better fit the requirements of vehicular networks. Using WLAN technology, cars can communicate directly with each other and infrastructure nodes, forming decentralized networks, usually referred to as Vehicular Ad Hoc Networks (VANETs). The advantages of WLAN make it a perfect candidate for safety and efficiency applications:

low cost It uses cheap, readily available chips that operate free of charge on a dedicated frequency band.

low delay It supports direct communication between vehicles, without the need for intermediate nodes like a base station or an access point.

good coverage It supports communication ranges over 800 m [8] and is not strictly limited to line of sight or blocked by rain.

no directionality In contrast to VLC or mmW, WLAN has no inherent directionality and is, therefore, well-suited for broadcast-based communication to all cars in close vicinity.

With these properties and the fact that safety and efficiency applications are regarded as one of the main drivers for the introduction of C-ITS, it is likely that WLAN will play a central role in future vehicular networks.

While WLAN is a cheap readily-available and time-tested technology, there are also doubts whether it can be fit for operation in vehicular environments. To support VANETs, the IEEE 802.11 standard was extended with a new operational mode that allowed immediate communication without prior connection setup and an adapted Orthogonal Frequency-Division Multiplexing (OFDM)-based physical layer (PHY), which is similar to IEEE 802.11a but with all timings doubled. This change transforms the 20 MHz signal of IEEE 802.11a to the 10 MHz signal proposed for automotive applications. In a nutshell, the switch to 10 MHz made the signal more robust against delay spread but, at the same time, more sensitive to Doppler and time variability of the channel [9]. This means that the change was not a clear improvement. In fact, it rebalanced a trade-off and was, therefore, not without doubts.

The concerns of the research community are reflected by a vast amount of publications that focus on the topic [10]–[13]. As researchers were worried about the reliability of WLAN in highly dynamic vehicular environments, they came up with a wide range of approaches to cope with the expected challenging channel effects. These works proposed, for example, to implement advanced receive algorithms [14], to introduce additional pilot symbols [11], to use differential encoding [15], or to exploit diversity through relaying or multiple antennas [16]. The design of high performance receivers and the evaluation of automotive WLAN in realistic scenarios are still subject to ongoing studies.

A problem faced by many researchers is that the available tools and research methodologies have inherent limitations that cannot easily be overcome. Besides analytical models, researchers rely on simulations and increasingly on measurements and field tests to study VANETs [17]. Simulations are often the first choice as they are easy to conduct and allow investigating new algorithms and strategies in a fast and reproducible manner. However, most works are based on custom PHY implementations in a scripting language like MATLAB [11], [15], [18]. Given the complexity of the PHY, we might be left with questions regarding implementation details or even the general correctness, especially if the source code is not published. Furthermore, it might be unclear whether the assumptions used in simulations are reasonable and hold in practice. Simulations alone, therefore, often lack the ultimate proof of their correctness and the applicability of the proposed approach or algorithm.

Measurements with hardware prototypes can overcome this issue but come with other limitations. Not only can they be expensive and time consuming to conduct, they also act as a black box for the user as the algorithms used in the transceiver are usually considered to be intellectual property of the vendor. This and the fact that the PHY implementation is fixed, limits their application in the academic context, especially with regard to testing new algorithms or adaptations of the technology.

1.2 Contribution

To overcome the limitations of current tools, we developed a complete Software Defined Radio (SDR)-based transceiver for automotive WLAN. SDRs are programmable radios that can be used to send and receive arbitrary waveforms [19], making them the perfect tool to prototype receivers and study novel wireless technologies. The transceiver presented in this thesis is the first implementation of the technology for GNU Radio, a General Purpose Processor (GPP)-based SDR framework, where signal processing is implemented on a normal PC. This architecture allows us to program the radio in a high-level language like C++ or Python, which makes the system well-suited for rapid prototyping [20].

The main benefit of our implementation is that it enables a unique workflow that is not available with other tools or platforms. An overview of this workflow is given in Figure 1.1, where we show the typical cyclic nature of the research process. It starts with an idea of either a completely new PHY or the improvement of an existing technology. Implementing this idea on a GPP-based SDR, we can use the same code

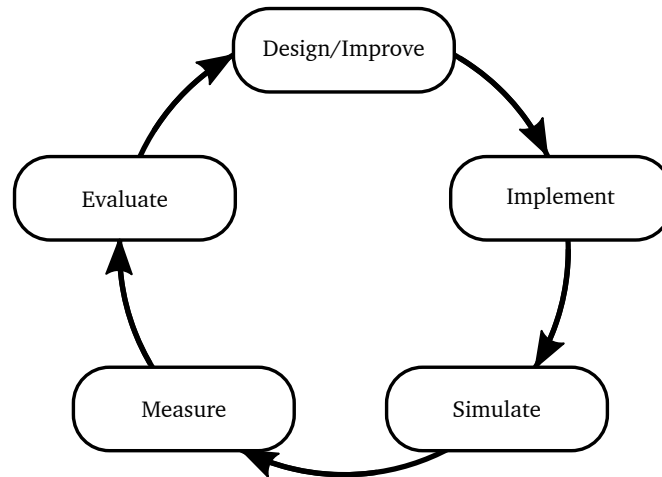


Figure 1.1 – Our SDR transceiver can be used throughout the whole research process, allowing us to overcome the usual disconnect between theory and practice.

base to assess the performance through simulations over various channels as well as to conduct over-the-air measurements in the lab or the real environment.

This would not be possible with other platforms, as there is always a disconnect that cannot be overcome. Normal PHY simulators, for example, are limited to their domain as they cannot be used to conduct over-the-air experiments. Hardware prototypes, in turn, provide limited information about their internals and cannot be used in simulations. For that reason, researchers are either limited to one domain or have a disconnect in their workflow when switching tools. This does not only increase the workload but might also raise concerns as to whether the simulated PHY and the hardware prototype are actually comparable.

With our transceiver, we are able to overcome this problem. Our GPP-based SDR implementation provides a seamless switch between simulations and real-world experiments and, therefore, bridges the gap between theory and practice. It is one tool that can be used throughout the whole process, allowing us to easily go through the cycle multiple times, while continuously improving or adapting our strategies based on insights from simulations and measurements. Ultimately, this tool helps us to get a better understanding of the challenges and the achievable performance of WLAN in a vehicular environment.

While this application was clearly the main driver for the development of our transceiver, its ability to access and if needed modify its components down to and including the PHY, makes it a valuable tool also in other contexts. Apart from performance evaluations of automotive WLAN, we also present a novel robust attack on the location privacy of vehicles that was enabled by our transceiver as it allows accessing identifying information that is hidden deep in the decoding process and, therefore, not accessible with other prototypes.

Overall, the scientific contributions can be summarized as follows:

- We develop an Open Source SDR-based WLAN transceiver, study its computational complexity, and validate it through interoperability tests with other IEEE 802.11p prototypes (Chapter 4).
- We show that it is possible to support time-critical functionalities like channel access and Automatic Gain Control (AGC) without giving up the advantages of a PHY implementation on a normal PC (Section 4.4).

Using our SDR-based transceiver, we address open research questions in VANETs:

- We conduct two field tests to study the performance of WLAN in an automotive environment (Section 5.1). The first compares the performance of our implementation with other prototypes, while the second uses our implementation to compare the performance of different receive algorithms.

- We characterize the impact of noise and interference on IEEE 802.11p to verify a PHY simulation model used by many network simulators (Section 5.2).
- We present a novel attack on the location privacy of vehicles and evaluate its large-scale impact through network simulations (Section 5.3).

To make our work available to the community and to allow reproduction of our results, we release the transceiver under an Open Source license. We are particularly proud that it proved useful to fellow researchers, who used our transceiver in many studies, ranging from algorithm design [21] to backscatter communication [22].

The thesis is based on the following peer-reviewed publications:

- B. Bloessl, M. Segata, C. Sommer, and F. Dressler, “Performance Assessment of IEEE 802.11p with an Open Source SDR-based Prototype,” *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, pp. 1162–1175, May 2018. DOI: 10.1109/TMC.2017.2751474, © 2018 IEEE.
- B. Bloessl, F. Klingler, F. Missbrenner, and C. Sommer, “A Systematic Study on the Impact of Noise and OFDM Interference on IEEE 802.11p,” in *9th IEEE Vehicular Networking Conference (VNC 2017)*, Torino, Italy: IEEE, Nov. 2017, pp. 287–290. DOI: 10.1109/VNC.2017.8275633, © 2017 IEEE.
- 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. DOI: 10.1145/2980100.2980104.
- 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. DOI: 10.1109/INFCOMW.2015.7179325, © 2015 IEEE.
- 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. DOI: 10.1109/ICCNC.2015.7069376, © 2015 IEEE.
- 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. DOI: 10.1145/2643230.2643240.

- 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,” *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 18, no. 3, pp. 81–90, Jul. 2014. DOI: 10.1145/2721896.2721913.
- B. Bloessl, M. Segata, C. Sommer, and F. Dressler, “Towards an Open Source IEEE 802.11p Stack: A Full SDR-based Transceiver in GNU Radio,” in *5th IEEE Vehicular Networking Conference (VNC 2013)*, Boston, MA: IEEE, Dec. 2013, pp. 143–149. DOI: 10.1109/VNC.2013.6737601, © 2013 IEEE.
- B. Bloessl, M. Segata, C. Sommer, and F. Dressler, “A GNU Radio Based Receiver Toolkit for IEEE 802.11a/g/p,” in *19th ACM International Conference on Mobile Computing and Networking (MobiCom 2013), 5th Wireless of the Students, by the Students, for the Students Workshop (S3 2013), Demo Session*, Miami, FL: ACM, Oct. 2013.
- B. Bloessl, M. Segata, C. Sommer, and F. Dressler, “Decoding IEEE 802.11a/g/p OFDM in Software using GNU Radio,” in *19th ACM International Conference on Mobile Computing and Networking (MobiCom 2013), Demo Session*, Miami, FL: ACM, Oct. 2013, pp. 159–161. DOI: 10.1145/2500423.2505300.
- 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. DOI: 10.1145/2491246.2491248, © 2013 ACM.

During my studies, I authored and co-authored further peer-reviewed papers and articles:

- B. Bloessl and F. Dressler, “mSync: Physical Layer Frame Synchronization Without Preamble Symbols,” *IEEE Transactions on Mobile Computing*, 2018, available online. DOI: 10.1109/TMC.2018.2808968.
- M. Nabeel, B. Bloessl, and F. Dressler, “Efficient Receive Diversity in Distributed Sensor Networks using Selective Sample Forwarding,” *IEEE Transactions on Green Communications and Networking*, 2017, to appear. DOI: 10.1109/TGCN.2017.2780196.

- F. Klingler, G. S. Pannu, C. Sommer, B. Bloessl, and F. Dressler, “Field Testing Vehicular Networks using OpenC2X,” in *15th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys 2017), Poster Session*, Niagara Falls, NY: ACM, Jun. 2017, pp. 178–178. DOI: 10.1145/3081333.3089322.
- M. Nabeel, B. Bloessl, and F. Dressler, “Selective Signal Sample Forwarding for Receive Diversity in Energy-Constrained Sensor Networks,” in *IEEE International Conference on Communications (ICC 2017)*, Paris, France: IEEE, May 2017. DOI: 10.1109/ICC.2017.7996320.
- M. Nabeel, B. Bloessl, and F. Dressler, “Low-Complexity Soft-Bit Diversity Combining for Ultra-Low Power Wildlife Monitoring,” in *IEEE Wireless Communications and Networking Conference (WCNC 2017)*, San Francisco, CA: IEEE, Mar. 2017. DOI: 10.1109/WCNC.2017.7925504.
- S. Joerer, B. Bloessl, M. Segata, C. Sommer, R. Lo Cigno, A. Jamalipour, and F. Dressler, “Enabling Situation Awareness at Intersections for IVC Congestion Control Mechanisms,” *IEEE Transactions on Mobile Computing*, vol. 15, no. 7, pp. 1674–1685, Jun. 2016. DOI: 10.1109/TMC.2015.2474370.
- M. Nabeel, B. Bloessl, and F. Dressler, “On Using BOC Modulation in Ultra-Low Power Sensor Networks for Wildlife Tracking,” in *IEEE Wireless Communications and Networking Conference (WCNC 2016)*, Doha, Qatar: IEEE, Apr. 2016, pp. 848–853. DOI: 10.1109/WCNC.2016.7564858.
- M. Segata, B. Bloessl, S. Joerer, C. Sommer, M. Gerla, R. Lo Cigno, and F. Dressler, “Towards Communication Strategies for Platooning: Simulative and Experimental Evaluation,” *IEEE Transactions on Vehicular Technology*, vol. 64, no. 12, pp. 5411–5423, Dec. 2015. DOI: 10.1109/TVT.2015.2489459.
- B. Bloessl and F. Dressler, “mSync - Frames without Preambles,” in *21st ACM International Conference on Mobile Computing and Networking (MobiCom 2015), 4th ACM Software Radio Implementation Forum (SRIF 2015), Poster Session*, Paris, France: ACM, Sep. 2015, pp. 11–11. DOI: 10.1145/2801676.2801678.
- B. Bloessl and F. Dressler, “mSync in Action,” in *21st ACM International Conference on Mobile Computing and Networking (MobiCom 2015), 7th Wireless of the Students, by the Students, for the Students Workshop (S3 2015), Demo Session*, Paris, France: ACM, Sep. 2015, pp. 24–24. DOI: 10.1145/2801694.2801698.
- F. Dressler, B. Bloessl, M. Hierold, C.-Y. Hsieh, T. Nowak, R. Weigel, and A. Koelpin, “Protocol Design for Ultra-Low Power Wake-Up Systems for Tracking Bats in the Wild,” in *IEEE International Conference on Communications (ICC*

- 2015), London, UK: IEEE, Jun. 2015, pp. 6345–6350. DOI: 10.1109/ICC.2015.7249335.
- M. Segata, S. Joerer, B. Bloessl, C. Sommer, F. Dressler, and R. Lo Cigno, “PLEXE: A Platooning Extension for Veins,” in *6th IEEE Vehicular Networking Conference (VNC 2014)*, Paderborn, Germany: IEEE, Dec. 2014, pp. 53–60. DOI: 10.1109/VNC.2014.7013309.
 - M. Segata, B. Bloessl, S. Joerer, C. Sommer, M. Gerla, R. Lo Cigno, and F. Dressler, “Towards Inter-Vehicle Communication Strategies for Platooning Support,” in *7th IFIP/IEEE International Workshop on Communication Technologies for Vehicles (Nets4Cars 2014-Fall)*, Saint-Petersburg, Russia: IEEE, Oct. 2014, pp. 1–6. DOI: 10.1109/Nets4CarsFall.2014.7000903.
 - S. Joerer, B. Bloessl, M. Huber, A. Jamalipour, and F. Dressler, “Assessing the Impact of Inter-Vehicle Communication Protocols on Road Traffic Safety,” in *20th ACM International Conference on Mobile Computing and Networking (MobiCom 2014)*, *6th Wireless of the Students, by the Students, for the Students Workshop (S3 2014)*, Maui, HI: ACM, Sep. 2014, pp. 21–23. DOI: 10.1145/2645884.2645885.
 - S. Joerer, B. Bloessl, M. Huber, A. Jamalipour, and F. Dressler, “Simulating the Impact of Communication Performance on Road Traffic Safety at Intersections,” in *20th ACM International Conference on Mobile Computing and Networking (MobiCom 2014)*, *Demo Session*, Maui, HI: ACM, Sep. 2014, pp. 287–289. DOI: 10.1145/2639108.2641737.
 - S. Joerer, B. Bloessl, M. Segata, C. Sommer, R. Lo Cigno, and F. Dressler, “Fairness Kills Safety: A Comparative Study for Intersection Assistance Applications,” in *25th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2014)*, Washington, D.C.: IEEE, Sep. 2014, pp. 1442–1447. DOI: 10.1109/PIMRC.2014.7136395.
 - M. Segata, B. Bloessl, C. Sommer, and F. Dressler, “Towards Energy Efficient Smart Phone Applications: Energy Models for Offloading Tasks into the Cloud,” in *IEEE International Conference on Communications (ICC 2014)*, Sydney, Australia: IEEE, Jun. 2014, pp. 2394–2399. DOI: 10.1109/ICC.2014.6883681.
 - S. Joerer, M. Segata, B. Bloessl, R. Lo Cigno, C. Sommer, and F. Dressler, “A Vehicular Networking Perspective on Estimating Vehicle Collision Probability at Intersections,” *IEEE Transactions on Vehicular Technology*, vol. 63, no. 4, pp. 1802–1812, May 2014. DOI: 10.1109/TVT.2013.2287343.

- M. Segata, B. Bloessl, S. Joerer, F. Dressler, and R. Lo Cigno, “Supporting Platooning Maneuvers through IVC: An Initial Protocol Analysis for the Join Maneuver,” in *11th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2014)*, Obergurgl, Austria: IEEE, Apr. 2014, pp. 130–137. DOI: 10.1109/WONS.2014.6814733.
- M. Segata, B. Bloessl, S. Joerer, C. Sommer, R. Lo Cigno, and F. Dressler, “Vehicle Shadowing Distribution Depends on Vehicle Type: Results of an Experimental Study,” in *5th IEEE Vehicular Networking Conference (VNC 2013)*, Boston, MA: IEEE, Dec. 2013, pp. 242–245. DOI: 10.1109/VNC.2013.6737623.
- 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, Nov. 2012, pp. 25–32. DOI: 10.1109/VNC.2012.6407441.
- B. Bloessl, S. Joerer, F. Mauroner, and F. Dressler, “Low-Cost Interferer Detection and Classification using TelosB Sensor Motes,” *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, no. 4, pp. 34–37, Oct. 2012. DOI: 10.1145/2436196.2436215.
- B. Bloessl, S. Joerer, F. Mauroner, and F. Dressler, “Low-Cost Interferer Detection and Classification using TelosB Sensor Motes,” in *18th ACM International Conference on Mobile Computing and Networking (MobiCom 2012), Poster Session*, Istanbul, Turkey: ACM, Aug. 2012, pp. 403–405. DOI: 10.1145/2348543.2348595.
- B. Bloessl, S. Joerer, N. Nordin, C. Sommer, and F. Dressler, “SaFIC: A Spectrum Analysis Framework for Interferer Classification in the 2.4 GHz Band,” in *31st IEEE Conference on Computer Communications (INFOCOM 2012), Demo Session*, Orlando, FL: IEEE, Mar. 2012.

Chapter 2

WLAN for Vehicular Networking

2.1	IEEE 802.11p	13
2.2	Medium Access	14
2.3	Orthogonal Frequency-Division Multiplexing	17
2.4	Physical Layer Challenges	19
2.5	Protocol Stacks	22
2.6	Research Methodologies	25

Ever since Marconi's first long-range radio transmissions in the late nineteenth century [57], people were fascinated by wireless communications. The broadcast nature of the channel and the possibility to communicate without a physical connection bears obvious advantages. With interest in wireless, the technology evolved at a fast pace. In only a century, it advanced from the first wireless telegraph transmissions, to broadcast radio, to satellite television, to cellular networks and Wireless LAN (WLAN), providing ubiquitous wireless Internet connectivity.

Today, we are on the verge of the next generation of wireless. Future networks will no longer focus on individuals only. Also, an ever-increasing number of things are designed to be online, connected any time and anywhere by means of wireless communication. Inspired by the vision of an Internet of Things (IoT), more and more devices are integrated into the network. Through communication, the sum becomes more than its parts and connectivity makes them appear *smart*.

Vehicles are a prime example of these *things*. Seeking for smart cities and smart transportation, people are working on technologies that allow future vehicles to exchange information with each other and drive cooperatively. This will make transportation safer, more efficient, and more comfortable than ever before. To make this become a reality, the standardization bodies in Europe, the US, and Japan are working on the corresponding standards. In Europe, ETSI is working on ITS-G5; in the US, IEEE is working on 1609 Wireless Access in Vehicular Environment (WAVE); and in Japan, ARIB is working on STD-T109. While all of these standards are based on WLAN, the technology is not without competition. More recently, 3GPP also turned their attention towards automotive applications and developed Cellular Vehicle-to-Everything (C-V2X) [58] as part of Long-Term Evolution (LTE) Release 14. Even though C-V2X is part of a cellular communication standard, it also supports infrastructure-less operation, allowing direct communication between cars. However, such a system would require shared spectrum that is available to users of all cellular providers. While C-V2X evolves at a fast pace, WLAN is at the moment better understood and the only technology that has dedicated spectrum allocated.

2.1 IEEE 802.11p

Given their unique characteristics, Vehicular Ad Hoc Networks (VANETs) ask for novel application-specific solutions on all layers of the network stack. To make WLAN fit for automotive applications, it requires several adaptations that are colloquially referred to as IEEE 802.11p. In fact, IEEE 802.11p is the amendment that added the last components to IEEE 802.11 so that higher-layer standards can use it as a base for a VANET communication stack. In this thesis, we follow the convention of the

community and use IEEE 802.11p when referring to all components of automotive WLAN. This comprises:

- Outside the Context of a BSS (OCB) mode, a new station mode that allows immediate communication without connection setup (initially introduced in IEEE 802.11p).
- Enhanced Distributed Channel Access (EDCA) for Quality of Service (QoS)-enhanced channel access that allows, for example, to prioritize safety messages (initially introduced in IEEE 802.11e).
- 10 MHz physical layer (PHY) mode with all timings doubled for greater robustness against delay-spread (initially introduced in IEEE 802.11j).
- Operation in the 5.9 GHz band (initially introduced in IEEE 802.11p).

The OCB mode is a straightforward but essential extension. Using a wildcard Basic Service Set Identifier (BSSID), it avoids the need to associate with the network and allows communication in a highly mobile network with short contact times. In contrast to the OCB mode, medium access and the changes to the PHY are more involved and, therefore, described in more detail.

2.2 Medium Access

IEEE 802.11 is well-known for its distributed and decentralized Carrier Sense Multiple Access (CSMA)-based medium access scheme, coordinated by the Distributed Coordination Function (DCF). While the standard also supports centrally coordinated channel access, it is less common and not applicable to VANETs, given the decentralized nature of the network. Using the DCF, stations continuously monitor the channel to determine whether the medium is busy. This carrier sensing is divided into *virtual* and *physical* carrier sensing and the channel is declared busy if either of them senses it busy.

Virtual carrier sensing relies on the *duration* field of overheard frames. The field can be used to reserve the channel for a time slot that extends beyond the duration of the current frame. This allows protecting, for example, a whole burst of frames or dependent transmissions like Acknowledgement Frames (ACKs) from interference. Every station that overhears a frame with a non-zero value in its duration field will sense the channel busy for the time indicated.

Physical carrier sensing, in turn, is further divided into *preamble detection* and *energy detection*. Preamble detection is the most reliable mechanism. It uses the characteristic, easily recognizable pattern of the preamble to detect frames even at low Signal to Noise Ratios (SNRs). Once a frame is detected, the receiver tries

to decode the signal field following the preamble. It is encoded in the most robust Modulation and Coding Scheme (MCS) and contains the length and encoding of the following data. With this information, the receiver can calculate the duration of the frame and sense the channel busy during its transmission (even if the signal fades or if the actual data symbols cannot be decoded). The second variant, energy detection, measures the RF energy on the channel and declares the channel busy if it exceeds a predefined threshold. Following the standard, the medium has to be sensed busy for power levels above -65 dBm for 10 MHz channels and -62 dBm for 20 MHz channels [59]. This method is less sensitive but works even if the preamble was missed or if the channel is occupied by a different technology.

The output of the carrier sensing module is used by the CSMA state machine to decide when a frame can be sent. In every transmission attempt, the CSMA algorithm uses two parameters: a minimum inter-frame space (the DCF Interframe Space (DIFS)) and a pseudorandom backoff period that randomizes channel access to avoid that all stations transmit at the same time, which would lead to collisions. The length of the backoff period is measured in integer multiples of the PHY-specific slot time and chosen uniformly from the interval between zero and the contention window CW. Every frame transmission starts with the minimum contention window $CW_{\min}$ and doubles it for every unsuccessful attempt up to a maximum value $CW_{\max}$.

When a frame is handed over to the WLAN module, it first checks if the channel was idle during the last DIFS. If this is the case, the frame is sent immediately. If not, the node continues to monitor the channel. Once it is idle for the duration of the DIFS, the node starts to count down backoff slots. If the channel turns busy during that time, the current number of backoff slots is kept and the node waits again until the channel is free for the duration of the DIFS before it can continue to count down slots. When the number of backoff slots reaches zero, the node transmits the frame. For unicast transmission, the transmitting station expects an ACK from the destination. If this is not received, the node doubles the congestion window (if it is smaller than $CW_{\max}$) and chooses a new backoff uniformly within the congestion window. Once the maximum number of retries is reached, the frame is dropped and the next frame starts again with the minimum congestion window $CW_{\min}$.

Table 2.1 – Channel access parameters for IEEE 802.11p.

Parameter	Value	Reference
slot time	$13 \mu\text{s}$	[59, Table 17-21]
SIFS	$32 \mu\text{s}$	[59, Table 17-21]
$aCW_{\min}$	15	[59, Table 17-21]

After a frame is sent, the device enters a post-transmit backoff that works similar to the normal backoff procedure. This mechanism ensures that the device does not capture the channel and starts sending packets back-to-back (i.e., sends frames that are spaced only by a DIFS), which could stall other devices. An overview of the PHY-specific timings of IEEE 802.11p can be found in Table 2.1

Using the normal DCF, stations used the same parameters for all frames. To support QoS and to allow prioritizing frames, the standard was extended with EDCA, which introduced four traffic classes that were named according to their initially intended use-cases: *Voice*, *Video*, *Background*, and *Best Effort*. These traffic classes can be parameterized with different parameters for the congestion window and the inter-frame space (which is called Arbitration Inter-Frame Space (AIFS) for EDCA). The AIFS is defined with the Short Interframe Space (SIFS), which is the inter-frame space for dependent transmissions like ACKs, and the slot time. For a given Access Category (AC), it is calculated as $AIFS[AC] = SIFS + AIFSN[AC] \cdot \text{slot time}$.

Using a smaller congestion window and a shorter inter-frame space, safety-related messages can be prioritized, since these parameters ensure that they have to wait shorter on average. If the sum of the AIFS and the maximum backoff period of one AC is shorter than the AIFS of another AC, the higher priority frame is guaranteed to be sent first if the transmitters can reliably sense each other.

Frames of different ACs are put into separate queues that are managed by independent instances of the CSMA algorithm, using parameters specific to the corresponding AC. In addition to normal carrier sensing, these queues are connected virtually, i.e., the channel is sensed busy by all CSMA instances if one queue transmits. Furthermore, if two queues wanted to access the channel at the same time, this *virtual collision* is resolved internally: the higher priority frame is sent, while the other CSMA instances back off.

Apart from ACs, IEEE 802.11e also introduced the concept of Transmission Opportunities (TXOPs). A TXOP can be configured per AC and defines a maximum time that the channel can be occupied once the station is allowed to access the medium. TXOPs were introduced to solve the *rate anomaly* of WLAN [60], which arises since the normal DCF algorithm only ensures fairness in terms of the number of frames that are sent. A station with bad connectivity might, however, have to use a more robust MCS, which results in longer frames that need more time to transmit. Although this station sends the same number of frames, it occupies the channel for a much longer time, which can lead to disproportional degradation of the overall network performance. Using TXOPs, we can establish time-based fairness and overcome the problem. Overall, the Medium Access Control (MAC) provides QoS-enhanced, decentralized channel access, which, for IEEE 802.11p, operates on top of an Orthogonal Frequency-Division Multiplexing (OFDM)-based PHY.

2.3 Orthogonal Frequency-Division Multiplexing

OFDM is a physical layer technology that is adopted by many state-of-the-art wireless communication technologies. Seeking for more throughput and higher bandwidths, the classical single-carrier technologies ran into problems. Considering the Shannon–Hartley theorem, we know that the throughput of a wireless channel is bound by its capacity C , which is, for a given bandwidth B , signal power S , and noise power N , calculated as

$$C = B \log_2 \left(1 + \frac{S}{N} \right). \quad (2.1)$$

Today, sophisticated MCSs allow us to optimize spectral efficiency and get close to the capacity bounds. Still, higher throughput ultimately requires a higher bandwidth, which, for a single carrier system, translates into a higher symbol rate. This approach, however, does not scale well: Once the bandwidth of a single-carrier signal exceeds the coherence bandwidth of the channel, the receiver has to employ complex channel estimation algorithms [61], [62].

Using multiple narrow-band subcarriers instead of a single wide-band carrier, OFDM allows us to overcome this issues. Figure 2.1 shows a simplified schematic of a single-carrier and a corresponding multi-carrier scheme. Both configurations use the same resources in time-frequency domain to transmit the same number of symbols and, therefore, provide a similar spectral efficiency. The main advantage of OFDM is its increased flexibility. It adds a degree of freedom that allows balancing between more carriers and longer symbol time, on the one hand, and less carriers and shorter symbol times, on the other hand. The single-carrier scheme in Figure 2.1,

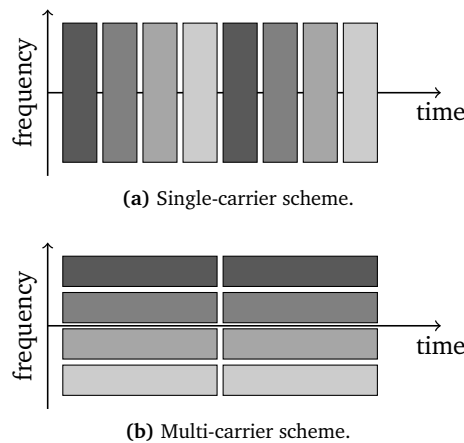


Figure 2.1 – Schematic time-frequency resource usage of a single-carrier and a corresponding multi-carrier scheme.

for example, is converted to a four-carrier scheme, where each OFDM symbol spans four times the original symbol time. It could, however, just as well operate with eight carriers spanning eight times the original symbol time.

This offers flexibility to adapt the PHY to the channel conditions of a particular application. If the coherence time is short, we can use shorter symbols; if the coherence bandwidth is small, we can use more narrow subcarriers. Usually, the system is set up so that the OFDM symbol time is smaller than the coherence time, and the subcarrier bandwidth is smaller than the coherence bandwidth. If these requirements are met, we can use a one-tap equalizer per subcarrier, which greatly simplifies the receiver design [61].

Another advantage of OFDM is its scalability towards higher bandwidths, which, in essence, means adding more subcarriers. WLAN and LTE are great examples, as both technologies support multiple channel bandwidths that only differ in the number of subcarriers. The main trick of OFDM and the reason for its high spectral efficiency is the subcarrier spacing. If earlier single-carrier PHYs shared the spectrum, they required a rather large guard band that assured that the signal of one carrier did not leak into the other. This degradation of the overall spectral efficiency can be overcome with OFDM. Using OFDM, the spectra of the individual subcarriers overlap and, therefore, interfere with each other. The important point is, however, that they do not interfere at the center frequencies of other subcarriers. This is often visualized like in Figure 2.2, where we plot the spectrum of adjacent subcarriers. As shown in the figure, the contributions of each subcarrier are zero at multiples of the subcarrier spacing, i.e., the center frequencies of adjacent subcarriers.

Fortunately, an OFDM signal like this is straightforward to generate through a Discrete Fourier Transformation [61]. Often the number of subcarriers is set to a power of two and the efficient Fast Fourier Transform (FFT) algorithm is used. IEEE 802.11a/g/p, for example, uses 64 subcarriers. Overall, the benefits of OFDM have led to a wide adaption of the technology in many state-of-the-art wireless com-

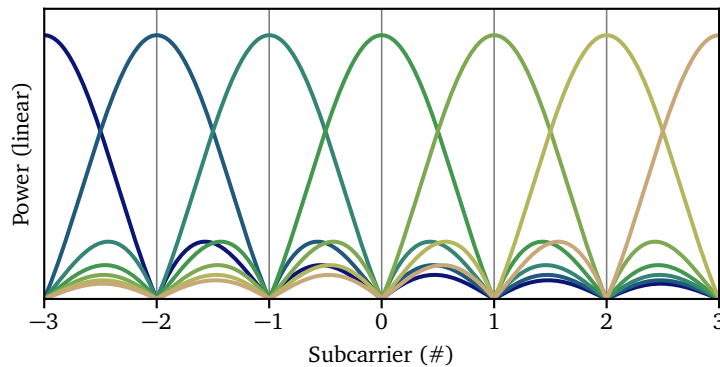


Figure 2.2 – Spectra of adjacent OFDM subcarriers.

munication standards like WLAN, LTE, digital audio broadcast, and both terrestrial and satellite TV.

However, also OFDM comes with drawbacks. One is the need for a guard period between successive OFDM symbols. The duration of the guard period is chosen with regard to the delay-spread of the channel. It ensures that an OFDM symbol does not leak into the useful symbol time of the successive symbol, which would introduce inter-symbol interference and degrade performance. Typically, a *cyclic prefix*, i.e., a cyclic extension of the FFT, is used to fill the guard time. For IEEE 802.11a/g/p, the OFDM symbol is extended by as much as one fourth of the symbol time.

Another drawback of OFDM is its high Peak-to-Average Power Ratio (PAPR). For single-carrier systems, the envelope of the signal is rather constant, limited by the amplitude differences of the constellation points. With OFDM, the contributions of the subcarriers occasionally add up, leading to peaks that can easily drive the power amplifier into saturation. To cope with this, the transmitter either has to use a low average power (to leave enough headroom) or accept nonlinearities. In any case, OFDM requires capable amplifiers that support a large linear range. This can, however, be a problem in practice, especially for highly integrated mobile devices. For that reason, LTE uses SC-OFDM, a variant with a more constant envelope, in the uplink.

2.4 Physical Layer Challenges

Designing a PHY for vehicular networks is a challenging task, since it has to perform in diverse environments and cope with high dynamics and severe multi-path effects [13], [63], [64]. In particular, the mobility leads to fast-fading effects that the receiver has to compensate for. The PHY of IEEE 802.11p is based on IEEE 802.11a but with all timings doubled, transforming the 20 MHz channels of IEEE 802.11a into the 10 MHz channels of IEEE 802.11p. Figure 2.3 compares the two modes in time-frequency domain and shows how doubling the timings stretches the frame in time domain and shrinks it in frequency domain. The area in time-frequency domain and, therefore, its spectral efficiency in terms of bits per Hertz per second remains constant. Since also the cyclic prefix (i.e., the guard time between symbols) is doubled, an IEEE 802.11p frame is more robust against delay spread, as reflected paths do not leak into successive symbols so easily. Furthermore, the 64 subcarriers are spread over a smaller bandwidth and, therefore, become more narrow, which makes the signal better suited for channels with small coherence bandwidths. Given the inverse relationship between the delay spread and the coherence bandwidth, this is just a different view on the same aspect.

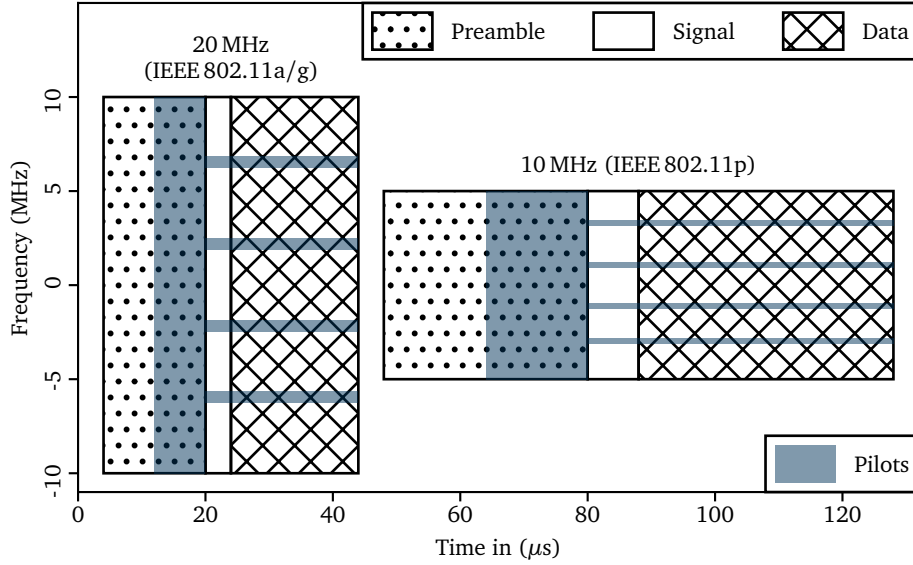


Figure 2.3 – Comparison of an IEEE 802.11a/g frame (left) and a corresponding IEEE 802.11p frame (right) in time-frequency domain.

These advantages, however, come at a price: The main drawback of 10 MHz channels is the increased frame duration, which makes the signal more sensitive against fast-fading effects. In VANETs, these effects are particularly pronounced given the small wavelength (≈ 5 cm at 5.9 GHz) and the high relative velocities. To make things worse, it becomes even more problematic in combination with the pilot pattern of IEEE 802.11a/g/p (i.e., the blue shaded area in Figure 2.3). Typically, channel estimation relies heavily on the initial estimate, which is calculated based on the block pilots at the beginning of the frame. The problem with IEEE 802.11p is that these pilots might become outdated during the reception of the frame, potentially degrading the performance of the PHY.

The relation between the relative vehicle speed, the coherence time of the channel, and the packet duration is shown in Figure 2.4. To calculate the coherence time, we consider a uniform scattering environment with an approximate decorrelation distance of 0.4 times the wave length λ [61]. This approximation is well established and used, for example, to estimate the minimum distance between the antennas of a multi-antenna transceiver. For a given relative vehicle speed v , we can calculate the coherence time τ as

$$\tau \approx 0.4 \frac{\lambda}{v} = 0.4 \frac{c}{f \cdot v}, \quad (2.2)$$

where c is the speed of light and f the carrier frequency, which we set to 5.9 GHz. The frame duration corresponds to QPSK- $\frac{1}{2}$ frames and the packet size refers to the

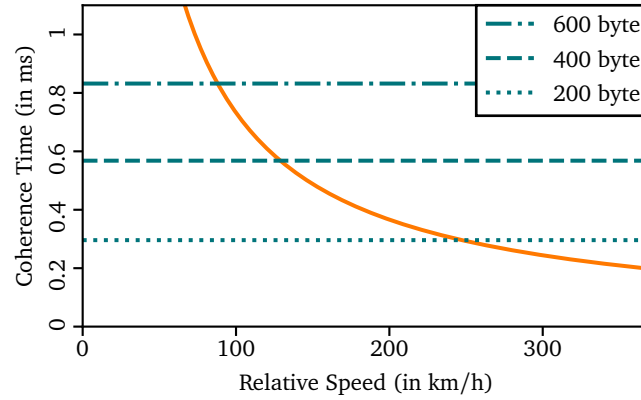


Figure 2.4 – Comparison of the channel's coherence time (solid, orange line) and the duration of QPSK- $\frac{1}{2}$ frames of different sizes (horizontal lines).

MAC Protocol Data Unit (MPDU), which comprises all data from the MAC, including the MAC header, the payload, and the Cyclic Redundancy Check (CRC). Using the coherence time, we can approximate how long the initial channel estimate is valid. As we can see from the plot, already a 600 Byte frame exceeds the coherence time for relative vehicles speeds of 100 km/h. Even for frames as short as 200 Byte, the duration is longer than the coherence time for relative speeds above 250 km/h. These results suggest that commonly used algorithms that rely only on the initial estimate of the channel might run into problems when used for vehicular applications.

A different view on the problem is presented in Figure 2.5. It shows the relationship between the payload size and the number of OFDM symbols for all MCSs of IEEE 802.11p. A second x-axis maps the size to the duration of the frame. To relate

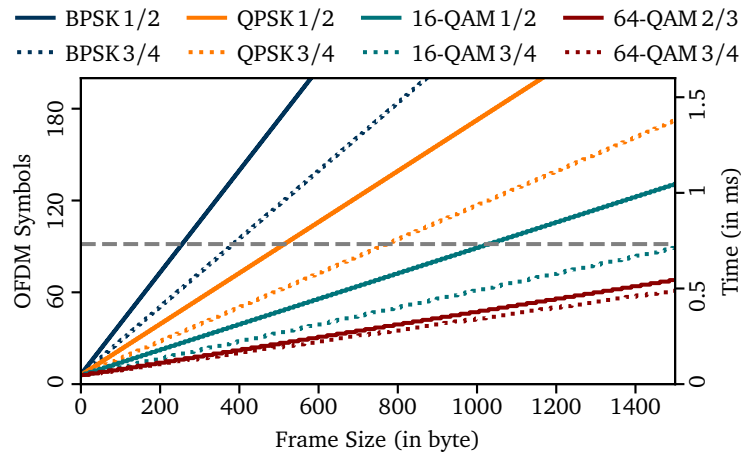


Figure 2.5 – Frame duration depending on the payload size for all MCSs. The dashed horizontal line corresponds to the coherence time of the channel for a relative speed of 100 km/h.

the duration with the dynamics of the VANET channel, we added a dashed horizontal line that corresponds to the coherence time at a relative speed of 100 km/h. Such a relative speed is typical for oncoming traffic in urban environments.

From the graph, we can see that already a relatively short Binary Phase-Shift Keying (BPSK)- $\frac{1}{2}$ frame with a size of less than 250 Byte is longer than the coherence time of the channel. This means that already for short frames, the initial channel estimate becomes outdated during the frame. Also for 16-Quadrature Amplitude Modulation (QAM)- $\frac{1}{2}$ frames, this already happens at a payload size of approximately 1000 Byte. The results suggest that a simple receiver that relies only on the initial channel estimate will run into problems when the frame duration gets close to (or exceeds) the coherence time. This raises the question whether a PHY that was designed for a relatively static indoor environment can cope with the dynamics of VANET. These observations motivated many studies that proposed novel channel estimation schemes for IEEE 802.11p [65], [66] or even recommended to adapt the PHY [11], [15]. Until today, the performance evaluation of the technology in realistic environments and the design of a receiver that provides a good compromise between performance and complexity are subject to ongoing studies. At this stage of the development process, it is, however, unlikely that the PHY and the MAC will undergo significant changes so that at least the first generation of VANETs will probably be based on IEEE 802.11p. To bring the technology on the road, we need a full protocol stack that integrates IEEE 802.11p with higher-layer protocols.

2.5 Protocol Stacks

Such VANET communication standards are currently developed in Japan, Europe, and the US. All of them are based on WLAN and mainly differ with regard to the frequency band and the channel access scheme.

IEEE 1609 WAVE and ETSI ITS-G5 use 10 MHz channels in the 5.9 GHz band, dedicated to vehicular communication. Given the spectrum scarcity, the allocation of dedicated channels shows the strong commitment of the regulatory bodies to support the technology and foster its deployment. Both standards define a designated Control Channel (CCH) and use the others as Service Channels (SCHs) or reserve them for emergency vehicles. They are based on IEEE 802.11p, using its 10 MHz OFDM PHY, OCB mode, and EDCA queues. The main difference is the implementation of the higher layers, in particular, the channel access scheme.

Coordinating channel access for vehicular applications is a challenging task as the network has to perform in different environments, ranging from rural areas with low vehicle densities to traffic jams with high network load. Especially in dense environments, it is crucial that devices adapt to the network load to get safety-

relevant messages reliably disseminated. The main issue for VANETs is that the congestion avoidance mechanism of normal WLAN is based on ACKs. If a transmitter does not receive an ACK, it considers the frame to be dropped due to interference (i.e., network congestion) and increases its congestion window. This does not work well for VANETs, as they are mainly broadcasting awareness messages to all cars in their vicinity. Since broadcasts are not acknowledged, they always use the minimal congestion window, which might overload the channel. Protocol stacks, therefore, complement the WLAN MAC with higher-layer channel access schemes.

WAVE is defined in the IEEE 1609 family of standards [67]. In contrast to ETSI ITS-G5, IEEE 1609 WAVE supports multi-channel operation with a single radio [68]. To avoid missing potentially safety-relevant information on the CCH, the radios have to be synchronized and use a split phase approach, where all radios periodically tune to the CCH. The rest of the time can be spent on an arbitrary SCH. While this allows using SCHs with a single radio, it also comes with drawbacks: Periodic channel switching requires tight synchronization and the introduction of guard intervals, which lowers the overall efficiency of the system. Furthermore, channel switching leads to higher collision probability at the start of a phase, since chances are higher that multiple nodes have frames to send [69].

ETSI ITS-G5 does not face these problems since it requires that its nodes always listen on the CCH. However, it also means that a second radio is required if a node wants to use SCHs. To avoid network congestion, ETSI ITS-G5 puts special focus on adaptive channel access that works well with changing node densities. It introduces the Decentralized Congestion Control (DCC) [70], which acts as a higher-layer state machine that regulates the traffic generated by a node. Based on the observed channel load, DCC switches between states that adapt the

- message generation interval,
- transmit power,
- MCS, and
- channel sense threshold.

This higher-layer mechanism works on top of the normal CSMA functionality of WLAN. While it enables broadcast-based networks to operate in dense environments, it also introduces new parameters that are not straightforward to configure. With a suboptimal parameter set, DCC does not necessarily stabilize but has been observed to oscillate between states [69].

With ARIB STD-T109 [71], Japan took a slightly different approach. The standard also uses the IEEE 802.11p PHY as a base but operates on a single channel at 760 MHz. Using a lower frequency band changes the propagation characteristics,

especially since the signal tends to better penetrate objects [72]. This leads to larger communication distances but also increases the interference range. Furthermore, ARIB STD-T109 limits the allowed modulations to BPSK, QPSK, and 16-QAM, i.e., it excludes the 64-QAM scheme of IEEE 802.11p.

Channel access is coordinated with a combination of Time-Division Multiple Access (TDMA) and CSMA. The TDMA scheme uses a 100 ms control interval that is divided into 16 slots. In contrast to ETSI ITS-G5 and IEEE 1609 WAVE, ARIB STD-T109 differentiates between transmissions from Roadside Units (RSUs) and mobile stations. Each TDMA slot is, therefore, subdivided into a period for RSUs and one for mobile nodes. Channel access during the RSU period follows a configured schedule, negotiated between the RSUs. Since RSUs have communication periods for exclusive channel access allocated, they do not need to sense the channel but can send frames directly and quasi back-to-back, spaced by only a SIFS.

The remaining slot time is allocated to the mobile stations, which use normal CSMA to compete for the channel. To incorporate the TDMA scheme into IEEE 802.11, the mobile nodes use the *virtual* carrier sensing function to declare the channel busy during the TDMA periods that are allocated to the RSUs.

The main task of all three protocol stacks is to provide the base for automotive applications. Today, vehicular networks are often mentioned in the context of autonomous driving. Here, cooperation and coordinated maneuvers could improve the efficiency, since vehicles would no longer be limited to their local sensors only. Apart from autonomous driving, there are many more applications that can roughly be categorized into safety, efficiency, comfort, and entertainment applications. Some applications, like rear-end collisions warning or intersection collision avoidance, match very well into one category (safety in this case). Others, like platooning, i.e., vehicles driving close behind each other, forming a *road train*, improve safety, efficiency, and comfort. Further applications that were already part of the basic set of application envisioned by ETSI include traffic information systems, emergency vehicle warning, and roadwork warning [3].

The concrete implementation of these applications is not part of the standard but subject to ongoing research. The standards only incorporate services that can serve as a base. ETSI ITS-G5, for example, specifies Cooperative Awareness Messages (CAMs) that are sent periodically by vehicles to make surrounding vehicles aware of their presence. They include information regarding heading, speed, and vehicle dimensions [73]. These messages are relevant for many safety applications including intersection collision warning systems. Decentralized Environmental Notification Messages (DENMs) are a second type of messages that are not sent periodically but when predefined events occur [74]. Such events are, for example, emergency braking, road hazards, or wrong way driving.

2.6 Research Methodologies

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 real-world 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. Depending on the aspect that is studied, different types of simulators are in use. Dedicated PHY simulators are tuned towards studying channel effects, interference, and signal processing algorithms. Here, researchers rely often on custom simulation models implemented in scripting languages like MATLAB [11], [15], [18]. While the level of detail can be the same as with a Software Defined Radio (SDR), these simulators do not support real-time operation. Furthermore, the implementations are often not tested (or cannot be tested) against real hardware, which can lead to doubts about their correctness, especially if the implementation is not published.

Apart from PHY simulators, discrete event simulations are a popular tool. They use a higher level of abstraction but allow investigating macro-scale network effects. To capture unique characteristics of vehicular networks and to produce realistic results, researchers couple road traffic simulators with network simulators bidirectionally. Well-known examples for such simulators are Veins, iTetris, and VSimRTI [75]. Focusing on larger scenarios, they lend themselves well for investigating the MAC and application layer. For performance reasons, these simulators usually employ simple channel and bit error rate models. There is, however, also the idea to model the PHY on signal level [76] and, thus, to combine physical layer and network layer simulators. In fact, our SDR implementation of IEEE 802.11p has already been integrated into Veins by an independent group [77].

Finally, trace-based studies offer an approach to realistic VANET simulations. Such a study was presented in [12], where the authors recorded raw signal samples in a field test, which were later used for offline, trace-driven simulations. We adopted this approach in one of our field tests (see Section 5.1.2). 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, 10^6 complex baseband 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.

Apart from simulations, real-world 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 [78]. Often a practical realization of a system make engineers aware of problems that might be faced in reality.

Today, many experiments are conducted with dedicated IEEE 802.11p prototypes from companies like Cohda Wireless [79], NEC [80], [81], and Denso [8], [82]. These devices provide complete communication stacks for IEEE 1609 WAVE or ETSI ITS-G5 and are, therefore, well-suited for testing VANET applications. The Cohda Wireless MK5 series, for example, comes with a complete implementation of the ETSI ITS-G5 stack, allowing us to focus on the implementation of the actual application. These devices are, however, rather expensive since they are no mass market products yet.

Apart from dedicated prototypes, it is possible to modify certain off-the-shelf WLAN cards to operate in IEEE 802.11p mode. The Unex DCMA-86P2 card, for example, is a MiniPCI WLAN card that was already used in many experiments [83]–[86]. This card is based on an Atheros chip that is supported by the *ath5k* Linux driver. To enable IEEE 802.11p operation, it is possible to adapt the Linux kernel driver to switch to 10 MHz mode and remove regulatory restrictions to tune to the Cooperative Intelligent Transportation System (C-ITS) channels in the 5.9 GHz band. Recently, the Linux kernel added IEEE 802.11p support for the Atheros *ath9k* driver. Together with Open Source implementations of VANET communication stacks, like *OpenC2X* [87], 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 limited access to the PHY (i.e., they act as a black box) and have fixed PHY implementations that cannot be adapted. These limitations can be overcome with 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, therefore, provide access to all data, allowing for a better understanding of the system.

Chapter 3

Software Defined Radio

3.1	Architectures	31
3.2	Frameworks	32
3.2.1	GNU Radio	33
3.2.2	Microsoft Sora	36

In the past, radios have been built with a clearly defined purpose. They were designed from the ground up to support a specific technology, i.e., to operate on predefined frequency bands and given Modulation and Coding Schemes (MCSs). Most of the time, there was no flexibility and a change of the technology also required changing the hardware. Software Defined Radios (SDRs), i.e., freely programmable radios, overcome this limitation [19]. They allow programming the whole communication stack down to and including the physical layer (PHY) in software, which adds flexibility and opens many opportunities for research and development. It is a simple concept that is about to revolutionize wireless. Using SDR, it is easily possible to demonstrate the feasibility of novel ideas through prototype implementations and provide the ultimate proof-of-concept. Especially for research on wireless networks, the ability to experiment with the technology should not be underestimated. Since we cannot directly perceive electromagnetic waves, we often lack an intuitive understanding of how waves propagate and how signals get distorted on their way to the receiver. With SDR, the spectrum becomes accessible, or as Thomas Rondeau puts it: “SDRs allow us to alter one of the fundamental forces of nature.” This allows researchers to experiment with wireless transceivers and gain a much better understanding of the technology.

The idea of programmable radios was first explored by Mitola [88], [89]. The basic concept is very simple. An ideal *Software Radio*, as shown in Figure 3.1, would directly connect Analog-to-Digital Converters (ADCs) and Digital-to-Analog Converters (DACs) to the antenna. To transmit an arbitrary waveform, the PC streams the digitized, sampled waveform to the DAC, which transforms it into an analog RF signal that is sent out through the antenna. This concept is very similar to a PC sound card, which does the same to generate acoustic waves. The ideal software radio, however, is impractical: Since modern wireless technologies use a rather high carrier frequency, a software radio would have to sample the signal at a very high rate. Already a 2.4 GHz signal, would result in 4.8E9 samples per second. Such high sample rates would not only require very capable ADCs but also drastically overload a typical PC.

Software *Defined* Radios solve this problem by down-converting the signal to center it around zero Hertz. Most SDRs nowadays use direct-conversion transceivers

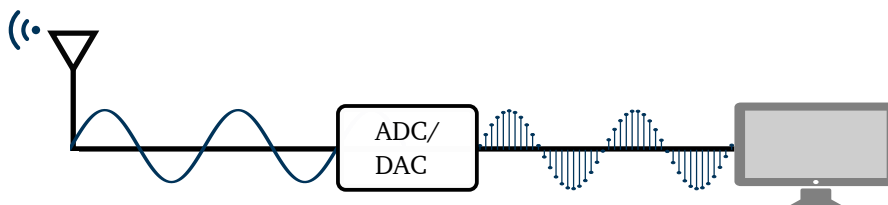


Figure 3.1 – Schematic overview of an ideal Software Radio.

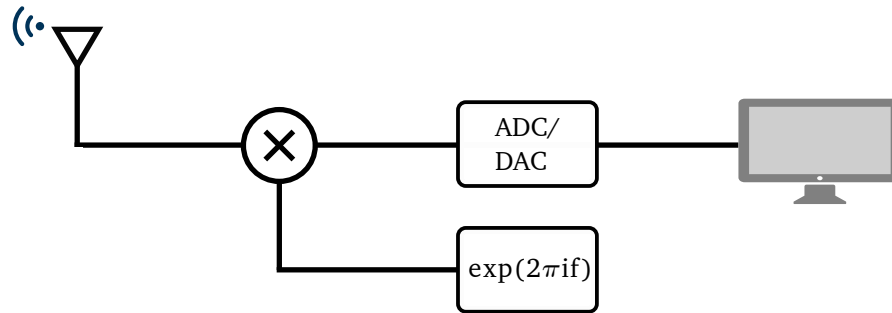


Figure 3.2 – Schematic overview of a direct-conversion Software Defined Radio that up- or down-converts the complex baseband signal to a carrier frequency f .

that operate on the complex representation of the baseband signal. As shown in Figure 3.2, such a transceiver mixes the signal with the carrier frequency to up- or down-convert it to baseband. Using this architecture, the in-phase and quadrature component of a 10 MHz signal is sampled with 10 Msps, independent of the carrier frequency. Given the lower sample rate, such a system is more practical, allowing us to implement wireless protocols in software. One of the biggest advantages of such an implementation is that we can hook into every component of the transceiver and analyze the data to gain a better understanding of the processes. This is in contrast to off-the-shelf devices, where signal processing is implemented on Application-Specific Integrated Circuits (ASICs), which, typically, act as a black box in two ways: We do not know what algorithms are implemented, and we cannot tap into their internal data. The ability to use these devices for research is, therefore, considerably limited. With SDR, we have access to all data and every detail of the PHY.

Using SDR, however, does not only help in understanding the internals of wireless transceivers. The information from the PHY can also be used to implement novel improvements and optimizations. Higher layers could, for instance, exploit the data from the PHY to better adapt to current channel conditions. Even though this breaks with the traditional layered design approach of network stacks, it can improve overall performance, especially in wireless networks [90]. The information from the PHY can also be used to extend the receiver with additional functionality. One could, for example, use an estimate of the channel or the time of arrival for localization [91], synchronized transmissions [92], or beamforming [93].

Apart from studying existing technologies, SDRs are also well-suited to experiment with novel ideas. This can be either alternate transceiver implementations (e.g., novel signal processing algorithms) or variations of the technology (e.g., alternate pilot patterns or coding schemes). In the past, it was only possible to test these ideas in simulations, which often lacked the definite proof of concept. Here, SDRs provide the required flexibility to test these ideas in practice. In this thesis, we will, for example, use our IEEE 802.11a/g/p transceiver to compare alternate

receive algorithms and study their performance in different environments. Apart from modifications of existing technologies, SDRs also allow experimenting with completely novel technologies. The ability to prototype the transceiver in software is a big advantage, which is particularly well-suited to evaluate application-specific technologies before designing an integrated transceiver. We followed this approach when designing an ultra-low power sensor mote for wildlife monitoring [39].

At least in the academic context, SDRs are often regarded as research platforms only. They are, however, not limited to this use-case. Today, consumer hardware like Wireless LAN (WLAN) or Long-Term Evolution (LTE) chips are flexible enough to add new features through firmware updates. Android phones, for example, load a baseband firmware for the LTE chip during boot, and the firmware of typical WLAN cards offers a level of flexibility that was high enough for the European Commission to consider regulating it. The Nexmon project, for example, shows that firmware modifications can be used to turn a WLAN card into a jammer [94] or offload applications onto the WLAN chip [95]. When classifying these devices, it might not be obvious where to draw the line between a normal radio and an SDR. For more professional hardware, like cellular base stations, things are clearer. Already in 2007, Vanu presented a software-defined base station that ran on off-the-shelf hardware and supported both Global System for Mobile Communications (GSM) and Universal Mobile Telecommunications System (UMTS) [96]. Today, it is possible to add support for new technologies (like NB-IoT, the Internet of Things (IoT) mode of LTE) through updates.³ Similarly, we could use SDR to future-proof wireless transceivers in cars. Given their long product-cycle, this would be an interesting option.

3.1 Architectures

When we look at SDRs from a more technical perspective, we can consider them as a bundle of software and hardware components. Depending on the component that implements the PHY, we can classify SDRs into different architectures, ranging from FPGA, over Digital Signal Processors (DSPs), to General Purpose Processor (GPP)-based architectures [20]. Differentiating based on the PHY implementation is reasonable since every SDR uses an FPGA at an early stage in the signal processing chain. Directly after the ADC, the FPGA is used to channelize (i.e., filter and resample) the signal. The discriminating factor is, therefore, how this channelized sample stream is processed.

Wireless Open-Access Research Platform (WARP) [97] is a prominent example of an FPGA-based SDR. The biggest advantage of this type of platform is that it

³<https://www.telekom.com/en/company/details/world-s-first-countrywide-rollout-of-narrowband-iot-494754>

can sustain high bandwidths, has deterministic timings, and introduces minimal latencies. These properties allow implementing complete communication stacks of state-of-the-art wireless standards. A prominent example is the WLAN reference design of WARP, which manages to meet the challenging timing requirements in the microsecond scale. The advantages, however, come at a price, as FPGA-based architectures are more expensive and harder to program. Today, better tooling improves the situation, but FPGA implementations are still less accessible and have longer development times. Furthermore, a design is always tied to a particular platform and cannot easily be ported.

On the other end of the spectrum are GPP-based architectures, where signal processing is implemented on a normal PC. This architecture is well-suited for rapid prototyping as the PHY is implemented in a high-level programming language like C, C++, or Python. Compared to an FPGA-based SDR, it is much easier to get started on this type of platform. After Mitola publicized the idea of an SDR, Vanu Bose was the first to prototype and evaluate a GPP-based real-time signal processing framework [98]. His seminal work, conducted in the context of MIT's *SpectrumWare* project, also laid the foundation of GNU Radio, the most popular real-time signal processing framework, which is heavily used in industry and academia. While rapid prototyping and accessibility are great advantages, GPP-based SDRs also have drawbacks: Even though modern multi-core CPUs allow us to parallelize signal processing tasks and support vectorized instructions through extensions like MME, SSE, or AVX; they still do not reach the computational performance of FPGAs. In addition, buffering of the samples and signal processing on a non-real-time operating system introduces jitter due to the non-deterministic scheduling of the processes. Furthermore, the transport of the samples between the device and the PC introduces delay, which is typically in the low microsecond scale [99]–[101]. The ability to meet the timings of communication standards is, therefore, limited.

To combine the advantages of both approaches, the current trend goes away from pure FPGA-based or GPP-based architectures. The idea is to make accelerators like FPGAs [102] or Graphics Processing Units (GPUs) [103] accessible and offload individual signal processing steps. This approach allows us to gradually move the bottleneck of GPP implementations closer to the hardware.

3.2 Frameworks

SDR frameworks constitute the software part of an SDR. They consist of two main parts: a runtime environment that handles the data flow and a library with basic signal processing algorithms. The fact that the runtime is of great importance becomes clear when we compare an SDR framework with a normal signal processing

library like the ones that come as part of Octave, MATLAB, or SciPy. These libraries are designed for iterative offline signal processing and, therefore, not well-suited for SDRs. Dedicated SDR frameworks like GNU Radio [104], [105], Microsoft Sora [106], RedHawk [107], or Iris [108], in turn, are designed to process a continuous sample stream in real-time. Note that in this context, we refer to the term *real-time* 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. The term does not imply any limits for the delay or the jitter of the system. This is in contrast to real-time as it is commonly used in the Computer Science context when soft or hard deadlines have to be met.

Iris and Sora are more specialized frameworks. Iris initially focused on cognitive radio applications, supporting experiments with opportunistic channel access, self-organization, rendezvous protocols, co-existence, and reconfiguration. Sora, in turn, targeted communication standards like WLAN and later LTE. GNU Radio and RedHawk, in turn, are application-independent, general-purpose frameworks. They do not target a particular standard or application but focus on the runtime environment and a library of basic signal processing tasks. RedHawk is special in that it provides a complete system framework. As the only framework, it can not only be used to create but also to provision and execute distributed signal processing systems. While these frameworks are well-known in academia and industry, GNU Radio is, by far, the most active and popular project. For that reason and since the work presented in this thesis relies on GNU Radio, we discuss it in more detail.

3.2.1 GNU Radio

GNU Radio is an Open Source signal processing framework that is typically used as the software part of a GPP-based SDR. It was started by Eric Blossom [104], based on *PSPECTRA*, an SDR framework developed in the context of MIT's *SpectrumWare* project [98]. Given the interest in SDR, it evolved into a whole ecosystem under the lead of Thomas Rondeau [105]. Today, GNU Radio is the most popular SDR framework that is used in many sectors, including academia, industry, amateur radio, government, and military. With GNU Radio, signal processing is implemented on a GPP like a normal PC. As discussed in Section 3.1, this architecture, on the one hand, introduces higher latency and jitter than FPGA frameworks but, on the other hand, lends itself well for rapid prototyping. This is supported through its use of high-level programming languages. For performance reasons, the core of GNU Radio is implemented in C++. It can, however, also be extended with Python, allowing for even faster development cycles.

A major advantage of GNU Radio, especially when compared to other SDR frameworks, is that it is hardware agnostic, i.e., it is not developed to be used with a specific radio front end. For historic reasons, the devices from Ettus Research are well supported, as the company was working closely with the GNU Radio community. Today, there are, however, also more cost-effective alternatives available. Devices like the BladeRF from Nuand, the HackRF One from Great Scott Gadgets, and the LimeSDR from Myriad-RF can interface with GNU Radio and support the bandwidth and frequency bands for WLAN.

Also GNU Radio itself is not tuned towards a particular application. Instead, it provides a solid base that can be used to realize any technology. GNU Radio, therefore, does not come with extensive implementations for wireless standards. In fact, an exemplary implementation for digital TV is the only technology specific code. Being a general-purpose framework, GNU Radio already served as the base for research in many areas, including satellite communications, cognitive radio, cooperative diversity, multi-antenna systems, localization, and radio astronomy. Application-specific functionality for these use-cases is implemented in so-called Out-Of-Tree (OOT) modules, which extend GNU Radio with custom blocks. Our IEEE 802.11p transceiver is one example of such an OOT module.

The central component of GNU Radio is a *block*. Typically, a block implements a specific signal processing task like a filter, an FFT, a modulator, or a synchronization algorithm. To create a transceiver, individual blocks are connected to form a *flow graph*, a data structure that describes more complex, higher order functionality through a combination of blocks. Initially, GNU Radio was focusing on stream-based data flows. This paradigm is natural for SDRs, as the radio front end provides a continuous stream of complex baseband samples. Since stream-based connections fall short when implementing packet-based transceivers, GNU Radio was extended with asynchronous message passing. These messages are implemented with a polymorphic data type so that blocks can exchange any information. Especially, frame-based operations like the calculation of a checksum, encoding, and scrambling benefit from message passing. Apart from introducing the notion of a frame, this also allows blocks to exchange out of band control information. A level controller, for example, could readjust the gain of the RF front end if the signal does not fully utilize the dynamic range of the ADCs. In our IEEE 802.11a/g/p transceiver, we make heavy use of both stream- and message-based data exchange between blocks.

To ease the creation of transceivers, GNU Radio comes with GNU Radio Companion (GRC), a graphical user interface to set up, configure, and run flow graphs. A screenshot of GRC is shown in Figure 3.3. It shows the main canvas where the flow graph is set up (left), the list of available blocks (right), and a console with status information (bottom). GNU Radio also comes with GUI elements to change parameters while the transceiver is running and graphical outputs that, for example,

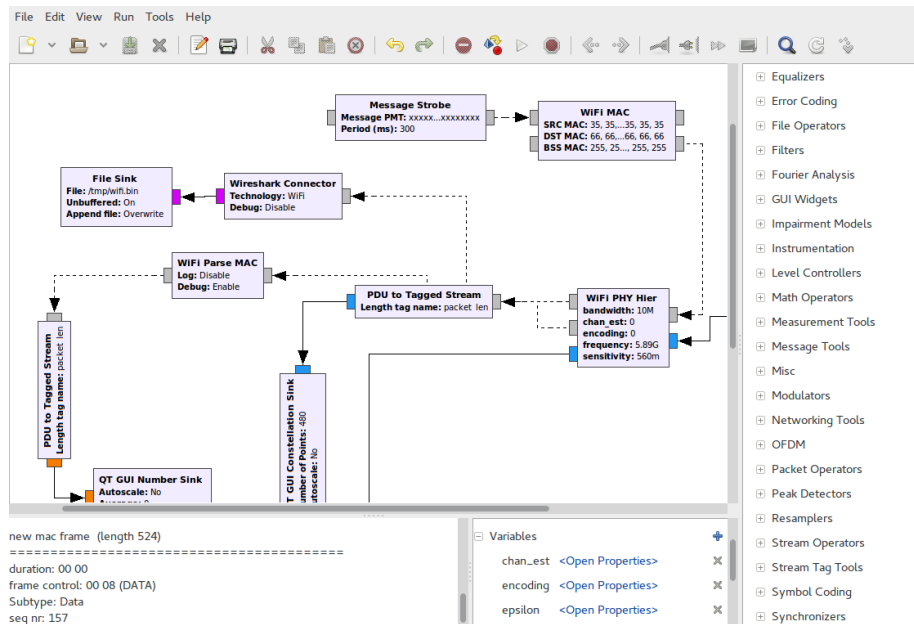


Figure 3.3 – Screenshot of GNU Radio Companion, a tool to setup and configure GNU Radio flow graphs.

display the signal in time and frequency domain. Especially the latter are invaluable tools to debug a flow graph. Apart from GUI elements, GNU Radio comes with a comprehensive block library, supporting, for example, file I/O and various signal processing functions. This library facilitates rapid prototyping, since most functionality is already available and allows concentrating on the technology-specific aspects of the transceiver.

When implementing complex technologies, a transceiver might comprise a large number of blocks, which can lead to complicated flow graphs. To structure a flow graph more clearly, GNU Radio supports *hierarchical blocks*, which encapsulate other blocks. This can be used to create reusable blocks that realize, for example, the PHY or the Medium Access Control (MAC) layer. Structuring a flow graph with hierarchical blocks, we can even resemble an ISO/OSI stack in GRC.

One of the most important aspects of GPP-based SDRs is their computational performance. The faster samples are processed, the higher we can set the sample rate and, therefore, capture a larger bandwidth. GNU Radio's complexity mainly stems from the runtime environment that was built to sustain high sample rates. To benefit from today's multi-core CPUs, GNU Radio starts each block in a separate thread to parallelize processing. Especially in the receiver, parallelization helps to enable high-bandwidth real-time signal processing.

Another major performance gain comes from the Vectorized Library of Kernels (VOLK) [109], a library to exploit Single Instruction Multiple Data (SIMD) extensions

of modern CPUs. The ever lasting strive for higher bandwidths can be addressed with better algorithms, faster PCs, or more efficient implementations. Volk optimizes the latter. Through SIMD extensions like MMX, AVX and SSE on Intel, or NEON on ARM, CPUs provide instructions that operate on vectors, instead of on individual items. These vector operations can lead to considerable speed-ups. For some operations, Rondeau, McCarthy, and O'Shea [109] measured a 16-fold performance increase. Especially in signal processing, SIMD instructions can be exploited in many frequently-used tasks like addition, multiplication, or type conversion. With VOLK, SIMD instructions become accessible for developers by providing wrappers around functions that select the best implementation for the CPU during runtime. Also our IEEE 802.11a/g/p transceiver makes heavy use of VOLK. Without exploiting this hardware acceleration the receiver would not be able to process the required bandwidth on a normal PC. To allow inspecting the performance of a flow graph, GNU Radio comes with *Performance Counters* [110]. Using performance counters, the blocks log metrics like queue utilization or CPU time. These values can be accessed and visualized during runtime. This feature greatly helps in understanding of the computational complexity of the flow graph and identifying bottlenecks.

3.2.2 Microsoft Sora

Apart from GNU Radio, also Sora, the Software Radio developed by Microsoft Research Asia [106], deserves particular attention. At least, at first sight, it is closely related to our work, especially, since it comes with *SoftWiFi*, an IEEE 802.11a/b/g implementation. While both our transceiver and Sora use a GPP-based architecture, they have different design goals and target different use-cases.

While we focus on a modular and easy to extend PHY, Sora wants to provide a platform that combines the flexibility of GPP-based SDRs and the performance of programmable hardware (i.e., DSPs and FPGAs). In this context, performance does not only refer to large bandwidths but also to low latencies in order to meet the timing requirements of the IEEE 802.11 standard. And indeed, Sora manages to decode 20 MHz Orthogonal Frequency-Division Multiplexing (OFDM) signals, while meeting the tough Acknowledgement Frame (ACK) timing requirements of IEEE 802.11, allowing it to interoperate with normal WLAN cards. This is impressive, especially, if we consider that Sora was already developed in 2009. In that regard, Sora presents a milestone in SDR development. To achieve this outstanding performance, Sora relies on a highly optimized hardware-software co-design. It had to be tied to a particular hardware and uses optimized software, resulting in a more complex implementation.

On hardware side, Sora uses a custom radio control board that is connected to the PC via PCI Express, which provides high-throughput, low-latency data transfer between the radio front end and the PC. Given the fact that Sora is developed by

Microsoft Research, it comes as no surprise that it is designed for Microsoft Windows. It is mainly implemented in C, using low-level assembly instructions at selected performance critical parts of the code. To provide the required bandwidth for IEEE 802.11a/b/g, Sora uses a static scheduling scheme that manually distributes signal processing task to CPU cores. Like GNU Radio, Sora uses vectorized instructions, supported through CPU extensions like SSE or AVX. In addition, Sora makes heavy use of Lookup Tables (LUTs), which avoid performing the same computations over and over again. Precomputing, for example, the scrambling and descrambling sequences provides a speedup of 13.6 [106].

Even with these optimizations, it is still not possible to acknowledge a unicast frame in time. It also requires to parallelize frame decoding and generation of the response. As soon as the MAC header is decoded, Sora already starts to generate an ACK. Once the whole frame is received and passed the Cyclic Redundancy Check (CRC), transmission of the precomputed frame only has to be triggered and does not require any further signal processing. A minor issue that arises is that parallel generation of the ACK is not fast enough for short frames. Sora, therefore, also caches ACKs for recent communication partners. While this sophisticated signal processing and performance of the hand-tuned PHY is truly impressive, the high throughput and low latency had to be traded off against a highly optimized implementation that is tied to their software and hardware platform, depending on Microsoft as a supplier. This might be seen as a major drawback, considering that the project did not see any development in the last years.

Chapter 4

An SDR-based WLAN Transceiver

4.1	Physical Layer	44
4.1.1	Transmitter	45
4.1.2	Receiver	47
4.1.3	Transceiver	55
4.2	Verification and Interoperability	56
4.2.1	AWGN Simulations	57
4.2.2	Interoperability Tests	59
4.2.3	Lab Measurements	60
4.3	Computational Complexity	64
4.4	Time-Critical Functionality	69
4.4.1	Channel Access	69
4.4.2	Automatic Gain Control	77
4.5	Conclusion	81

In previous chapters, we provided an overview of Wireless LAN (WLAN) and highlighted the issues that arise when using it for automotive applications. Given the high mobility, diverse propagation environments, and varying user densities, Vehicular Ad Hoc Networks (VANETs) ask for an adapted application-specific network stack. It is well recognized that the development of such a stack is challenging and should not be done with analytical analysis and simulations only. The importance to design and test wireless networks over-the-air in realistic environments is recognized in general [78] and for VANETs in particular [17]. To foster experimentation with automotive WLAN, we developed an IEEE 802.11a/g/p transceiver for a General Purpose Processor (GPP)-based Software Defined Radio (SDR) framework. Having a complete software implementation of WLAN, we are able to inspect and, if needed, modify all aspects of the physical layer (PHY).

In this chapter, we detail the design, implementation, validation, and evaluation of our GNU Radio-based IEEE 802.11a/g/p SDR transceiver. It is the first real-time-capable GPP-based WLAN transceiver for the 10 MHz and 20 MHz Orthogonal Frequency-Division Multiplexing (OFDM) modes of IEEE 802.11 that does not rely on hardware-specific or platform-specific features. Our implementation provides a complete PHY that supports all frame sizes and Modulation and Coding Schemes (MCSs). It can run in real-time on a normal PC and was extensively tested and verified with both commercial WLAN cards and IEEE 802.11p prototypes.

The development of a GNU Radio-based IEEE 802.11a/g/p transceiver was motivated by the fact that existing alternatives have limitations that cannot easily be overcome. For GNU Radio, there was only a proof-of-concept OFDM transceiver that did not implement a specific standard and was not generic enough to be adapted for WLAN. In addition, there was an IEEE 802.11b transceiver, which, however, implemented the very different Direct-Sequence Spread Spectrum (DSSS) PHY [111]. Mango Communications, the vendors of Wireless Open-Access Research Platform (WARP), provide a free-to-use reference implementation for IEEE 802.11. Built for a Field-Programmable Gate Array (FPGA)-based SDR architecture, it provides low latency and deterministic timing, which makes it well-suited for timing sensitive experiments like studies of the Medium Access Control (MAC) layer [97]. The drawback of this approach is, however, that it is not so straightforward to adapt and tied to the rather expensive platform. Furthermore, the FPGA implementation cannot easily be used for simulative performance evaluations, limiting its application to measurements only. Closer related to our work is *SoftWiFi*, an IEEE 802.11a/b/g implementation that was released with Microsoft's Sora [106]. Its best-known feature is probably the support for unicast communication with commodity devices. Through many application-specific optimizations, Sora manages to meet the challenging Acknowledgement Frame (ACK) timings of WLAN. Supporting such low latencies, however, also comes at a price: it had to be designed for a custom radio

control board that is sold by Microsoft, it uses operating system specific features of Microsoft Windows, and it had to be manually optimized to support this particular application. Sora, therefore, sacrifices typical advantages of a GPP-based SDR to meet the MAC timings of WLAN.

We made the deliberate decision to focus on different aspects. While we also rely on a GPP-based SDR framework, we are not targeting a full network stack but focus on the PHY. Our main goal is to provide a solid and accessible implementation that can be easily adapted, allowing it to serve as a base for all kinds of studies. Using GNU Radio, we are not tied to a particular platform: On software side, GNU Radio runs on Windows, macOS, and Linux, supporting even ARM-based embedded devices. On hardware side, GNU Radio supports a wide range of SDR radio front ends, ranging from low-cost SDRs like the HackRF One to more capable radios that support multiple transmit and receive streams. Developed for VANETs, our transceiver was validated with normal WLAN cards and various IEEE 802.11p prototypes. Our modular design lends itself well to plug in and test different channel estimation algorithms, which were the main focus of many studies [21], [65]. As an example and proof-of-concept, we implemented both baseline and state-of-the-art channel estimation algorithms that were designed specifically for VANET.

Another unique benefit of our transceiver is that the GPP-based approach allows using the same implementation for simulations and measurements. Simulations can be used in the prototyping phase to evaluate algorithms in a controlled and reproducible environment. In a second phase, the very same implementation can be used for over-the-air measurements in the lab and in field tests. While our transceiver focuses on the PHY, we used the *split-functionality* approach to implement standard compliant channel access for broadcast transmission. We will show that this can be added without increasing the complexity of the PC implementation, i.e., without sacrificing the accessibility of a GPP-based implementation. We think that this is an interesting add-on, especially for broadcast-based vehicular communications. Overall, our implementation provides unique advantages that are not available with other IEEE 802.11a/g/p prototypes:

Open Source Both our transceiver and the software framework are Open Source software. To foster the use of SDR for research on VANETs and to allow reproduction of our results, we released the code to the community. Fellow researchers can, therefore, study and, if needed, modify all details of the implementation.

Designed for VANETs Our transceiver is tested with both commercial WLAN cards and other IEEE 802.11p prototypes. Furthermore, it features channel estimation algorithms that were developed specifically for VANETs.

Modular Our transceiver adopts GNU Radio's block-based design that allows us to easily exchange receiver components with alternate implementations.

PC implementation The PHY is implemented in high-level programming languages (Python and C++), which makes it accessible, easy to adapt, and well-suited for rapid prototyping.

Platform independent Our PHY implementation does not rely on software-specific or hardware-specific features. It can run on Windows, macOS, and Linux systems, supporting ARM or x86-based CPUs. Furthermore, it is compatible with a wide range of radio front ends.

Integrated workflow Since the PHY is implemented completely in software, we can use the same code in simulations and over-the-air experiments, which allows us to overcome the disconnect between theory and practice.

We believe that our transceiver has many applications for WLAN in general and for VANETs in particular. Since the PHY implementation is independent of the carrier frequency of the signal, it is equally well-suited to study IEEE 1609 Wireless Access in Vehicular Environment (WAVE), ETSI ITS-G5, and ARIB STD-T109. The chapter is based on the following publications:

- B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "Performance Assessment of IEEE 802.11p with an Open Source SDR-based Prototype," *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, pp. 1162–1175, May 2018. DOI: 10.1109/TMC.2017.2751474, © 2018 IEEE.
- 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," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 18, no. 3, pp. 81–90, Jul. 2014. DOI: 10.1145/2721896.2721913.
- 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. DOI: 10.1109/INFCOMW.2015.7179325, © 2015 IEEE.
- B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "Towards an Open Source IEEE 802.11p Stack: A Full SDR-based Transceiver in GNU Radio," in *5th IEEE Vehicular Networking Conference (VNC 2013)*, Boston, MA: IEEE, Dec. 2013, pp. 143–149. DOI: 10.1109/VNC.2013.6737601, © 2013 IEEE.

- 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. DOI: 10.1145/2491246.2491248, © 2013 ACM.

as well as further peer-reviewed papers [28], [31], [32].

4.1 Physical Layer

The IEEE 802.11a/g/p amendments of WLAN introduce similar OFDM PHYs that only differ in terms of the frequency band and the channel bandwidth. IEEE 802.11g and IEEE 802.11a are 20 MHz PHYs that operate on 2.4 GHz and 5 GHz, respectively. IEEE 802.11p, in turn, operates on the Cooperative Intelligent Transportation System (C-ITS) frequency band at 5.9 GHz and has all timings doubled, resulting in 10 MHz channels. For an SDR, doubling all timings translates into a mere change of the sample rate. The very same implementation can, therefore, be used for either mode.

IEEE 802.11a/g/p use OFDM PHYs with 64 subcarriers, out of which 52 are actually used. The others are allocated to guard subcarriers at the edge of the spectrum and a DC subcarrier at the center. The 64 subcarriers are spread over the signal bandwidth, resulting in a subcarrier spacing of 156.25 kHz for IEEE 802.11p and 312.50 kHz for IEEE 802.11a/g. The sample rate of our transceiver is set to the channel bandwidth, i.e., we are not oversampling the signal. With this configuration, we can process 64 samples with a 64-bin Fast Fourier Transform (FFT) to convert between time and frequency domain. To mitigate Intersymbol Interference (ISI), a cyclic prefix corresponding to one fourth of the symbol length is inserted between consecutive symbols, resulting in $1.25 \cdot 64 = 80$ samples per OFDM symbol. To calculate an initial estimate of the channel, two OFDM symbols at the start of the frame serve as block pilots. In addition to that, the data symbols use four subcarriers

Table 4.1 – PHY parameters of the supported OFDM modes.

Parameter	IEEE 802.11p	IEEE 802.11a/g
Bandwidth	10 MHz	20 MHz
OFDM subcarrier	64	64
Subcarrier spacing	156 kHz	312 kHz
OFDM symbol time	8 μ s	4 μ s
Guard Time	1.6 μ s	0.8 μ s
Comb Pilot Spacing	2.2 MHz	4.4 MHz

as comb pilots, allowing to adapt the channel estimate during the reception of the frame. An overview of the parameters is provided in Table 4.1.

For robustness against bit errors, the PHY uses a convolutional code with coding rates of $\frac{1}{2}$, $\frac{2}{3}$, or $\frac{3}{4}$ for Forward Error Correction (FEC). The coded data bits are modulated on the data subcarriers, using BPSK, QPSK, 16-QAM, or 64-QAM. The combination of coding rate and modulation scheme is called the MCS. Valid combinations are summarized in Table 4.2. All of them are supported by our transceiver.

4.1.1 Transmitter

Compared to the receiver, the implementation of the transmitter is rather straightforward. In fact, there was already a GNU Radio implementation available prior to our work [112]. This transmitter was, however, partly implemented in Python and designed for an older version of GNU Radio that lacked many features for seamless packet-based operation. Furthermore, it was limited to fixed-sized frames and did not support setting the MCS per frame. We, therefore, reimplemented it from scratch in C++, translating the frame format specification of the IEEE 802.11 standard to the stream paradigm of GNU Radio. A screenshot of its structure in GRC is shown

Table 4.2 – Our transceiver supports all MCSs. (Reproduced from [23], © 2018 IEEE.)

Modulation	Code Rate	Transmission	Reception
BPSK	$\frac{1}{2}$, $\frac{3}{4}$	✓	✓
QPSK	$\frac{1}{2}$, $\frac{3}{4}$	✓	✓
16-QAM	$\frac{1}{2}$, $\frac{3}{4}$	✓	✓
64-QAM	$\frac{2}{3}$, $\frac{3}{4}$	✓	✓

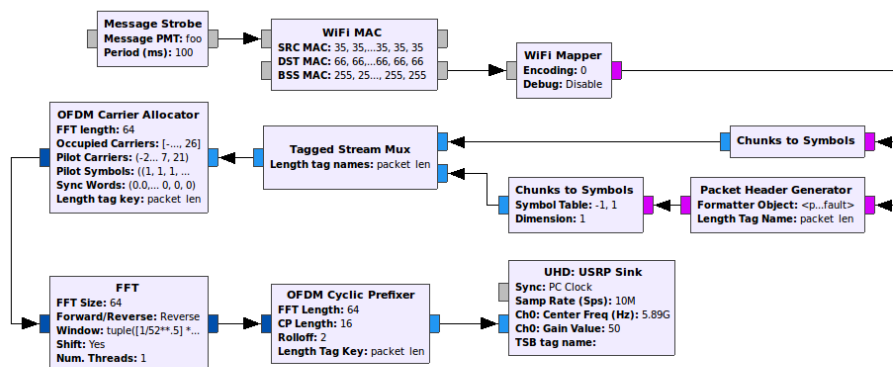


Figure 4.1 – Screenshot of the transmitter in GNU Radio Companion.

in Figure 4.1. The input to the transmitter is the payload of the frame, which is subsequently

- prefixed it with a MAC header that, for example, contains the frame type and the source and destination address;
- 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 inverse FFT;
- prepended with a cyclic prefix to cope with ISI; and
- filtered to improve the spectral shape.

The OFDM data symbols generated through this process get prefixed with a preamble and a BPSK- $\frac{1}{2}$ -encoded signal field that informs the receiver about the length and encoding of the following data.

Apart from the clearly defined procedures of the transmitter, its implementation is further simplified since its computational demands are rather relaxed. Given the fact that we can pre-compute the whole frame before streaming the samples to the SDR, we only have to make sure that the stream does not stall, as interruptions would corrupt the frame. This is, however, no problem in practice. The computational performance of the transmitter, therefore, does not impact its general functionality but mainly its latency and the maximal achievable throughput. Another group profiled our transmitter and suggested optimizations for memory allocation and initialization [113]. These optimization are, however, for GNU Radio blocks that are only used by our transceiver. To avoid duplicating code, we did not reimplement these components but accept the performance penalty.

An architectural limitation of our GPP implementation is its non-deterministic processing time. Calculating the samples on a normal PC introduces jitter that makes it impossible to generate frames with a precise inter-frame space. In practice, it is, however, easily possible to work around this limitation by pre-computing a signal with zero-padded frames and streaming this signal to the radio front end in one go.

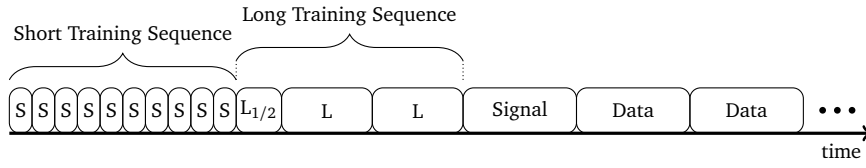


Figure 4.2 – WLAN frames comprise a short and a long training sequence for synchronization; a signal field, containing information about the length and encoding of the frame; and the data symbols, carrying the actual payload. (Reproduced from [23], © 2018 IEEE.)

4.1.2 Receiver

Compared to the transmitter, the design of the receiver is much more challenging. IEEE 802.11a/g/p uses a complex PHY with a bandwidth of up to 20 MHz. Even without oversampling, the receiver, therefore, has to process the baseband signal with a rate of 20 Msps. Realizing a receiver on a GPP-based SDR is an important contribution of our work and will, therefore, be described in more detail. Here, the novelty is not the algorithm but the design, implementation, and evaluation of a modular implementation on a normal PC. Given the fact that there was only a basic proof-of-concept OFDM implementation for GNU Radio (that supported only small bandwidths) and the highly optimized Sora framework (that was strictly tied to a hardware and software environment), it was not clear whether such a receiver would be possible in the first place.

The design of the receiver is mainly motivated by the frame structure of IEEE 802.11a/g/p frames, which is depicted in Figure 4.2. To synchronize on the frame, it starts with a short and a long training sequence, usually called the short and long preamble. Despite their names, both parts of the preamble have the same length, spanning over two OFDM symbols. The short preamble consists of a short pattern that repeats ten times. This signal is well-suited for frame detection based on the autocorrelation of the signal.

4.1.2.1 Frame Detection

When implementing frame detection, the computational complexity is of particular interest. Naturally, the frame detection algorithm has to process the whole sample stream to recognize the start of a frame. Even when the receiver is already synchronized and about to decode a frame, it might make sense to continue frame detection and search for another preamble to resynchronize on an interfering frame with a higher power. This process is called *capturing* and is known to be implemented on typical WLAN receivers [114], [115].

The need to use an algorithm with a low computational complexity rules out a naive implementation based on the cross-correlation of the sample stream with

the known pattern of the preamble. If n is the length of the correlation sequence, this approach would require n complex multiplications per baseband sample. A more efficient method for frame detection is based on the well-known Schmidl-Cox algorithm [116], which exploits the autocorrelation of the short preamble. A detailed comparison of the autocorrelation and cross-correlation methods is available in Chia-Horng [117]. Following [117, Algorithm 1], we calculate the autocorrelation a of the baseband sample stream s with a lag of 16 samples, which corresponds to the length of the repeating pattern of the short preamble.

$$a[n] = \sum_{k=0}^{N_{\text{win}}+15} s[n+k] \bar{s}[n+k+16] . \quad (4.1)$$

Here, $\bar{s}$ denotes the complex conjugate of s and N_{win} is a configurable parameter to apply a moving average, which we set to 48. Including the lag of the autocorrelation, we average values that span 64 samples, which corresponds to the size of an OFDM symbol. While this value was found to work well in our experiments, we did not conduct a separate parameter study for this parameter. Since the absolute value of the autocorrelation depends on the input power level and, therefore, the gain setting of the device, we normalize the value with the average input power level p to calculate the autocorrelation coefficient as

$$p[n] = \sum_{k=0}^{N_{\text{win}}-1} s[n+k] \bar{s}[n+k] ; \quad (4.2)$$

$$c[n] = \frac{|a[n]|}{p[n]} . \quad (4.3)$$

Here, $|a[n]|$ denotes the magnitude of a . Exemplary courses of the autocorrelation coefficient c at different SNRs are shown in Figure 4.3. In the plot, we consider an Additive White Gaussian Noise (AWGN) channel without any hardware impairments, like frequency or clock offsets. Since the autocorrelation stays high during the short preamble and since we do not average over the whole preamble length, we can see the typical plateau of the autocorrelation coefficient during frame start.

The calculation of the autocorrelation coefficient can be composed from simple mathematical functions that are readily available in GNU Radio. An overview of the receiver flow graph in GRC is depicted in Figure 4.4, where we annotate the stream of p , a , and c . The blocks make heavy use of vectorized instructions through GNU Radio's Vectorized Library of Kernels (VOLK), allowing a typical PC to process even 20 MHz signals without dropping samples. A more detailed study of the computational complexity is presented in Section 4.3.

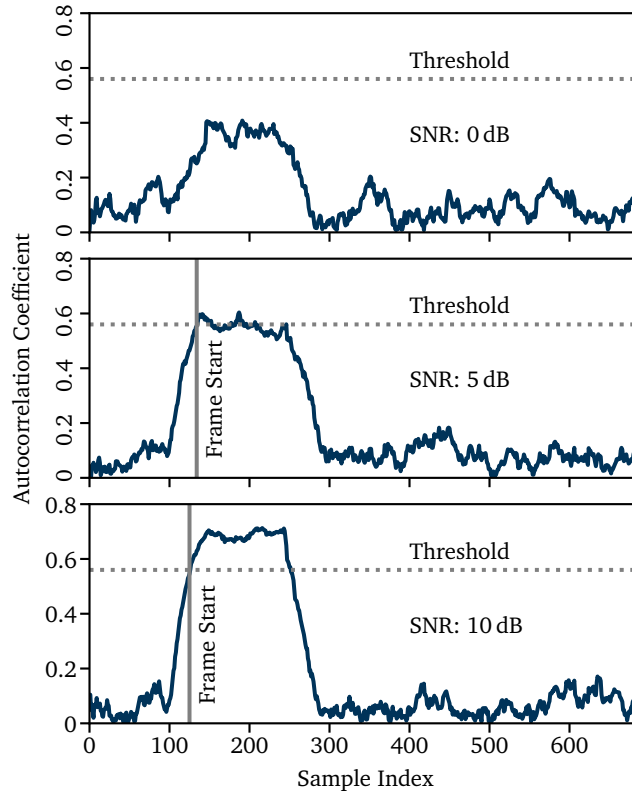


Figure 4.3 – Typical course of the autocorrelation coefficient during frame start at different SNR levels. Frame detection is triggered once the autocorrelation coefficient exceeds a predefined threshold (dotted line).

The autocorrelation coefficient is monitored by our *Sync Short* block (see Figure 4.4), which triggers frame detection once the correlation exceeds a predefined threshold. This threshold defines the sensitivity of the transceiver and balances the trade-off between computational overhead and frame error rate. If we set the threshold too low, the receiver might trigger on noise, resulting in unnecessary signal processing. If we set the threshold too high, the receiver might miss frames, resulting in bad performance. Based on a parameter study that we detail in Section 4.1.2.6, we selected a threshold of 0.56.

4.1.2.2 Symbol Alignment

Once a frame is detected, we have to derive the position of the OFDM symbols in order to align the FFT in the receiver and decode the frame. This is done in the *Sync Long* block (see Figure 4.4). Since the plateau of the short preamble is not well-suited for this task, we use the long preamble for more precise alignment. At this stage, we employ the computationally more complex cross-correlation with the known

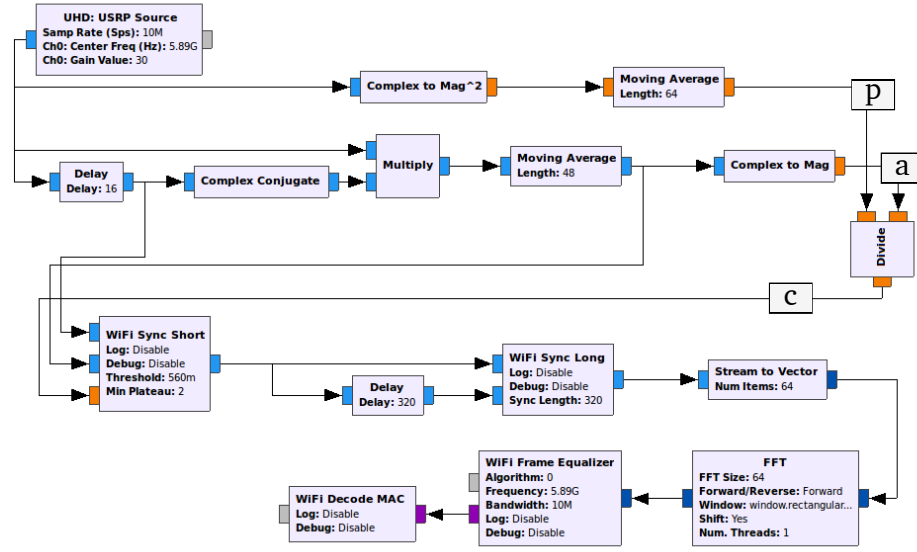


Figure 4.4 – Screenshot of the receiver in GNU Radio Companion. The signal for the power p , the autocorrelation a , and the autocorrelation coefficient c are annotated.

time-domain pattern of the long preamble LP. This pattern spans over 64 samples and repeats 2.5 times during the long preamble (see Figure 4.2). Cross-correlation with this pattern results in 64 complex multiplications per sample, which is feasible at this stage since we only have to calculate it in a small window of length N_{preamble} following the frame start. In our implementation, we set the length of the window to the conservative value of 320 samples, corresponding to the total length of the short and long preamble.

Cross-correlating the signal with the preamble yields very localized peaks that allow precise alignment. The characteristic course of the cross-correlation at various SNR levels is depicted in Figure 4.5. Like in the previous plot, we consider an AWGN channel and no hardware impairments like clock or frequency offsets. To determine the frame start, the receiver orders the values of the cross-correlation by their magnitude and searches the three peaks $N_{\mathcal{P}}$ (using the $\arg \max_3$ operator, which selects the indices of the three highest values).

$$N_{\mathcal{P}} = \arg \max_3 \sum_{n \in \{0, \dots, N_{\text{preamble}}\}}^{63} s[n+k] \overline{\text{LP}}[k], \quad (4.4)$$

The first OFDM data symbol starts 64 samples after the last peak, which is the peak with the highest sample index.

$$n_{\mathcal{P}} = \max(N_{\mathcal{P}}) + 64, \quad (4.5)$$

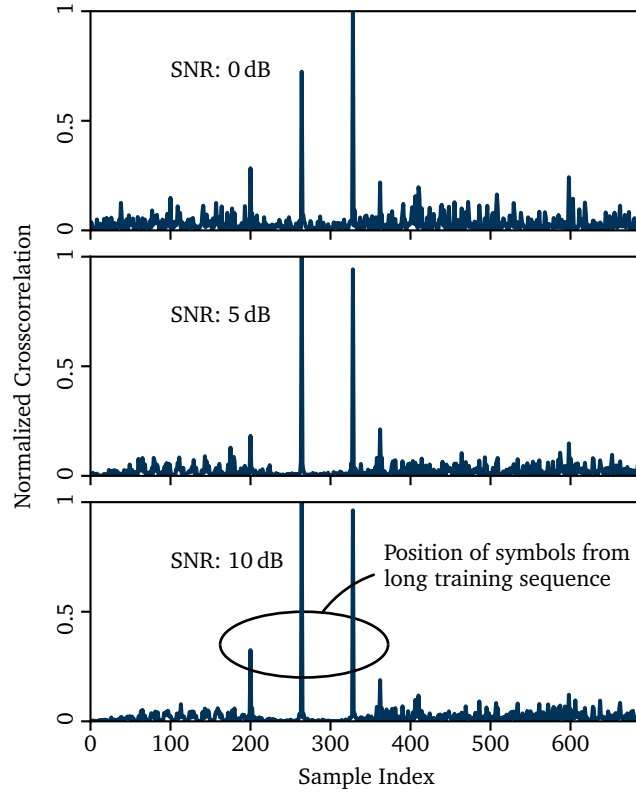


Figure 4.5 – Cross-correlation of a frame with the known pattern of the long preamble, as calculated by the receiver to determine OFDM symbol alignment.

Knowing the start of the frame, we can remove the cyclic prefix before passing the samples to down-stream blocks. Since most channel estimation algorithms use the long preamble as block pilots to calculate an initial estimate of the channel, we also forward the symbols of the long preamble. As shown in Figure 4.2, the long preamble symbols are sent back-to-back, whereas the data symbols use a cyclic-prefix of 16 samples.

$$\begin{aligned}
 s \leftarrow & \left(\underbrace{s[n_p - 128], \dots, s[n_p - 65]}_{\text{long preamble 1}}, \underbrace{s[n_p - 64], \dots, s[n_p - 1]}_{\text{long preamble 2}}, \right. \\
 & \left. \underbrace{s[n_p + 16], \dots, s[n_p + 79]}_{\text{first data symbol}}, \underbrace{s[n_p + 80 + 16], \dots, s[n_p + 80 + 79]}_{\text{second data symbol}} \right). \quad (4.6)
 \end{aligned}$$

The fact that we first cross-correlate the signal and then search for the peaks is the reason why we duplicate the signal between the Sync Short and Sync Long block. Since there is no easy way to traverse the sample stream in reverse direction, we locate the peaks within one branch and use a delayed copy to extract the symbols.

Using GNU Radio's stream tags, we annotate the first sample of a frame to indicate its start to the following blocks. At this stage of the decoding process, we do not know the length of the frame. We, therefore, pass on samples that correspond to a maximum sized frame, encoded with the least efficient MCS, i.e., a frame with the longest possible duration. Copying samples to downstream blocks can, however, be preempted if another frame is detected. This happens either if a high power frame interferes and triggers frame detection or if a short frame is immediately followed by another frame. Through this mechanism, we implement capturing and support small frames that are sent with short inter-arrival times like Request To Send (RTS)/Clear To Send (CTS) pairs or ACKs.

4.1.2.3 Frequency Offset Correction

An additional task that is performed by the Sync Short and Sync Long block is frequency offset correction. Both blocks use the method proposed in [118], which exploits the cyclic nature of the short and long preamble to estimate the frequency offset. If we, for example, consider the short preamble with its 16 sample pattern, then, ideally, $s[n]$ would correspond to $s[n+16]$. However, due to imperfections of the oscillators in sender and receiver, they will operate at slightly different frequencies. Therefore, $s[n+16]$ is slightly rotated with regard to $s[n]$, resulting in a non-zero argument for $s[n]\bar{s}[n+16]$. Neglecting noise, the argument of that product corresponds to 16 times the rotation $\Delta\alpha$ that is introduced by the frequency offset between consecutive samples. To estimate this offset, we average over the length of the preamble and calculate

$$\Delta\alpha = \frac{1}{16} \arg \left(\sum_{n=0}^{N_{\text{short}}-1-16} s[n] \bar{s}[n+16] \right), \quad (4.7)$$

where N_{short} is 160 samples, which corresponds to the length of the short training sequence.

Finally, the frequency offset is compensated through

$$s[n] \leftarrow s[n] e^{i(n\Delta\alpha)}. \quad (4.8)$$

For even more fine-grained correction, we use the long preamble in the Sync Long block in a similar manner to compensate the residual frequency offset.

4.1.2.4 Channel Estimation

Since the Sync Long block already extracted the OFDM symbols, we can directly use a 64-bin FFT to switch to frequency domain. At that stage, we have to employ a channel estimation algorithm to estimate and reverse the perturbations induced

by the channel. This is a crucial part of the receiver as it has a major impact on its performance. Especially in the VANET context, channel estimation algorithms received significant attention, as there are concerns whether simple schemes will be able to cope with the high dynamics of vehicular networks [10], [11], [13], [65], [66]. To ease experimentation with new algorithms, we implemented a generic interface, allowing the user to easily extend the transceiver with new channel estimation algorithms and even change them during runtime. We implemented both baseline and state-of-the-art algorithms, developed specifically for VANETs.

A more detailed comparison of the algorithms and their performance in a realistic environment is presented in Section 5.1.2. For the performance evaluation in this chapter, we use the Least Squares (LS) equalizer, which is a simple algorithm that is often used as a baseline and cited to be a typical candidate for hardware implementations [12], [13].

The algorithm 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. Denoting the estimate of a value X as $\hat{X}$, we calculate the channel H at subcarrier k as

$$\hat{H}(k) = \frac{Y_1(k) + Y_2(k)}{2X_{LP}(k)}, \quad (4.9)$$

where $Y_{1,2}$ are the two received copies of the long training sequence and X_{LP} its known value. With the LS equalizer, this initial estimate is kept during the whole frame. While computationally very efficient, it is well-known that this algorithm suffers as frames get longer or the coherence time of the channel gets shorter [12], [13]. Using the vector of the channel estimates $\hat{H}$, we correct data symbol X through

$$\hat{X} = \frac{X}{\hat{H}}. \quad (4.10)$$

4.1.2.5 Decoding

After correcting the symbols, we demap the noisy subcarrier constellations to the closest ideal constellation points, which, in turn, directly correspond to bit sequences. At bit-level, the receiver decodes the information using the reversed process implemented in the transmitter. To decode the convolution code, we use a highly optimized hard-bit Viterbi decoder that was contributed by another group [113]. Using Lookup Tables (LUTs) and SSE2 instructions, the decoder provides significant improvements over our previous implementation, which was identified as a performance bottleneck [113]. As the discussion on the computational complexity in Section 4.3 will show, this is an important contribution as even the improved decoder presents a CPU bottleneck. A potential improvement of the receiver would be the use of soft-bits [119] instead of hard-bits. We, however, did not explore this

option in detail, as the need to associate each bit with a confidence level would introduce considerable complexity in an already performance critical component of the receiver.

The first OFDM symbol contains the signal field, which encodes the length and MCS of the following data. This symbol is decoded separately, and the information is used to configure the demodulation and decoding process.

4.1.2.6 Sensitivity

While the structure of the receiver is complex, there are not many free parameters that we have to choose. One interesting exception is the frame detection threshold, which balances sensitivity against computational overhead. A low threshold increases the sensitivity as even low SNR frames can trigger the decoding process. However, a low threshold also leads to false positives when frame detection is triggered by noise. These false positives cause unnecessary decoding attempts when no signal is present, increasing the computational overhead. A high value for the threshold, in turn, would lead to false negatives when frames are missed and never processed by the decoder.

To determine the threshold, we simulated the frame error rate of 435 Byte BPSK- $\frac{1}{2}$ frames over an AWGN channel. We chose the most robust MCS, since it is most sensitive to threshold changes. This is because false positives are completely independent of any frame transmissions and only cause computational overhead. False negatives, in turn, are worse for BPSK- $\frac{1}{2}$, since chances are high that the missed frame could have been decoded. If, in contrast, a 64-QAM- $\frac{3}{4}$ frame is missed, it is likely that the signal quality was in any case not good enough to decode the frame.

In this and the following experiments, we often use 435 Byte frames, which we believe is about the size of a typical VANET frame. There is, however, no correct size, as it heavily depends on the

- application or the frame type within one application,
- the certificate information, which can be a hash of a certificate or a complete certificate chain.

Furthermore, since our experiments assess PHY performance, the frame size refers to the MAC Protocol Data Unit (MPDU), which corresponds to the overall size as indicated in the signal field. This includes the MAC header, the Logical Link Control (LLC), the payload, and the CRC.

The frame delivery ratio for different sensitivities is shown in Figure 4.6. The error bars indicate the 95 % confidence intervals, which are based on 20 runs with 100 frames per run. As expected, the decoding performance of the receiver improves with lower values for the threshold. Between 0.68 and 0.62, we can see a rather big

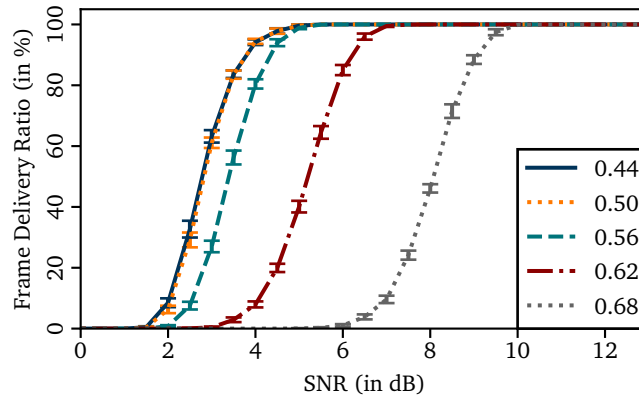


Figure 4.6 – Delivery ratio of 435 Byte BPSK- $1/2$ frames at different sensitivity levels.

improvement, shifting the error curve by about 2 dB. This indicates false negatives, as decodable frames are missed by the receiver. For smaller thresholds, the performance improvement gets smaller, and the curve converges.

Based on these results, we select a value of 0.56 for our transceiver. This value is close to the converged curve and, therefore, provides nearly optimal decoding performance. Still, in this experiment, we only looked at one aspect of the trade-off, i.e., we selected the threshold only based on the frame error rate. Nevertheless, we think that this is reasonable, since the primary goal is to provide good decoding performance. Apart from that, we provide a detailed analysis of the computational complexity of the receiver in Section 4.3.

4.1.3 Transceiver

As shown in Figure 4.7, we can combine transmit and receive chains to a complete transceiver. To create a more structured flow graph layout, we encapsulated transmit and receive chains in a hierarchical block that presents the PHY. If this transceiver is used with a half-duplex radio, the SDR will, by default, receive and only switch to transmit mode when a frame is supposed to be sent. This switch is handled by the SDR and does not require any additional logic in our implementation.

Finally, the received frames can be exported in the PCAP format, a popular packet capturing format, which allows analyzing the traffic with network monitoring software like Wireshark. Similar to a normal WLAN card, we annotate metadata of the frame in the Radiotap header, which includes information like the MCS or the bandwidth. A screenshot of the transceiver's graphical user interface is depicted in Figure 4.8. It shows a log of the received frames in the console (bottom left) and in Wireshark (top left). Furthermore, it visualizes the signal in time domain (top right) and shows a constellation plot of a QPSK signal (bottom right). Apart from logging

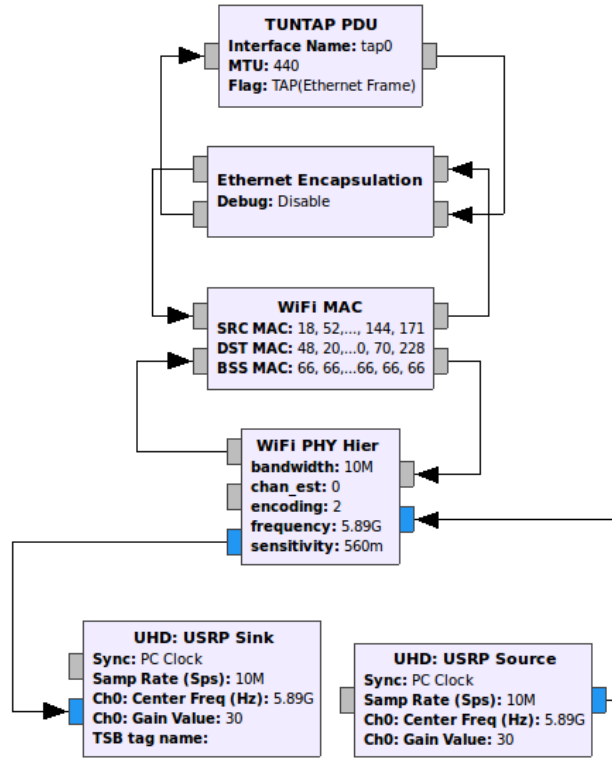


Figure 4.7 – Screenshot of the transceiver in GNU Radio Companion.

and visualizing 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.

4.2 Verification and Interoperability

After presenting the design of our SDR-based IEEE 802.11a/g/p transceiver, we verify its correctness in several steps: We start with AWGN simulations, which allows us to compare the performance of our implementation with published results. While this can show that the results are reasonable, it does not verify the correctness. This is done in subsequent interoperability tests with both commercial WLAN cards and IEEE 802.11p prototypes. Apart from pure functionality, the computational performance of the implementation is very relevant for a GPP-based SDR. It is crucial that we can reliably process all samples, as dropped samples would lead to frame losses at the receiver. We, therefore, measure the CPU time of individual

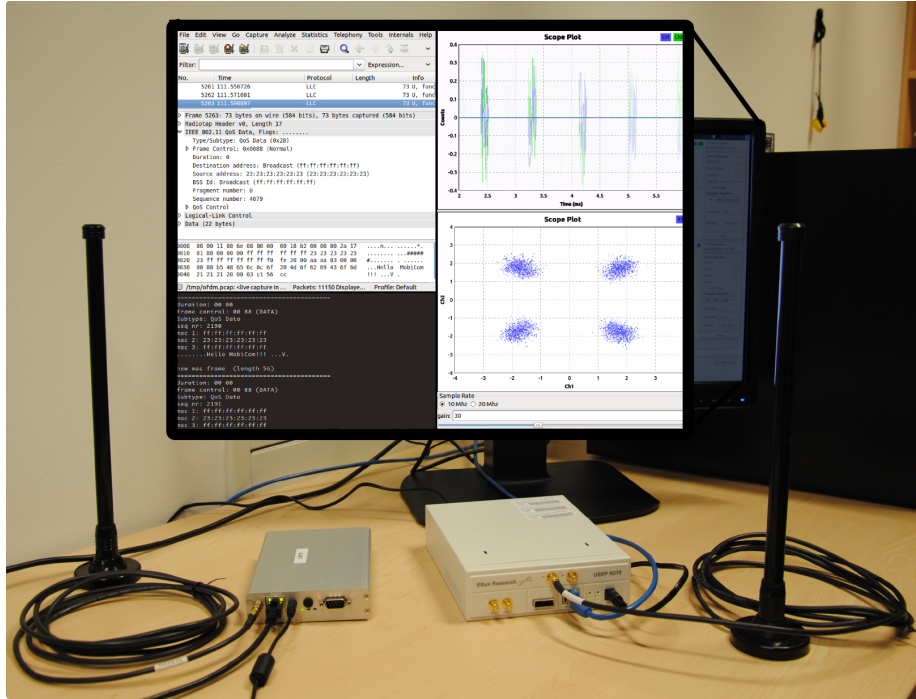


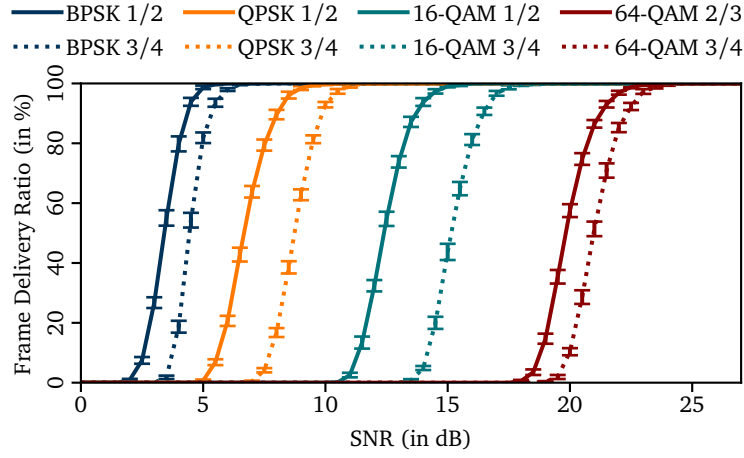
Figure 4.8 – Screenshot of the transceiver’s graphical user interface while decoding a QPSK frame. (Reproduced from [31].)

transceiver components with varying traffic loads ranging from a completely idle to a completely saturated channel.

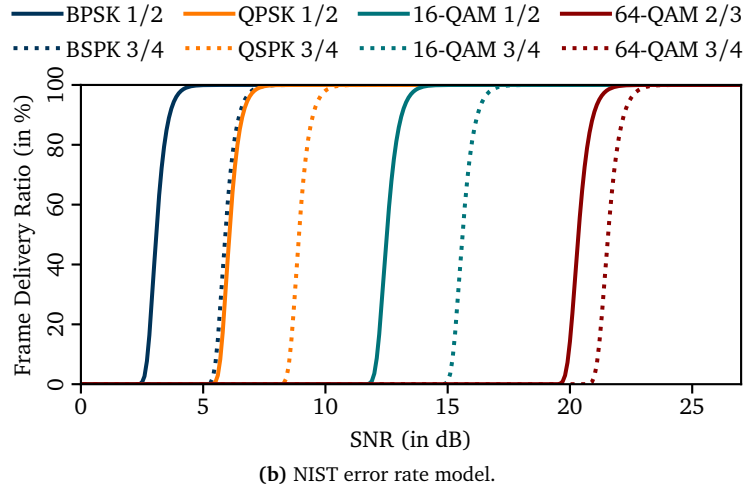
4.2.1 AWGN Simulations

To test the general functionality of the implementation and to show that it produces reasonable results, we conducted simulations over an AWGN channel. In this experiment, we do not consider hardware impairments like frequency or sample clock offsets. Measuring the frame delivery ratio, we highlight the transceiver’s applicability for simulations and present baseline results that can be compared to related works. The results for 435 Byte frames and all MCSs are shown in Figure 4.9a.

On a basic level, we see the transceiver can decode its own frames and produces reasonable results in the sense that more robust MCSs work at lower SNRs. Furthermore, there are no systematic problems, as all MCSs converge to 100 % for high SNRs. Apart from these basic insights, we can compare our results to results available in the literature. The NIST PHY simulation model [120], for example, is adopted by popular network simulators like *Veins* or *ns-3*. Many research results, therefore, rely on the validity and accuracy of this model. It is analytically derived and validated with commercial WLAN cards in a testbed. To compare our results with



(a) Our transceiver. (Reproduced from [23], © 2018 IEEE.)



(b) NIST error rate model.

Figure 4.9 – Delivery ratio of 435 Byte frames over an AWGN channel.

the NIST model, we plot the corresponding error curves in Figure 4.9b. Considering the fact that receiver implementation comprises many design decisions that impact the results, the curves match surprisingly well. For some MCSs, the NIST model is slightly more conservative in the sense that it requires 1 dB to 2 dB higher SNR for a similar performance. The BPSK- $\frac{3}{4}$ curve exhibits the biggest difference to the model. While our implementation shows a clear difference between BPSK- $\frac{3}{4}$ and QPSK- $\frac{1}{2}$, the NIST model produces very similar curves. This, however, seems to be an artifact of the NIST model, as also the testbed results in Pei and Henderson [120] show a clear difference between these MCSs.

Similar results are also reported by Mittag et al. [76], who present a detailed PHY simulation model for ns-3. Like our SDR implementation, this model operates on the complex baseband representation of the sampled waveform, i.e., it uses the

same level of abstraction. This implementation is, however, not real-time capable and cannot be connected to an SDR radio front end. It is, therefore, limited to simulations only. While the results presented in the paper are for 500 Byte frames, they also match well with our error curves, shifted about 2 dB towards higher SNRs, which can be explained by the larger frame size. Overall, this first experiment shows that our simulation results match well with the literature and that our transceiver provides reasonable performance, at least over simple AWGN channels.

4.2.2 Interoperability Tests

In the simulations, we fed back the sample stream generated by the transmitter into the receiver without involving any hardware. While this produced reasonable results, we did not yet prove that we are able to transmit and receive standard compliant IEEE 802.11 frames over the air. To show that this is the case, we performed extensive interoperability tests in our lab and validated our implementation with both commercial IEEE 802.11a/g cards and IEEE 802.11p prototypes.

In these experiments, we used the same code as in the simulations together with an Ettus Research B210 SDR radio front end. The B210 covers a large frequency range, including both the 2.4 GHz and the 5.9 GHz band. While we selected this particular radio for our tests, we did not use any hardware-specific features. Our transceiver works just as well with other radios that are supported by GNU Radio, operate on the frequency band of interest, and provide enough bandwidth for the PHY mode that is supposed to be studied.

An exemplary list of some devices that we tested with our transceiver is provided in Table 4.3. The list contains widely used chips from Intel and Apple, used in popular commercial devices like the MacBook Air. If the card supported it, we tested both the 2.4 GHz and the 5 GHz band. We managed to establish bi-directional communication with each device, using either MCS, which highlights the correctness of our implementation and shows that we are able to send and receive standard compliant frames.

Table 4.3 – WLAN cards and IEEE 802.11p prototypes used in our interoperability tests. (Reproduced from [23], © 2018 IEEE.)

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	✓

Apart from commercial WLAN devices, we also tested IEEE 802.11p prototypes. The Unex DCMA-86P2 is a commercial IEEE 802.11p-capable MiniPCI WLAN card that was used successfully in field tests. It is based on an Atheros chip supported by the *ath5k* Linux driver. To switch to IEEE 802.11p, we had to modify the kernel driver to switch to 10 MHz and adapt the regulatory domain settings tune to the 5.9 GHz band. Apart from the Unex DCMA-86P2, we tested an *ath9k*-based Atheros card. This card 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 1609 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. Again, we manage to set up bidirectional communication, with all prototypes, supporting our claim of a standard compliant PHY implementation. Furthermore, by going beyond simulations, these tests show that we implemented a transceiver that is able to deal with impairments of real hardware and unsynchronized clocks.

4.2.3 Lab Measurements

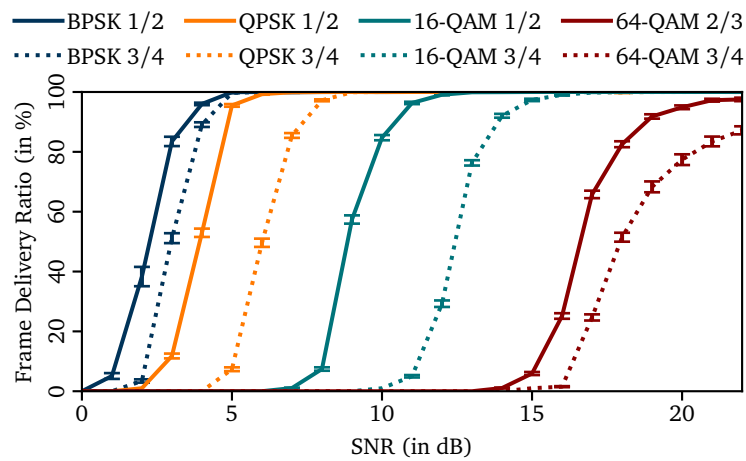
Apart from sole connectivity test, we wanted to further quantify the performance of sender and receiver by comparing them to commercial devices and IEEE 802.11p prototypes. In these experiments, we evaluate the performance of the transceiver in a controlled lab environment over simple, static channels. These experiments are important to establish our implementation as a tool for experimentation with WLANs. Providing comparable performance to other WLAN cards means that there are no systematic problems and we can use commercial devices and our transceiver interchangeably.

4.2.3.1 Transmitter

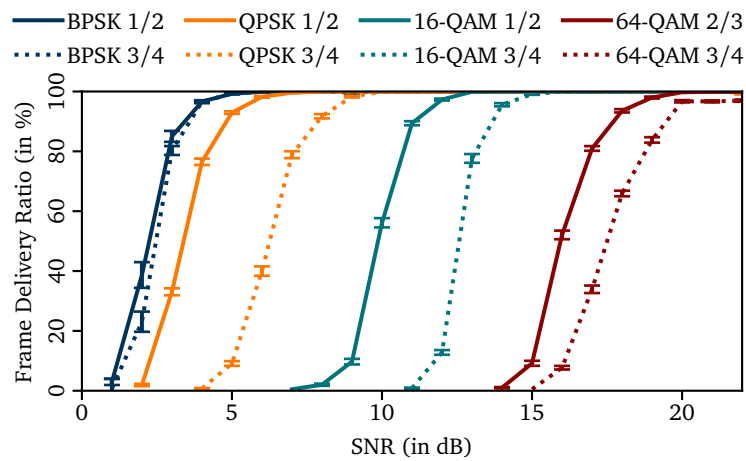
To assess the performance of the transmitter, we connect an Ettus Research N210 equipped with a CBX daughterboard via cable to a Unex DCMA-86P2 WLAN card. The cable connection rules out interference and guarantees stable channel conditions, required to compare the performance of different devices. The cable is, however, no artificial simplification for the transceivers as the devices are not synchronized or benefit in any other way from this setup. The Unex DCMA-86P2 is able to operate on the 5.9 GHz band with a bandwidth of 10 MHz. To avoid damaging the devices, we use attenuators to reduce the signal power. In the experiment, we send 133 Byte frames with either MCS and vary the output power. For each configuration, i.e., pair of MCS and output power, we send 10 000 frames on WLAN channel 172 at 5.86 GHz. A summary of the setup is given in Table 4.4.

Table 4.4 – The most relevant components of our test system for lab measurements. (Reproduced from [30], © 2013 IEEE.)

Component	Type
CPU	Intel Core i7-2600 CPU 3.40GHz
RAM	16 GByte
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



(a) SDR transmits frames to a commercial WLAN card.



(b) Sender and receiver are commercial WLAN cards.

Figure 4.10 – Frame delivery ratio between two devices that are connected via cable. The frame size is 133 Byte. (Reproduced from [30], © 2013 IEEE.)

The receiving WLAN card is put into monitor mode, where it adds a Radiotap header to each overheard frame. This header contains metadata like signal and noise levels, which are used to calculate the SNR at the receiver. The results of the experiment are depicted in Figure 4.10a. What we can immediately see is that their qualitative course is similar to our simulations results. Like in the previous plots, the error bars correspond to the confidence intervals of the mean for a confidence level of 95 %. To compare the results to a commercial device, we repeated the measurements with another Unix device as a sender. Since the performance can vary between individual cards, we made sure to use the same receiver for both measurements. The results are shown in Figure 4.10b. We can see that the results of the WLAN transmitter match well with our SDR transmitter. The exception is the 64-QAM- $\frac{3}{4}$ encoding, where we experience worse performance with the SDR.

Since we do not see such an effect in simulations and since the system 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 waveform. For these reasons, we believe that the deviation in the results is caused by hardware impairments. Device characteristics that might cause such a behavior are oscillator drift and, more likely, nonlinearities in the amplifier that might disturb the signal, which leads to packet errors, especially with higher order modulations.

4.2.3.2 Receiver

Similar to the transmitter, we also want to quantify the performance of the receiver. In a first experiment, we compare our receiver to an *Air Live X.USB* WLAN card, operating in IEEE 802.11a mode, i.e., we were using the 5 GHz band and a bandwidth of 20 MHz. For the SDR, we used an Ettus Research N210 with an XVCR2450 daughterboard, which is specifically designed for the 2.4 GHz and the 5 GHz band. This time, the experiment was conducted over the air in an office environment. We chose the 5 GHz band to avoid the crowded 2.4 GHz Industrial Scientific and Medical (ISM) band, where uncontrolled interference could invalidate the results. Due to the limited space, we had to use attenuators at the transmitter to lower the signal power and generate packet error curves. The antennas were 3 dBi VERT2450 dipoles, and the distance between sender and receiver was approximately 6 m. The PC and software configuration were the same as in the previous experiment.

As transmitter, we used a Unix DCMA-86P2 to send 63 Byte frames. At the time of the experiment, we only had the normal Linux driver available. This driver was limited to packet injection with the most robust MCS, limiting us to BPSK- $\frac{1}{2}$ frames. In our experiment, we varied the transmit power between 0 dBm and 18 dBm in

steps of 1 dBm. For every power setting, we conducted 200 measurement runs, sending 100 frames per run.

The resulting frame delivery ratio with 95 % confidence intervals is plotted in Figure 4.11. Note that in contrast to typical error curves, we plot the transmit power on the x-axis and not the SNR. This stems from the fact that, at the moment, the receiver does not estimate the SNR. By increasing the transmit power by 1 dB, we know that the SNR is 1 dB higher, but we do not know the absolute value. The values on the x-axis can, therefore, be interpreted as relative increases of the SNR. The results show that our receiver and the commercial card show comparable performance. For higher transmit powers, both devices approach 100 %, which highlights the fact that there are no systematic errors in our receiver. Furthermore, the width of the interval in which the frame delivery ratio rises matches very well with the commercial card. A limitation of this measurement method is that we cannot guarantee that the received SNR is similar for both devices. Due to channel effects, one device might experience a higher attenuation, which would mean that the SNR curves of the receiver might be shifted with regard to Figure 4.11. We, however, tried different antenna placements and found the results to be stable, indicating that the performances of the devices are indeed similar.

While IEEE 802.11a/g/p define very similar PHYs, we also wanted to test the performance of IEEE 802.11p in order to emphasize the transceiver's applicability for research on VANETs. Therefore, we conducted a second experiment in which we sent frames with a Cohda Wireless MK2, an IEEE 802.11p prototype that was used in many field tests. The measurements are, again, performed over the air in an office environment, but, this time, on the otherwise vacant 5.9 GHz band, using a bandwidth of 10 MHz. The experiment was conducted with an earlier version of the

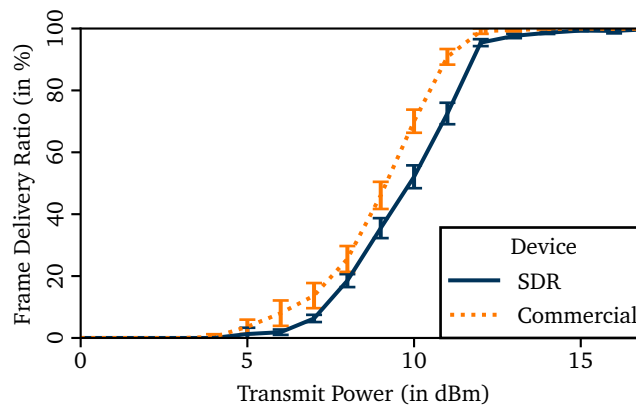


Figure 4.11 – Comparison of the receive performance of a commercial device and our SDR implementation. (Reproduced from [33] with permission, © 2013 ACM.)

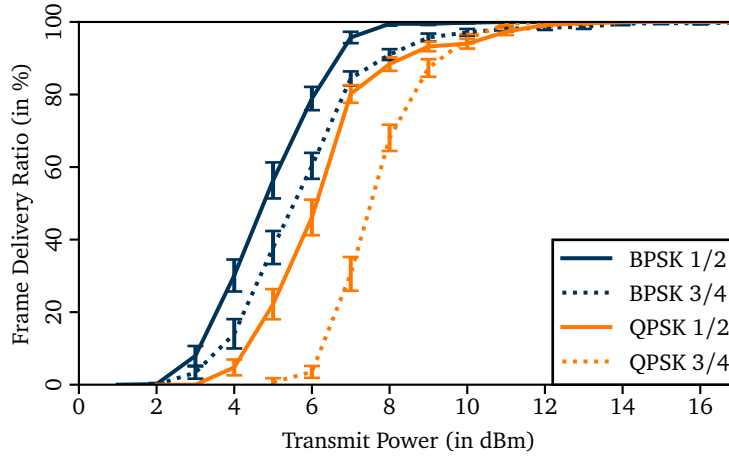


Figure 4.12 – Delivery ratio of 95 Byte frames that are sent from a Cohda Wireless MK2 and received with our SDR implementation. (Reproduced from [33] with permission, © 2013 ACM.)

receiver, which was limited to BPSK and QPSK modulations. At that stage, we only had a basic channel estimation algorithm implemented that did not normalize the amplitude, limiting us to phase-shift keying. The current version of the transceiver does not have this limitation but implements a full IEEE 802.11a/g/p transceiver that supports all frame sizes and MCS. We varied the transmit power of the MK2 in 1 dBm steps and performed 30 runs for every configuration, sending 100 frames per run. The results with 95 % confidence intervals are plotted in Figure 4.12. We can see that at least the MCSs that use BPSK and QPSK modulations work, as their frame delivery ratio approach 100 %. Furthermore, the results are reasonable in the sense that higher bitrates suffer from higher packet losses.

Overall, the results indicate that our transceiver works with 10 MHz and 20 MHz channels and provides reasonable performance that is comparable to commercial WLAN cards. Furthermore, it does not suffer from systematic errors, highlighting its applicability for research on WLANs.

4.3 Computational Complexity

When it comes to GPP-based SDRs, the computational performance and the ability to process samples in real-time are critical factors. A correct implementation that is not capable of processing the sample stream on a normal PC would be of limited use. Dropped samples could lead to packet loss in the receiver and, ultimately, to wrong interpretation of the measurement results, for example, if lost frames are regarded as effects of the wireless channel or shortcomings of the receive algorithm. Overall, this could limit the transceiver’s applicability for academic studies. Furthermore,

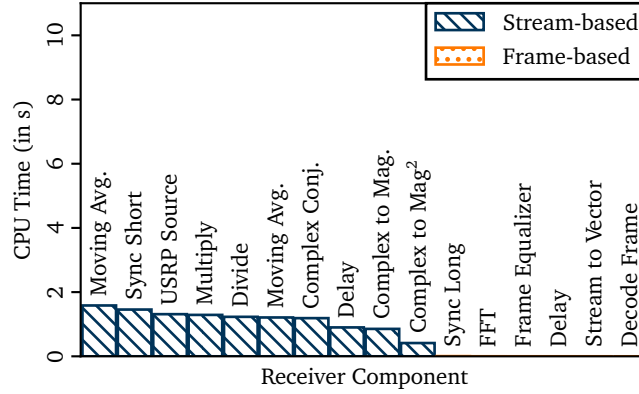
when the SDR detects that samples are dropped, the driver interrupts reception and continues only if the sample queue is completely flushed. Therefore, even a single overrun can have a great impact and cause the loss of several frames. Given its importance, we have an in-depth look at the computational complexity of the transceiver.

Compared to the receiver, the computational complexity of the transmitter is less challenging as the whole waveform can be pre-generated before streaming the samples to the SDR. We only have to make sure that the sample stream does not stall, which would corrupt the signal. This is, however, no problem in practice.

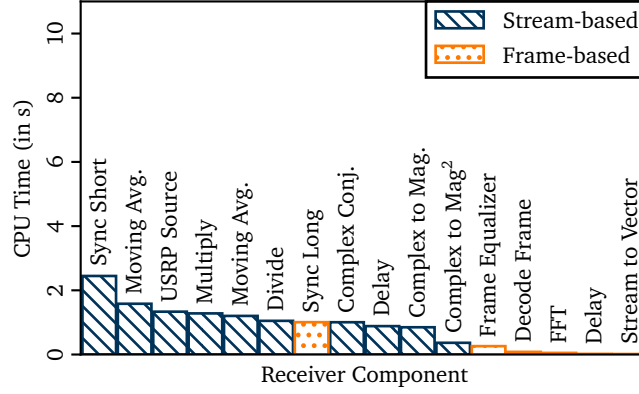
The receiver, in turn, is more complex. While Sora's SoftWiFi showed that an optimized implementation can run in real-time on a normal PC, it was not clear whether it is possible for a general purpose framework that does not rely on hardware or operating-system-specific features. Fortunately, GNU Radio helps us to implement efficient transceivers, as it allows us to profile a flow graph while it is running. To identify bottlenecks, GNU Radio's *performance counters* monitor metrics like CPU time and fill state of the queues per block and exposes them through *control port* [110]. Since there is no easy way to export this data, we created a measurement tool that connects to the transceiver, resets all performance counters, waits for 30 s, and logs the counters to a CSV file.

In the context of this work, we are mainly interested in the CPU time of individual receiver components. Depending on the hardware, GNU Radio supports multiple ways to measure CPU time. We used the more accurate *thread clock*, which measures the time that the thread is scheduled by the operating system. The alternative *monotonic clock* logs the time that a block is scheduled by GNU Radio, i.e., the time it needs to execute the function that processes the samples. This metric is less accurate since it is likely that the thread does not have exclusive access to a CPU during that time. Depending on the load of the system or the amount of I/O operations, there might be a significant difference between the duration of the function call and the actually used CPU time.

To evaluate the performance, we used GCC 5.4.0 to compile GNU Radio and our transceiver in *release mode*. This compiler configuration strips debug symbols and enables all code optimizations. To assert that we use the Single Instruction Multiple Data (SIMD) instructions that provide the best performance for our particular CPU, we used the profiling tool of VOLK, which tests alternate implementations for common signal processing functions and stores the best implementation in a configuration file. With this, we do not have to optimize our transceiver for a particular architecture but can rely on VOLK to choose the optimal instructions for us. The performance tests were conducted on a desktop PC with an Intel i7-7700K CPU and 16 GByte memory. The receiver was configured to use the LS algorithm and operate on channel 178 at 5.89 GHz with a bandwidth of 10 MHz.



(a) Idle channel.

(b) 435 Byte QPSK- $1/2$ frames (73 OFDM data symbols) sent with a rate of 10 frames per second. (Reproduced from [23], © 2018 IEEE.)**Figure 4.13** – CPU utilization of individual signal processing blocks.

To have a baseline, we start with a simple experiment that measures the CPU time of the receiver components when the channel is idle. In this scenario, the receiver does not decode any frames, only false positives of the frame detection algorithm might trigger decoding attempts. The results of the experiment are depicted in Figure 4.13a, which shows the overall CPU time during the 30 s measurement period per block. In this and the following graphs, the blocks are shaded according to their role in the receiver. When considering the computational complexity, we can divide components into roughly two groups.

The first kind of blocks, which we labeled *stream-based*, operates on the whole sample stream. In our receiver, these blocks comprise the SDR source and the blocks that are involved in frame detection. Following the description of our receiver, these blocks calculate the autocorrelation of the signal to recognize the repeating pattern of the short preamble. Since these blocks process all samples, their computational demand is independent of actual frame transmissions. Doubling the bandwidth to

20 MHz IEEE 802.11a/g signals, would, therefore, approximately double their CPU time. The last stream-processing block in the receiver chain is the *Sync Short* block. It monitors the autocorrelation and acts a valve for the subsequent decoding chain. Once the autocorrelation coefficient raises above a configurable threshold, it marks the beginning of a frame and streams samples into the decoding chain. The blocks that work on this subset of samples are involved in decoding the frame. They are labeled *frame-based* in the figures.

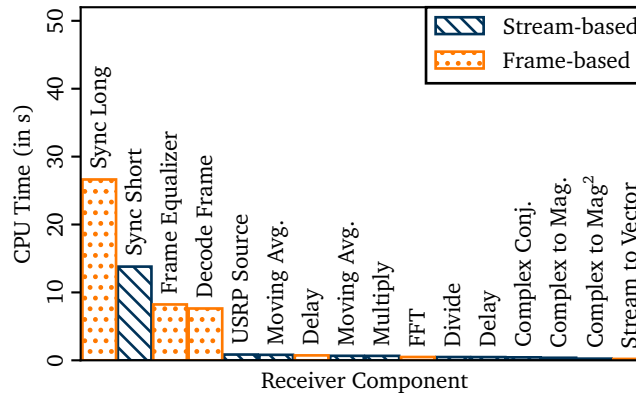
As expected, the frame-based blocks are not triggered when the channel is idle. This shows that the false-positive rate of the frame detection blocks is negligible. A more subtle observation is also worth noting: When running the receiver we do not experience any overruns, i.e., the PC is able to keep up with the incoming sample stream. This is not immediately clear, given the fact that it has to process 10^6 complex floating point numbers per second, and calculating the autocorrelation involves many computationally complex operations like multiplication and division. Furthermore, the fact that the CPU time of all components is below 2 s indicates that we still have a lot of headroom for the decoding process.

To quantify the complexity of the decoding process, we conducted the same measurements but, this time, sending 435 Byte QPSK- $\frac{1}{2}$ frames with a rate of 10 Hz. This corresponds to the highest beacon rate of a vehicle according to the ETSI ITS-G5 standard [73]. The results of these measurements are shown in Figure 4.13b. Compared to the idle channel, we can see that both blocks that are involved in synchronization, i.e., *Sync Short* and *Sync Long*, require more CPU time. However, the increase is only marginal and the overall load of the system is still low.

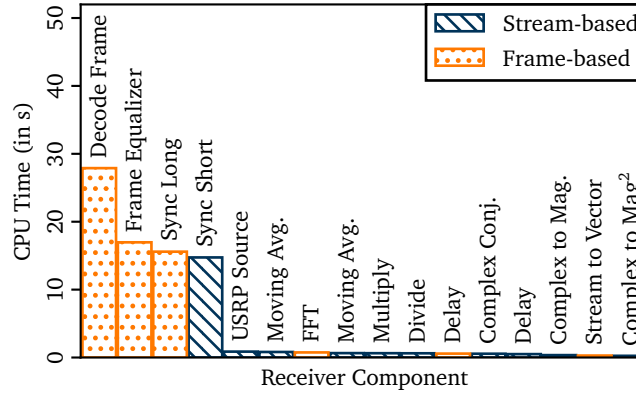
The performance figures of these and the following experiments should be regarded as exemplary and qualitative evaluations. The actual numbers could vary significantly depending on the compiler, compile flags, the GNU Radio version, the operating system, and the CPU. Especially the latter might have a major impact, considering the wide range of clock speeds, hierarchical cache structures, and different instruction sets. Nevertheless, we have observed similar qualitative results on other platforms. In particular, we have tested our implementation on a 2013 MacBook Air with a 1.6 GHz Intel i5 CPU. The receiver worked without dropping samples also on this platform. This shows that even laptops are real-time-capable, highlighting the transceiver's applicability for field tests.

In these low-traffic scenarios, the frame-based blocks use only marginal CPU time. When switching to higher channel loads, the blocks would scale with the number of frames per second. A frame rate of 20 Hz would, therefore, approximately double their CPU time.

Apart from this typical traffic pattern, we also wanted to consider the most challenging case, i.e., a fully saturated channel. To maximize the load of the receiver, we sent 64-QAM- $\frac{3}{4}$ frames, which combines the modulation with the highest order



(a) 36 Byte frames, corresponding to 2 OFDM data symbols.



(b) 1500 Byte frames, corresponding to 56 OFDM data symbols. (Reproduced from [23], © 2018 IEEE.)

Figure 4.14 – Distribution of the computational load with a fully saturated channel of 64-QAM- $3/4$ encoded frames.

with the highest coding rate. This maximizes the load of both the demodulator and the decoder. What we did not know in advance was the overhead caused by the frame-based blocks. In particular, we did not know the relationship between the constant overhead per frame (like synchronization and decoding of the signal field) and the overhead that scales with the size of the frame (like demodulation and decoding). For that reason, we conducted two experiments: One with a frame size of only 36 Byte, which comprises the MAC and LLC header but no data payload, and another one with 1500 Byte frames. In both scenarios, we saturated the channel by sending the frames with an inter-frame space of 58 μ s. This corresponds to the Arbitration Inter-Frame Space (AIFS) of *Voice*, i.e., the Access Category (AC) with the highest priority, and zero backoff slots. To guarantee precise timing, we generated the sample stream with proper spacing in advance and streamed it to the device in one go.

The results for the short frames are shown in Figure 4.14a, and the results for the large frames are shown in Figure 4.14b. In both cases, the load is dominated by the frame-based blocks. However, the load distribution between blocks is different. For the short frames, the load is dominated by the blocks that are involved in synchronization. These blocks comprise Sync Short, Sync Long, and the Frame Equalizer, which decodes the signal field and calculates an initial estimate of the channel.

For the longer frames, the decoder causes the highest load. It required nearly 28 s CPU time in the 30 s measurement interval. That means it required one CPU core nearly exclusively. While this experiment already considers a worst case scenario, it also shows that there is not a lot of headroom left. When aiming for higher bandwidths, this component would, at the moment, present the bottleneck of the receiver. Overall, the results from these measurements showed that the receiver is able to process the signal reliably without dropping samples, highlighting its real-time capability even for fully saturated channels.

4.4 Time-Critical Functionality

When developing our IEEE 802.11a/g/p transceiver, the PHY was clearly the main focus. Still, after its implementation and validation, we wanted to further explore the potential and the limitations of the architecture. If we consider time-critical functionalities like channel access or Automatic Gain Control (AGC), the relevant timings are in the microsecond scale. Our GPP-based SDR architecture, however, introduces latencies in the millisecond scale [101]. These latencies stem from pre-processing the samples on the radio, streaming them to the PC, and processing them on a non-real-time operating system. Implementing standard compliant channel access or AGC on the PC is, therefore, not possible.

4.4.1 Channel Access

One way to implement channel access for GPP-based SDRs is to relax the timings by increasing the inter-frame space. The increase, however, has to be significant, spanning about three orders of magnitude [121], [122]. The resulting system is, of course, no longer standard compliant and makes less efficient use of the channel. Yet, it is a simple method to implement WLAN-like channel access on the PC.

The only option to implement standard compliant channel access for our SDR architecture is to extend the FPGA. The idea is to move time-critical functionality in hardware to achieve low latency and deterministic timing, while leaving the non-time-critical parts in software, maintaining the benefits of a GPP-based PHY implementation. This *split-functionality* approach was first explored by Nychis et al.

[99]. Their prototype featured a GNU Radio implementation of a Carrier Sense Multiple Access (CSMA) MAC that realized carrier sensing, backoff processing, and dependent packet processing inside the FPGA of an SDR from Ettus Research. More recently, Di Francesco et al. [123] presented a similar architecture for an embedded SDR. However, both works do not focus on standard compliance but study the feasibility of the split-functionality approach on the respective platforms. We use the same approach to realize standard compliant channel access for IEEE 802.11 broadcast transmissions.

4.4.1.1 Concept and Implementation

To implement CSMA, we had to extend all components of the system, i.e., our transceiver, the USRP Hardware Driver (UHD), which interfaces the Ettus Research N210, the FPGA image, and the firmware of the ZPU soft-core running on the FPGA. An overview of the system is shown in Figure 4.15. We started by creating a GNU Radio block with four inputs for the individual ACs. This block tags data packets with CSMA metadata, i.e., AIFS and the randomly chosen number of backoff and post-TX backoff slots. All random numbers are, therefore, generated on the host. The tags are propagated through the transmit chain until they reach the *USRP Sink* block, which orchestrates the SDR through the UHD. Here, the CSMA parameters

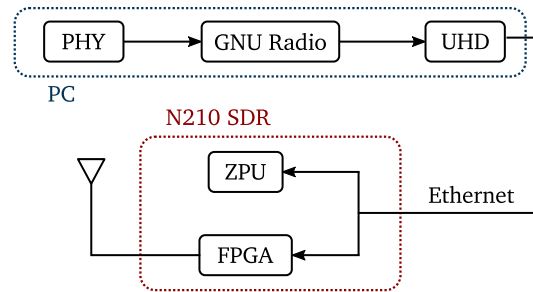


Figure 4.15 – Conceptual overview of our CSMA implementation. (Reproduced from [28] with permission.)

Table 4.5 – Most relevant components of our setup. (Reproduced from [28] with permission.)

Component	Type
GNU Radio	Version 3.7
UHD	Version 003 006 001
SDR	Ettus Research N210
Daughterboard	XCVR2450
Xilinx ISE	Version 12.3

are extracted from the tags and added as metadata to the samples before they are sent over Ethernet to the FPGA.

On the FPGA, the parameters are used to configure the CSMA state machine, and the samples are stored in memory until the state machine triggers their transmission. Note that, currently, we maintain a single queue for all frames as opposed to one queue per AC. We refrained from adding multi-queue support, since the available memory on the FPGA is too small to instantiate multiple queues. Finally, we extended the firmware of the ZPU soft-core to provide a control interface to set parameters that are not frame-specific, like slot time and the Clear Channel Assessment (CCA) threshold.

While we base our implementation on the most recent versions of GNU Radio and the UHD, we used an older version of the Xilinx ISE to compile the FPGA image, since we experienced timing problems when using more recent versions. Table 4.5 summarizes the most important components of our setup.

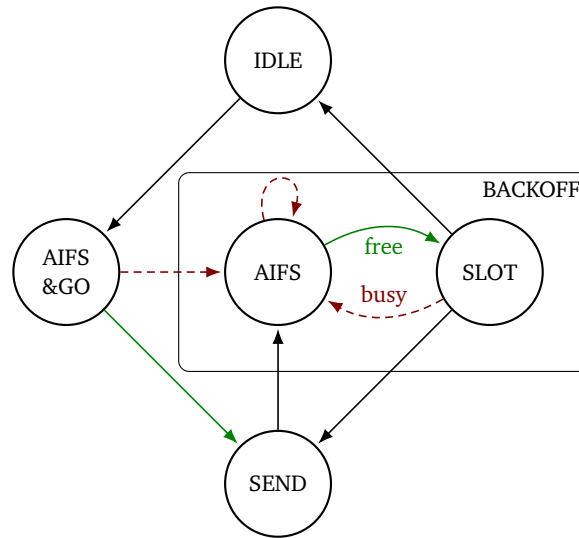
Clear Channel Assessment

Since virtual carrier sensing is not relevant for broadcast transmissions, we only consider physical carrier sensing. Here, we limited our implementation to energy detection only, since preamble detection requires us to demodulate and decode at least the signal field on the FPGA. This would require additional functionality on the FPGA, including frame detection, synchronization, demodulation, and decoding. Implementing these PHY algorithms in hardware would contradict our GPP-based approach and also exceed the resources of the Ettus N210.

For energy detection, we pipe all samples from the receive chain to a custom Verilog module and calculate the power per sample. The power values are averaged over a window of configurable size and compared to a threshold. If the average power exceeds the threshold, we report the channel as busy to the CSMA state machine. The threshold can be configured over the control channel implemented on the ZPU soft-core. For our tests and evaluations, we used a moving average of eight samples, which corresponds to a time window of $0.8\ \mu\text{s}$ at 10 MHz.

CSMA State Machine

To send a frame, its samples are transferred to the SDR and buffered in memory until the CSMA state machine triggers their transmission. Each frame is annotated with its AIFS duration and random variables for the backoff and post-TX backoff slots. An overview of the state machine is depicted in Figure 4.16. It starts in the IDLE state and remains there until a frame is loaded on the SDR. Once a frame is buffered in the SDR, it switches to the AIFS&GO state. If the medium is free for an AIFS while in the AIFS&GO state, we send the frame immediately. Otherwise, we



switch to the AIFS state and start the normal backoff procedure. We stay in AIFS until the medium is free for an AIFS without interruption. When this is the case, we switch to the SLOT state and start counting down backoff slots. If the medium turns busy while in the SLOT state, we reset the current slot timer and switch back to AIFS. Otherwise, we send the frame after waiting for the configured number of backoff slots. A frame transmission is triggered by entering the SEND state, where we also remain during the transmission. Once the frame is transmitted, we enter the post-TX backoff, which does not differ from the normal backoff logic. Since we enter the post-TX backoff even if no frame is buffered, we might switch from the SLOT state back to the IDLE state.

4.4.1.2 Evaluation

Verification and evaluation of our implementation have been performed in three steps. We verify

- the CCA mechanism,
- the inter-arrival time of frames from a fully saturated channel,
- and interoperability with commercial devices.

In general, the Enhanced Distributed Channel Access (EDCA) parameters, i.e., AIFS durations and the maximum number backoff slots, are configured by higher layers. However, the IEEE 802.11p as well as IEEE 1609 WAVE [68] and ETSI ITS-G5 [124] agree on the parameters listed in Table 4.6. Furthermore, these standards

Table 4.6 – Channel access parameters for the different access categories [59, Table 9-138]. (Reproduced from [23], © 2018 IEEE.)

AC	$CW_{\min}$	AIFSN	AIFS
Background	$aCW_{\min} = 15$	9	149 μs
Best Effort	$aCW_{\min} = 15$	6	110 μs
Video	$(aCW_{\min} + 1)/2 - 1 = 7$	3	71 μs
Voice	$(aCW_{\min} + 1)/4 - 1 = 3$	2	58 μs

set the Transmission Opportunity (TXOP) of all ACs to zero, effectively disabling the mechanism and falling back to packet-based fairness. We employ this parameter set for all measurements.

Energy Threshold for CCA

To implement energy detection, we have to set a threshold that defines the power level above which the channel is sensed busy. In our case, the threshold does not define an absolute level but is expressed in the raw values that are output by the Analog-to-Digital Converter (ADC). Calibration with a reference device in order to set the threshold to the power levels defined in the standard is possible but was not necessary for the following experiments. Instead, we manually set the threshold between the power level of the noise floor and a frame transmission.

To test the timing accuracy of our implementation, we used another SDR to monitor the channel. We synchronized the clock of the monitoring device with the clock of the device that accesses the channel. This way we prevented a relative clock drift and made sure that the sampling frequencies are in sync. Furthermore, we set all backoffs to zero so that the channel is accessed deterministically after an AIFS.

In the first experiment, we used an SDR to occupy the channel and block frame transmissions. The results of this experiment are shown in Figure 4.17. During the first 100 μs , the channel is blocked with noise. After that, we see that the frame transmission is delayed even after the channel turns free. In this case, we configured the AIFS to 58 μs , i.e., the inter-frame space of the *Voice* AC and measured a value of 59.8 μs . The additional 1.8 μs can be explained by the 1 μs RX-TX turn around time of the MAX2829 transceiver IC [125], which is used on our RF front end, and the 0.8 μs averaging window of the energy detector. Moreover, this complies with the upper limit of 2 μs defined in the standard.

We made similar measurements also with different inter-frame spaces to assure that the timing does not drift for larger values (which it actually did with a more recent version of the Xilinx ISE) and observed similar results. The constant additional delay of 1.8 μs could be compensated by subtracting it from the AIFS, resulting in a more precise timing. We, however, did not implement this, since the CCA delay

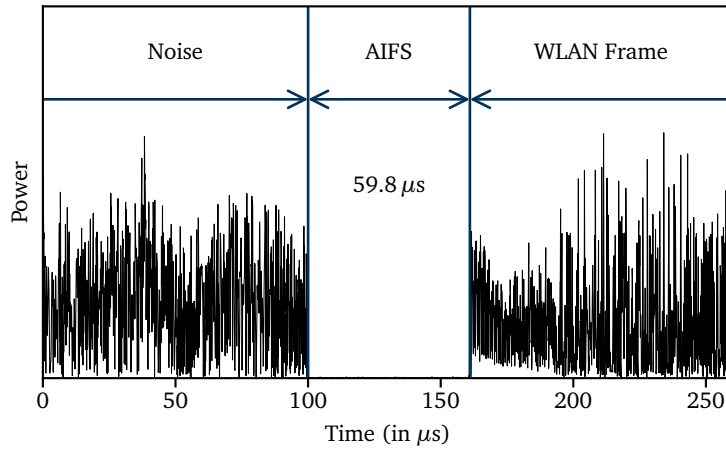


Figure 4.17 – Power measurements to verify AIFS timing and to determine channel access delay. (Reproduced from [23], © 2018 IEEE.)

and the RX-TX turnaround time are already considered in the calculation of the slot time, so that also the current version is within the bounds defined by the standard.

Inter-arrival Time

In the next step, we tested basic CSMA functionality. A convenient way to do this is to saturate the channel and measure the inter-arrival time of frames. If a single device saturates the channel, the CSMA mechanism is as follows: the device sends a frame, enters the post-TX backoff, and sends the next frame immediately after the post-TX backoff. The post-TX backoff spans an AIFS plus a random number of backoff slots, uniformly distributed between 0 and CW_{min} . The inter-arrival times should, therefore, be discrete and uniformly distributed in the CW.

Like in previous experiments, we used a Linux PC with a Unex DCMA-86P2. To measure the inter-arrival time, we extended the receive interrupt handler of the card with logging functionality. We configured the SDR to saturate the channel by sending frames as fast as possible and configured different ACs. The distribution of the inter-arrival times of the *Voice* and *Video* ACs can be seen in Figure 4.18. Each histogram is based on more than 30 000 frames. The dashed lines indicate the slot boundaries where the transmissions are expected. Note that in this and the following histogram, we added a constant offset of 3 μs when plotting the slot boundaries. This honors RX-TX turn around time and a slight offset that seems to be introduced by the clock resolution of the Linux PC.

We can also see that the card waits for the mandatory AIFS duration plus a random number of backoff slots. This verifies the slot time, the AIFS duration, and the CW_{min} setting. Furthermore, it shows that the backoff slots are distributed approximately

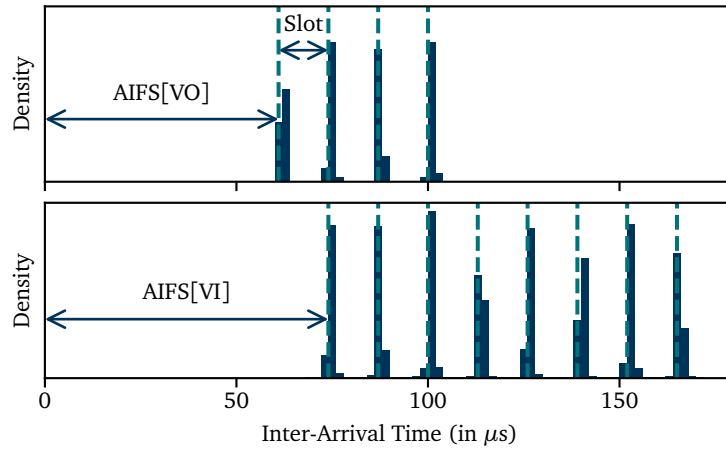


Figure 4.18 – Distribution of the inter-arrival time when using the ACs for Voice and Video. (Reproduced from [23], © 2018 IEEE.)

uniform, as expected. In addition to that, the histograms give a good impression about the accuracy of the implementation. We repeated the measurements for the other ACs and observed similar results (data not shown).

Interoperability

In a final experiment, we verified interoperability in terms of fairness with IEEE 802.11p prototypes. We started with the Cohda Wireless MK2 using firmware version 4.0.14615. To assert that we configured the device correctly, we conducted the same measurements as for the SDR. The results are plotted in Figure 4.19 (top plot).

Clearly, the distribution does not correspond with the expected result. It turned out that the Cohda Wireless MK2 does not implement the post-TX correctly and sends consecutive frames deterministically after an AIFS. We configured different ACs and observed different AIFS, which showed that we used the right Quality of Service (QoS) queues. The post-TX backoff, however, did not work for any AC. The technical support of Cohda Wireless meanwhile confirmed the bug. Since the post-TX backoff is a crucial part of the CSMA mechanism, especially for saturated channels, we had to exclude the MK2 from the experiments.

Instead, we switched to the Unex DCMA-86P2 that we already used in our initial measurements. However, physical connectivity is not enough for the fairness test, we also need a correctly parameterized and standard compliant MAC. This required further modifications of the *ath5k* driver. In particular, we had to instantiate and configure the QoS queues and set the slot time and the Short Interframe Space (SIFS). With these changes the QoS queues are enabled, but all packets go to the default queue. For setting the AC per packet, we used the Radiotap header. When

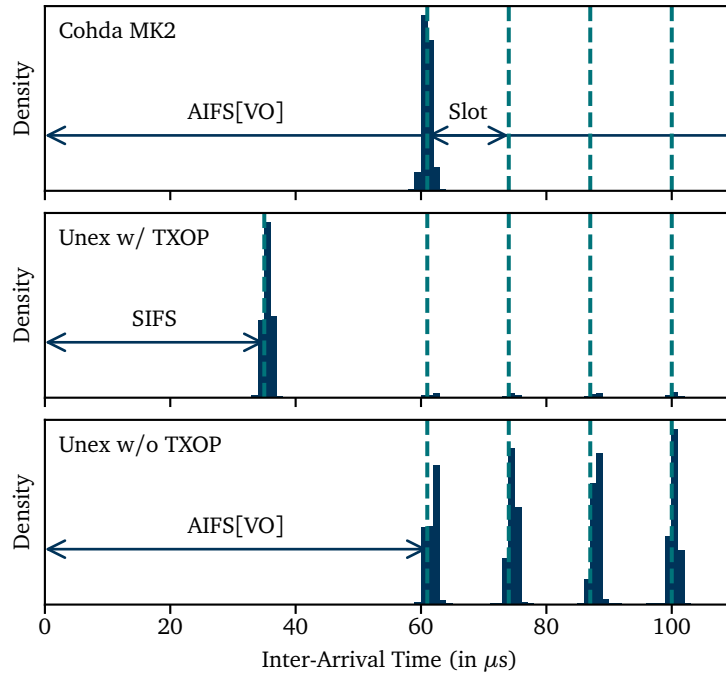


Figure 4.19 – Inter-arrival time of a Cohda Wireless MK2 (top) and a Unex card with (middle) and without (bottom) TXOP set. (Reproduced from [28] with permission.)

the WLAN card operates in monitor mode, the Radiotap header allows to annotate metadata (like the MCS or the signal power) to a frame. We exploited a field that is currently not used on TX side to signal the AC to the kernel.

Since we made major modifications to the driver, we validated the changes by measuring the inter-arrival time. At first, we observed the distribution shown in the middle of Figure 4.19. The results are not as expected but indicate that the driver uses TXOPs by default. That means that a device that wins contention will use the channel for the whole TXOP, during which it sends frames spaced only by a SIFS. After the TXOP, the device enters a post-TX backoff and uses the expected inter-frame space. This is indicated by the small bars at the slot boundaries. The ratio between the packets that are sent after a SIFS and after a post-TX backoff depends on the frame size and the duration of the TXOP. In this case, the TXOP was long enough to cover several frames, so that most frames are spaced only by a SIFS. Following these experiments, we explicitly disabled all TXOPs and repeated the measurements. This time we observed the expected timing distribution depicted in the plot at the bottom of Figure 4.19. We also tested the other ACs and observed the correct distribution. With these experiments, we made sure that the correct EDCA queues are used and that the AIFS, SIFS, slot time, and $CW_{\min}$ are configured correctly.

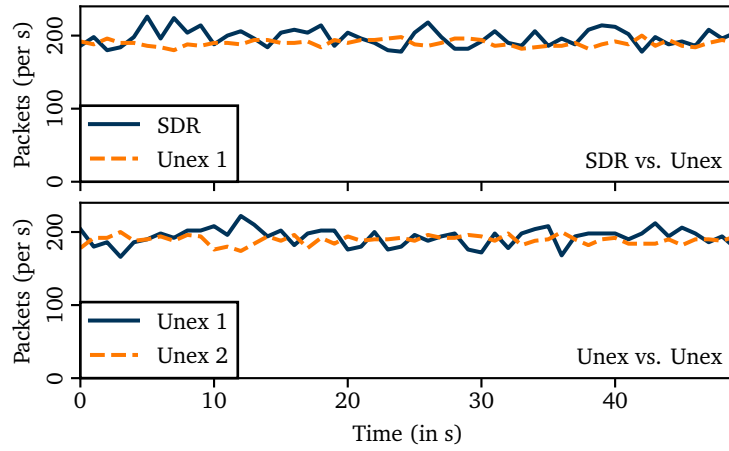


Figure 4.20 – Interoperability test between an SDR and a Unex card (top) and two Unex cards (bottom), showing that each device gets a fair share of the channel. (Reproduced from [28] with permission.)

Using the validated Unex devices, we are able to conduct fairness measurements as a final step towards ensuring the correctness of the implementation. We use a PC with a Unex card to log all frames and saturate the channel, in one experiment, with an SDR and a Unex card and, in another experiment, with two Unex cards. The average throughput over time is plotted in Figure 4.20. In both cases, we can observe a fair share of the channel and a similar variation of the throughput. This shows the interoperability of our implementation with commercial devices.

To summarize, we presented an approach to extend a GPP-based SDR with standard compliant channel access for broadcast transmissions. Our implementation follows the split-functionality approach, where only time critical components, like carrier sensing and CSMA logic, are implemented in hardware. This architecture preserves the flexibility of a software implementation but, at the same time, allows for standard compliant channel access for broadcast transmissions. We validated our implementation through extensive timing measurements and interoperability tests with commercial devices. The results highlight the feasibility of the approach and show that our implementation is able to meet all timing requirements of the standard.

4.4.2 Automatic Gain Control

When we leave the lab to conduct field tests, we face another challenge as the receiver has to process frames with very different power levels. This effect is particularly pronounced in VANETs where the maximum transmit power is higher than in normal WLANs. This was not an issue in our previous experiments, since relatively static channels allowed us to configure the gain manually. However, in a more realistic

scenario, the receiver has to process frames from multiple mobile communication partners connected through fast-fading multi-path channels. As we will show in our experiments, there is no fixed gain setting that works well for all input power levels. Either weak frames do not take full advantage of the dynamic range of the ADC, resulting in quantization noise, or high power frames drive the receiver into saturation, distorting the signal. A static setting would, therefore, lead to performance issues, which can be overcome by adjusting the gain for every received frame. 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 non-linearities. For WLAN, we have only a short time to adjust the gain. Considering a 10 MHz signal, there are only the 16 μ s of the short preamble to detect the frame, estimate the power level, and set a proper receive gain. Since the long preamble is used for initial channel estimation, also the transients introduced by the gain change should be finished before the long preamble starts.

Given these requirements, an integrated WLAN card implements AGC directly on the RF transceiver chip. A general purpose SDR, however, cannot provide AGC out of the box, since the requirements are highly application-specific. With Long-Term Evolution (LTE), for example, the mobile handset is connected to a single base station, allowing the device to adjust the gain gradually over time. Apart from that, the ideal gain setting also depends on signal parameters like the Peak-to-Average Power Ratio (PAPR).

With regard to our transceiver, the tough timing constraints require an FPGA implementation, since the PC would be orders of magnitudes too slow [99]. Like in our CSMA implementation, we extended the Open Source firmware of the Ettus Research N210 using Xilinx ISE 12.3. (This particular version of the development framework worked well in our tests, while more recent versions led to timing errors.) Since we control the gain, we also have to interact with the RF transceiver, which makes the code also specific to a particular daughterboard. We used a XCVR2450 daughterboard, which is equipped with a MAX2829 RF transceiver. This transceiver is designed specifically for WLAN applications [125]. It allows operation in the 2.4 GHz and 5 GHz band and supports two alternate modes to set the receive gain. By default, the N210 uses the SPI interface. This serial communication, however, introduces additional delay, which might be problematic given our tough timing constraints. We, therefore, set the gain directly through I/O pins. With the MAX2829, readjusting the receive gain leads to transients with considerable distortions, but it resettles after only 40 ns.

On the FPGA, there are two possible locations in the signal processing chain where we could hook in AGC: Either directly after the ADC or after the signal is channelized and downsampled. An implementation directly after the ADC has the advantage that it minimizes delay but also bears problems: Since RF transceivers

can only tune to fixed frequencies, the N210 has to perform a shift in digital domain. Directly after the ADC, the signal is, therefore, not necessarily centered around the carrier frequency, making it harder to exploit the autocorrelation of the preamble for frame detection.

For that reason, we implemented AGC using the channelized samples and, therefore, operate on the same sample stream that is also sent to the PC. For AGC, we have to detect a frame, estimate its power level, and adjust the gain to bring it as close as possible to a desired reference power level. We determined the reference level empirically with a comprehensive set of experiments in which we measure the frame delivery ratio of different combinations of frame input power levels and receive gain settings. With these measurements, we are able to create a lookup table that maps the input power level to a gain setting that works well for our platform. Analytical calculations or simulations would have to be adapted to reflect the limitations of the hardware and would be hard to parameterize in order to realistically capture the characteristics of the transceiver chip. Furthermore, the high PAPR of the signal makes it at least not straightforward to map a given input power level to an optimal receive gain setting.

The gain of the MAX2829 transceiver can be distributed across two amplifiers, a Low Noise Amplifier (LNA) close to the antenna and a Variable Gain Amplifier (VGA) at a later stage of the signal processing chain. While the LNA provides a coarse resolution with only three steps for gain values of 0 dB, 15 dB, and 30 dB, the VGA offers a more finer grained resolution, covering 62 dB in steps of 2 dB. Both stages combined provide a gain range of 92 dB. Since the gain range of the VGA is larger than the steps of the LNA, most gain levels could be configured with different gain distributions between LNA and VGA. In those cases, we set the LNA to the highest possible value to minimize the overall noise figure of the receiver. When no frame is detected the AGC will switch to an intermediate level of 46 dB.

On the FPGA, we use two methods for frame detection: The first uses the autocorrelation of the short preamble, similar to the PC implementation. The second triggers frame detection if the input power level exceeds a predefined threshold. Using the second method, we can also detect high-power frames that overdrive the ADCs, distorting the cyclic pattern of the preamble.

To highlight the need for AGC and to show its performance improvements over fixed gain configurations, we used a setup with two WLAN receivers. One was placed nearby the SDR and was transmitting with high power (15 dBm). Another one was placed several meters away and was transmitting with low power (−15 dBm). In a first experiment, we plotted the signal power in time domain to measure the time that we need to reconfigure the gain. This delay is interesting, since it was unclear whether the processing delay of the N210 is low enough for our application, i.e., whether it is possible to implement AGC based on the channelized samples.

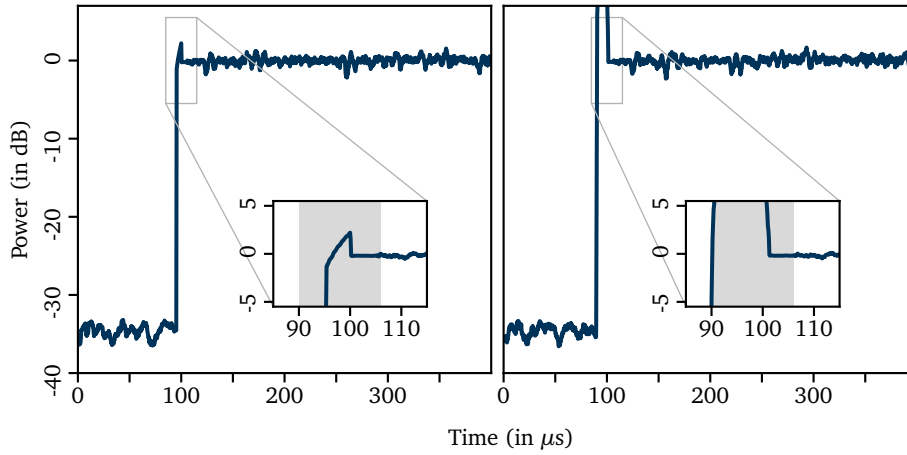


Figure 4.21 – Power over time for a low-power frame that got amplified by AGC (left) and a high-power frame that got attenuated (right). The power is normalized to the reference power level that the AGC tries to adjust to. (Reproduced from [23], © 2018 IEEE.)

Figure 4.21 plots the results for the low-power transmitter (on the left) and for the high power transmitter (on the right). The power is normalized to the reference power level that the AGC tries to adjust to. Two observations are interesting in this context: First, we see that both signals are adjusted to a very similar power level, demonstrating basic functionality of the AGC, i.e., that the low power signal is amplified, while the high power signal is attenuated. Second, the time to detect the frame, reconfigure the gain, and stabilize after a transient phase is shorter than the short training sequence of the frame. This ensures that the frame is not corrupted by changing power levels, proving the feasibility of our approach. The graphs show IEEE 802.11p frames with a short preamble of $16\mu\text{s}$. As we can see in the insets, the signal stabilizes on the target power level after about $12\mu\text{s}$. Since the delay is mainly caused by averaging of the power level, our AGC implementation also works with higher sample rates. It even supports IEEE 802.11a/g.

In a second experiment, we highlight the need for AGC by showing performance issues of fixed gain configurations. We use the same setup with a high and low power transmitter and measure the delivery ratio of 128 Byte BPSK- $\frac{1}{2}$ frames. For both configurations, we compare AGC to the whole range of fixed gain settings. Figure 4.22a shows the frame delivery ratio of the low power transmitter. As expected, we can see that the frames cannot be decoded with low receive gains, as the low gain in combination with the low power level results in a high relative quantization noise. The opposite happens for high-power signals (see Figure 4.22b). Here, frames can be received with low receive gains, while higher receive gains drive the ADCs into saturation, resulting in clipping noise. For both configurations, we also measured

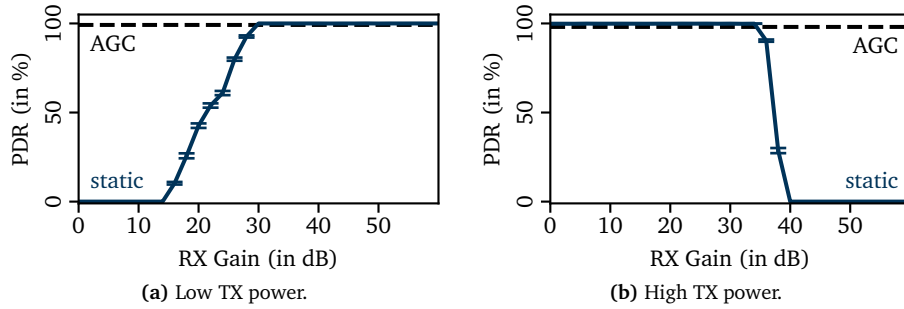


Figure 4.22 – Frame delivery ratio with fixed receive gains and AGC. (Reproduced from [26], © 2015 IEEE.)

the delivery ratio with AGC enabled and were able to receive close to all frames. Being able to correctly tune the input amplifier even in dynamic scenarios turns our SDR solution into a viable measurement equipment also for outdoor experiments.

4.5 Conclusion

By demonstrating the feasibility of Automatic Gain Control (AGC), we conclude the development, verification, and evaluation of our IEEE 802.11a/g/p transceiver. It presents the first physical layer (PHY) implementation for a general purpose Software Defined Radio (SDR) framework that is functionally complete, as it supports all frame sizes and Modulation and Coding Schemes (MCSs). Our contribution is not the development of new algorithms but the realization of the receiver based on a General Purpose Processor (GPP) real-time signal processing framework. We verified the implementation through simulations and interoperability test with both commercial Wireless LAN (WLAN) adapters and IEEE 802.11p prototypes. Since real-time capabilities are essential, we studied the computational complexity of individual receiver components at different channel loads. By considering fully saturated channels, we showed that our transceiver manages to process all samples even in the most challenging scenarios. While the PHY was clearly the main focus of our work, we also explored the possibility to implement time-critical tasks like channel access and AGC. Following the *split-functionality* approach, we extended the Field-Programmable Gate Array (FPGA) with timing-critical functionality, while keeping the whole PHY on the PC. This allowed us to add functionality without giving up any of the benefits of a GPP-based implementation.

Our goal was not to create an, in all aspects, optimal transceiver but to provide a solid base that can be adapted for all kinds of studies. We believe that its ability to experiment with the PHY in combination with standard-compliant channel access for broadcast transmissions makes the transceiver particularly interesting for Vehicular

Ad Hoc Networks (VANETs). To foster the use of SDR in the vehicular networking community and to allow reproduction of our results, we released the software under an Open Source license. Our work can be seen as a first step towards a fully Open Source IEEE 1609 Wireless Access in Vehicular Environment (WAVE) or ETSI ITS-G5 stack, which was identified by the community as a requirement for reproducible field tests. To demonstrate its applications and to highlight its ability to address open research questions in VANETs, we present exemplary studies that were enabled by our prototype.

Chapter 5

WLAN for Vehicular Networking

5.1	Field Tests	86
5.1.1	Performance of IEEE 802.11p Prototypes	87
5.1.2	Trace-Driven Performance Evaluation	90
5.2	Impact of Intra-Technology Interference	97
5.2.1	Simulative Evaluation	100
5.2.2	Experimental Evaluation	101
5.2.3	Off-the-Shelf Testbed	103
5.2.4	Conclusion	105
5.3	The Scrambler Attack	106
5.3.1	Related Work	107
5.3.2	Description of the Attack Vector	108
5.3.3	Applicability to Current Hardware	109
5.3.4	Implications for Privacy	110
5.3.5	Evaluation of Impact	110
5.3.6	Countermeasures	115
5.3.7	Conclusion	116

In this chapter, we use our Software Defined Radio (SDR)-based IEEE 802.11 a/g/p transceiver as a tool to address selected research questions in vehicular networks. To highlight our prototype's applicability for a wide range of use-cases, we present three different studies that, apart from the relevance of the results themselves, show the flexibility of our tool.

At first, we use our transceiver in two field tests to study the performance of IEEE 802.11p under realistic conditions. Given the fact that Wireless LAN (WLAN) was designed for relatively static indoor setups, its applicability for vehicular applications is of great interest. In separate field tests, we compare the performance of different IEEE 802.11p prototypes and the performance of different receive algorithms. After the field tests, we study the impact of noise and intra-technology interference on IEEE 802.11p. With this study, we validate a physical layer (PHY) simulation model that is used by many popular network simulators. Here, we settle a current dispute in the community as to whether noise of interference has a more detrimental impact on the PHY. Apart from validating the model, the study also shows the most interesting use-cases of our transceiver. Following the research process described in the introduction, we progress from simulations to over-the-air experiments, which, together, provide a coherent picture, increasing the confidence in our results.

Finally, we present an application that goes beyond signal processing and PHY performance. Using internal data from the decoding process, we develop a novel attack on the location privacy of Vehicular Ad Hoc Networks (VANETs) and evaluate its impact through network simulations. While the data used in the attack is available in every transceiver, it is not exposed by normal WLAN cards. This is not an issue with SDRs, as they allow us to tap into data from all processes and exploit weaknesses in the pseudo-random number generators of typical WLAN chips to track vehicles, circumventing the privacy protection mechanisms of VANET protocol stacks.

The chapter is based on the following publications:

- B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "Performance Assessment of IEEE 802.11p with an Open Source SDR-based Prototype," *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, pp. 1162–1175, May 2018. DOI: 10.1109/TMC.2017.2751474, © 2018 IEEE.
- 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. DOI: 10.1145/2980100.2980104.
- B. Bloessl, F. Klingler, F. Missbrenner, and C. Sommer, "A Systematic Study on the Impact of Noise and OFDM Interference on IEEE 802.11p," in *9th IEEE*

Vehicular Networking Conference (VNC 2017), Torino, Italy: IEEE, Nov. 2017, pp. 287–290. DOI: 10.1109/VNC.2017.8275633, © 2017 IEEE.

- 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. DOI: 10.1109/ICCNC.2015.7069376, © 2015 IEEE.

5.1 Field Tests

After many years of research, VANETs have become a major technology, and the corresponding communications standards are well-advanced. At this stage of the development process, it is unlikely that the PHY will undergo significant changes, so that, at least, the first generation of VANETs will be based on IEEE 802.11p. Before the higher-layer standards are finalized, and the technology is rolled-out on a large scale, field tests become increasingly relevant. In the worst case, they allow us to identify weaknesses in system design that might have been overlooked in simulations. In the best case, they serve as the ultimate proof-of-concept that demonstrates the readiness of the technology to car manufacturers, regulatory bodies, and the general public.

Apart from demonstrating the feasibility of the technology, field tests also provide quantitative performance results. With a final specification for the PHY, we can study IEEE 802.11p from different angles, addressing questions like:

- What is the performance of a given transceiver in a realistic environment?
- What receiver complexity is necessary to reach a given minimal performance?
- What are the practical limitations of the technology?

In the first field tests, we compare our implementation to other IEEE 802.11p prototypes to establish it as a credible tool among prototypes that are based on commercial WLAN cards [83]–[86] and dedicated IEEE 802.11p transceivers from Cohda Wireless [79], NEC [80], [81], and Denso [8], [82]. Apart from showing our transceiver’s applicability for field tests, the experiment provides exemplary results for the achievable performance in a realistic environment. Furthermore, the comparison of different prototypes gives an indication about the potential performance gains of better hardware and more specialized implementations.



Figure 5.1 – Setup of our field test. (Reproduced from [23], © 2018 IEEE.)

5.1.1 Performance of IEEE 802.11p Prototypes

The field test is conducted near Paderborn, Germany. We equip two cars with IEEE 802.11p prototypes, as shown in Figure 5.1. As transmitters, we use a Cohda Wireless MK5 and our SDR implementation with an Ettus N210 using an XCVR2450 daughterboard. In the SDR, we used the Least Squares (LS) equalizer, which was, at the time of the field tests, the only implemented channel estimation algorithm. While both the MK5 and the SDR are not calibrated measurement devices, we set their output power to about 10 dBm. For the MK5, this can be configured. For the SDR, we adjust the gain and the signal amplitude, assuming a maximum output power of 20 dBm as listed in the datasheet. Both devices are controlled by a laptop, which alternates between sending frames from the MK5 and the SDR. We leave enough timing margins to make sure that there are no collisions. Like in the Additive White Gaussian Noise (AWGN) simulations, we use 435 Byte frames to resemble the size of a typical Cooperative Awareness Message (CAM). We set the Modulation and Coding Scheme (MCS) to Quadrature Phase-Shift Keying (QPSK)- $\frac{1}{2}$, since we do not want to use the simplest scheme but show that the receiver is able to deal with higher order modulations. Furthermore, QPSK- $\frac{1}{2}$ was found to provide a good compromise between throughput and robustness, and was, therefore, selected as the default MCS for periodic awareness messages in ETSI ITS-G5 [126].

On the receive side, we use another MK5, another SDR, and a Unex DCMA-86P2. The receivers use 9 dBi dipole antennas mounted at the center of the roof (see Figure 5.1). Note that each receiver uses its own antenna and, therefore, experiences independent fast-fading effects. For that reason, we can compare average reception rates but not the success of individual transmissions. Both cars log their positions every 0.5 s with a u-blox NEO-7N, a high precision GPS receiver. The cars' positions at frame transmissions are later interpolated linearly 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.

5.1.1.1 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 reference receiver, we plot the delivery ratio of frames generated by the SDR and the MK5 in Figure 5.2. 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 5.2 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.

While the results are promising for our transceiver, the measurements should not be mistaken for a performance comparison between the SDR and the MK5. This would require more precise power calibration and measurements with the exact same antenna placement, as the antenna's position on the roof can have a significant impact [13], [127]. Apart from performance characteristics, the plot shows that the environment naturally led to packet loss. While this is expected in realistic conditions, it also highlights that we did not choose overly simplistic scenarios.

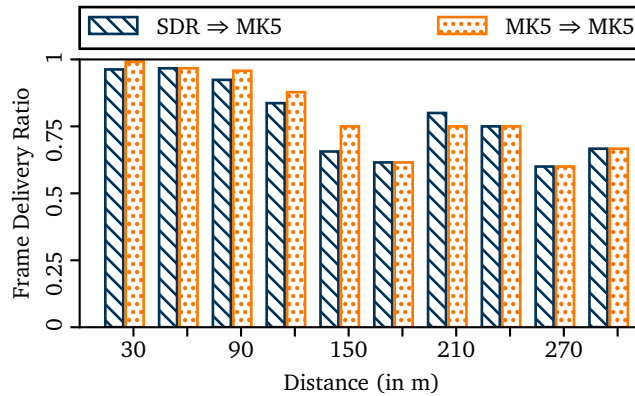


Figure 5.2 – Delivery ratio of frames sent from an MK5 and an SDR, using another MK5 as reference receiver. (Reproduced from [23], © 2018 IEEE.)

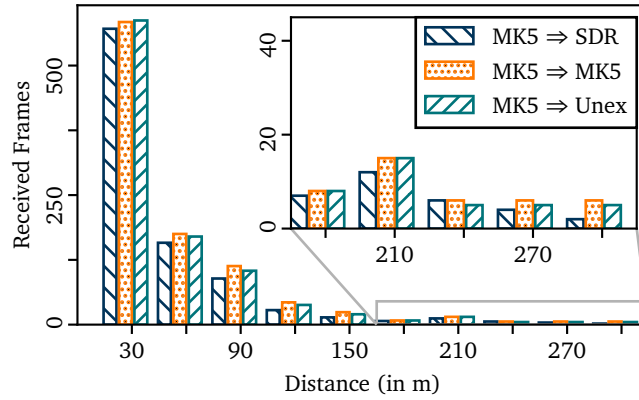


Figure 5.3 – Received IEEE 802.11p frames per device. (Reproduced from [23], © 2018 IEEE.)

5.1.1.2 Receive Performance

Having tested our transceiver’s ability to send standard compliant frames, we compare the receive performance of the different prototypes. Using the MK5 as sender, we plot the number of frames received by the SDR, the MK5, and the Unex card in Figure 5.3. 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 with independent experiments that compared a predecessor of the MK5 with off-the-shelf cards [79].

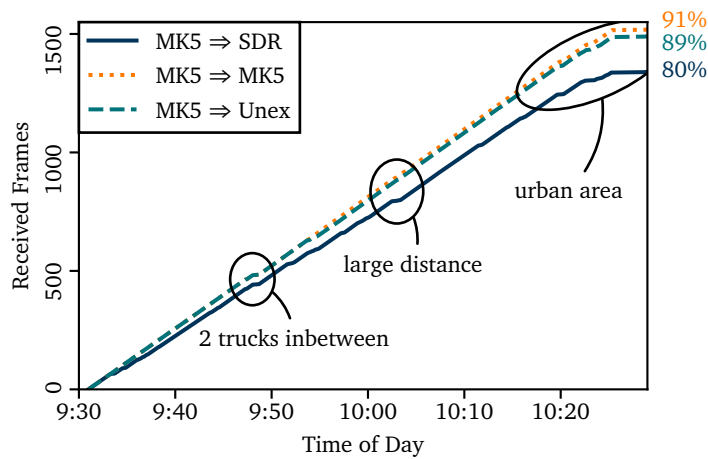


Figure 5.4 – Cumulative number of packets received over time. (Reproduced from [23], © 2018 IEEE.)

Another view on the same data is shown in Figure 5.4, 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 or 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. Cohda Wireless, for example, uses a sophisticated two-stage decoder that is optimized for high mobility [10], [128]. 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 [13], [66].

Overall, the field test shows that our transceiver does not only provide reasonable performance in simulations and measurements in the lab but also in a realistic environment on the street, which highlights our transceiver's applicability for all steps of the research process.

5.1.2 Trace-Driven Performance Evaluation

The first field test already gave a hint about the shortcomings of the LS equalizer. In the second field test, we have a closer look at this receiver component and study the performance of different channel estimation algorithms. When WLAN was proposed as a base for VANETs, the research community was concerned whether the PHY could work in a vehicular environment. It was widely agreed that, at least, simple receive algorithms will run into problems given the high mobility of the network [12], [13], [66]. There are two ways to deal with that problem: One is to adapt the PHY using, for example, additional pilot symbols [11] or differential encoding [15]. The other is to employ more advanced receive algorithms that can cope with the high dynamics [14], [65].

Following the latter line of research, we conduct a field test to compare the performance of selected IEEE 802.11p channel estimation algorithms. In the experiment, we use one car as a transmitter that sends frames with different sizes and MCSs at a high rate. Another car acts as a dedicated receiver. Here, we do not immediately decode the frames but use an Ettus Research N210 SDR to record the raw signal (i.e., the complex baseband samples). This allows us to post-process the recorded

samples with different algorithms and directly compare their performance. In the context of this experiment, we extended our transceiver with four state-of-the-art channel estimation algorithms and used them to decode the signal trace.

5.1.2.1 Channel Estimation

The LS equalizer is a simple algorithm that is often used as a baseline and cited to be a candidate for hardware implementations [12], [13]. In a nutshell, it uses the long training sequence of IEEE 802.11p as block pilots to estimate the channel. Denoting the estimate of a value X as $\hat{X}$, we calculate the channel H at subcarrier k as

$$\hat{H}(k) = \frac{Y_1(k) + Y_2(k)}{2X_{\text{LT}}(k)}, \quad (5.1)$$

where $Y_{1,2}$ are the two received copies of the long training sequence and X_{LT} its known value. The LS equalizer does not adapt but uses this initial estimate for the whole frame. While computationally very efficient, it is well known that this algorithm suffers as frames get longer or the coherence time of the channel gets shorter [12], [13].

The Least Mean Squares (LMS) algorithm overcomes this limitation. Starting with the same initial estimate as the LS equalizer, it updates the channel after the i -th Orthogonal Frequency-Division Multiplexing (OFDM) symbol using the constellation point $\hat{X}_i$ that the received symbol Y_i was demapped to as

$$\hat{H}_i(k) = (1 - \alpha)\hat{H}_{i-1}(k) + \alpha \frac{Y_i(k)}{\hat{X}_i(k)}. \quad (5.2)$$

With α , we apply a low-pass filter to average the channel coefficients in time domain. Neither the LS nor the LMS algorithm average in frequency domain but consider each subcarrier independently.

The Comb equalizer, in contrast, interpolates linearly in frequency domain using the four comb pilots that are interleaved with the data symbols. Following Fernandez, Stancil, and Bai [65], we use the mean value of the pilots at the border of the spectrum and interpolate with the vector $[m_p, P_1, P_2, P_3, P_4, m_p]$, where $P_{1..4}$ are the four comb pilots and m_p their mean. This interpolation is done for every OFDM symbol. Afterwards, a low-pass filter similar to Equation (5.2) can be applied to also filter in time domain.

The Spectral Temporal Averaging (STA) equalizer is a state-of-the-art algorithm designed to cope with the dynamics of VANETs [65]. The core idea is to filter in both time and frequency domain by updating the channel estimates in two steps. First,

the current symbol is decoded, and the current channel estimate is calculated as

$$\hat{H}_{i, \text{curr}}(k) = \frac{Y_i(k)}{\hat{X}_i(k)} . \quad (5.3)$$

These estimates are used to average in frequency domain by calculating a moving average over β adjacent subcarriers as

$$\hat{H}_{i, \text{update}}(k) = \frac{1}{2\beta + 1} \sum_{n=k-\beta}^{k+\beta} \hat{H}_{i, \text{curr}}(n) . \quad (5.4)$$

In the second step, $\hat{H}_{i, \text{update}}$ is used to average in time domain with a similar low-pass filter as in Equation (5.2).

The LMS and the STA equalizer are decision-directed, using the decoded data symbols to adapt channel estimates. This also implies that wrong decoding decisions will lead to feedback errors that possibly degrade the receive performance. In this thesis, we stick to Fernandez, Stancil, and Bai [65] and select $\alpha = 0.5$ and $\beta = 2$ as parameters for the STA algorithm. For better comparability, we used the same α also with the LMS equalizer. Furthermore, we wanted to isolate the effects of time and frequency selectivity and, therefore, did not apply any averaging in time domain with the Comb algorithm.

5.1.2.2 Signal Trace Recording

To compare the performance of these algorithms in a realistic environment, we conducted a field test near Paderborn, Germany. The hardware and the most important

Table 5.1 – Hardware setup and most relevant parameters used in the field test. (Reproduced from [25] with permission.)

	Parameter	Value
Measurement	Distance	56 km
	Duration	56.5 min
	Frames	> 25 000
	Encoding	BPSK- $1/2$, QPSK- $1/2$
	Frame Size	200 Byte, 500 Byte, 800 Byte
	Channel	178 (5.89 GHz)
Hardware	Sender	based on Atheros AR9280
	TX Power	18 dBm
	SDR	N210 w/ CBX daughterboard
	RX Gain	29 dB
	Antennas	ECOM9-5500 (9 dBi dipole)
	GPS Receiver	u-blox NEO-7N

parameters of the field test are summarized in Table 5.1. As transmitter, we used a WLE200NX mini-PCIe WLAN card, which is based on an Atheros AR9280 chipset that is supported by the Linux *ath9k* driver. Using a recent Linux kernel, this chip supports IEEE 802.11p, i.e., 10 MHz transmissions on the Cooperative Intelligent Transportation System (C-ITS) band at 5.9 GHz. Since the WLE200NX is not sold as an IEEE 802.11p device, we conducted preliminary experiments to assert that the 5.9 GHz band is not attenuated or distorted by hardware filters. In these experiments, we set the bandwidth to 10 MHz and sent frames on both the C-ITS band and on the regular IEEE 802.11a bands at 5.3 GHz and 5.5 GHz. We used an SDR as spectrum analyzer but found no differences, i.e., the output power level and the spectral shape were the same on either band. A limitation of the commercial IEEE 802.11a card is its maximum transmit power of 18 dBm, which is below the maximum power level of VANETs. ETSI ITS-G5, for example, allows up to 23 dBm and 33 dBm on service and control channels, respectively [124]. In our experiments, this did not constitute a problem since the distance between sender and receiver was never larger than 140 m and could easily be covered.

During the experiment, we sent frames with random payloads, cycling through the six combinations of BPSK- $\frac{1}{2}$ and QPSK- $\frac{1}{2}$ frames with sizes of 200 Byte, 500 Byte, and 800 Byte. The average frame rate was about eight frames per second, allowing us to generate a large data set with over 25 000 transmissions during the 56.5 min experiment. The frames were sent on channel 178 at 5.89 GHz.

On receive side, we used an Ettus Research N210 with a CBX daughterboard that covers the frequencies from 1.2 GHz to 6 GHz. The SDR was used to record the raw signal, dumping the samples directly to an SSD drive without any signal processing. Using 4 Byte floats for the real and imaginary parts of the complex baseband samples, the 10 Msps stream from the SDR resulted in 80 MByte/s that had to be stored. During the 56.5 min field test, we captured over 270 GByte sample data. The receive gain of the SDR was set to 29 dB, which corresponds to 92 % of its maximum.

Both cars were equipped with 9 dBi ECOM9-5500 dipole antennas mounted on the roof of the cars as shown in Figure 5.5. We equipped the cars with NEO-7N high precision GPS receivers from u-blox and logged their position and speed every 0.5 s. The values at the time instances of frame transmissions were later interpolated linearly based on the GPS time series. Figure 5.6 plots the GPS trace of our 56 km drive, color-coding the surroundings, which range from freeway, to highway, to rural, to city environments. All data is recorded in one take so that the whole setup, i.e., cars, devices, and positions of the antennas remain constant, allowing for a direct comparison between the environments.

On the freeway, we tried to resemble realistic situations by letting one car fall back to later accelerate and overtake. This way, we do not only capture situations



Figure 5.5 – Picture of the measurement setup.

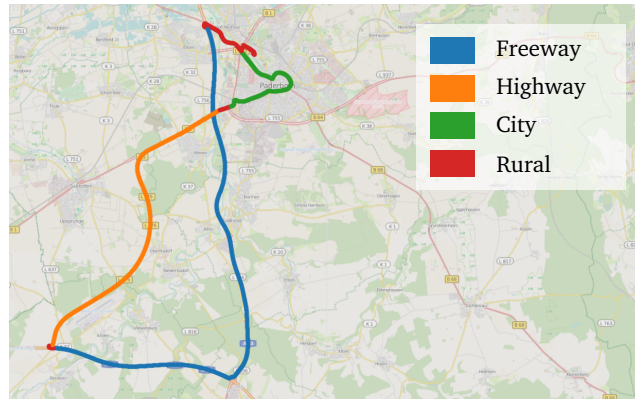


Figure 5.6 – GPS trace of the measurement, showing the various environments. (The map data is © OpenStreetMap contributors, the path is reproduced from [25] with permission.)

with different absolute speeds but also with different relative speeds. In the other environments, it was not easily possible to overtake each other. We, therefore, stuck to varying the distance and letting other cars and trucks get in between sender and receiver. We spent considerable time in each environment, sending 7373 frames in the city, 5624 frames in rural areas, 4842 frames on the highway, and 8019 frames on the freeway.

5.1.2.3 Evaluation of the Signal Trace

After the field test, we post-processed the sample stream with the four implemented channel estimation algorithms. Figure 5.7 shows the number of received frames for each algorithm and environment. On top of the bars, we annotated the percentage of frames that could be decoded by the best algorithm, which was in either case STA. We can see that the difference between the algorithms is rather small, with each algorithm decoding over 94 % of the frames. On the one hand, this shows that our

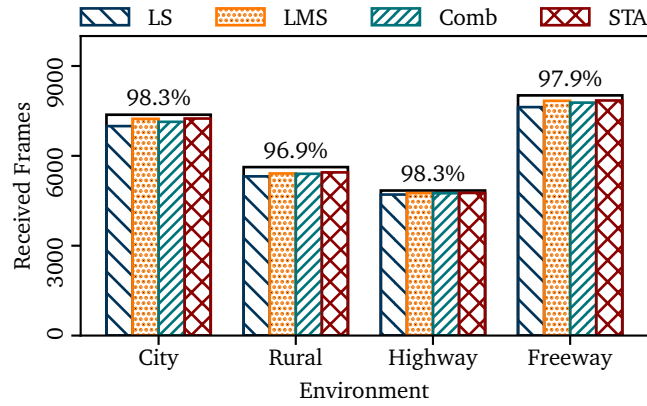


Figure 5.7 – Performance comparison of receive algorithms in different environments. (Reproduced from [25] with permission.)

SDR implementation works well also in realistic environments, on the other hand, it shows that even the simple LS algorithm is able to decode most frames, at least in that particular scenario. This suggests that even WLAN chips that are not specially designed for IEEE 802.11p could provide reasonable performance.

We, however, expect these differences to be more pronounced with higher relative speeds, for example, if the cars approach each other. In that case, the channel changes faster, leading to shorter coherence times. This could degrade the performance of the LS equalizer, as the initial channel estimate becomes outdated during the frame, causing bit errors and, ultimately, dropped frames.

An indicator of the detrimental effect of the frame sizes on the LS algorithm is shown in Figure 5.8 where we plot the percentage of QPSK- $\frac{1}{2}$ frames received at speeds above 80 km/h. While a higher absolute speed does not necessarily imply a more time-variable channel, it is, at least, correlated, since it determines the relative

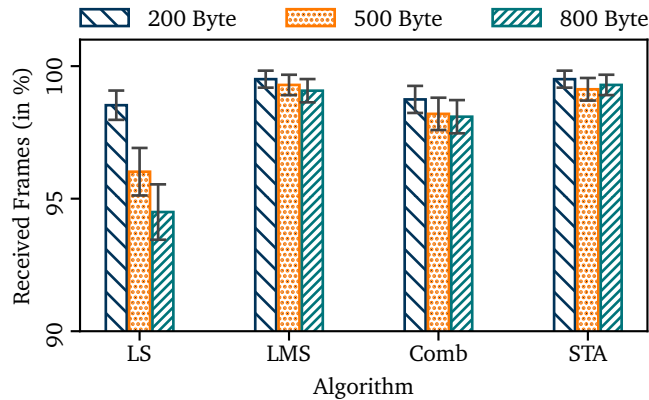


Figure 5.8 – Impact of frame size on the receive performance of the selected algorithms. (Reproduced from [25] with permission.)

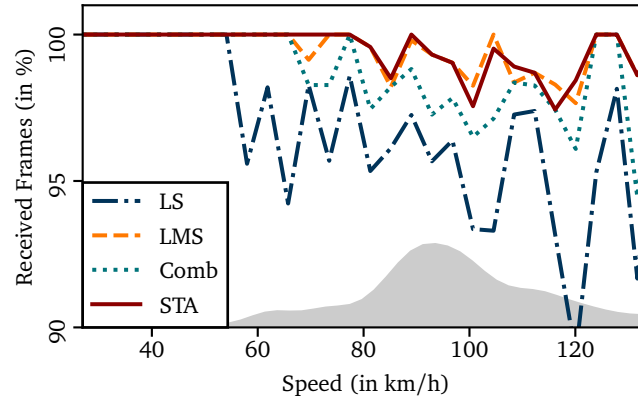


Figure 5.9 – Impact of speed on the receive performance of the selected algorithms. (Reproduced from [25] with permission.)

speed to static reflectors like street signs or the guardrail. We split the data based on the frame sizes and annotate the 95 % confidence intervals. The plot shows that the LMS, Comb, and STA equalizers are able to cope also with larger frame sizes. The LS algorithm, in turn, shows lower performance with larger frames, highlighting the impact of outdated channel estimates.

A different perspective on the data is given in Figure 5.9, where we plot the percentage of received QPSK- $\frac{1}{2}$ frames in the freeway environment against the transmitter’s speed. In the plot, we split the data into 30 bins and calculated the frame error rate per bin. The data basis, i.e., the number of frames per bin varies significantly, as there are, for example, only a few frames at low speeds on the freeway. To indicate the distribution of in total over 4000 frames, we added a density plot to the figure. Shaded in gray, it shows the distribution of transmitter speeds at which frames were sent.

As expected, the trend is that more frames are lost at higher speeds. The plot also highlights the impact of fast fading on the LS algorithm. While the graph does not look like in a textbook, it shows the LS equalizer’s sensitivity to speed. Compared to the other algorithms, the LS algorithm shows a higher variance and lower reception rate. Summarizing the results of our field test, we can conclude that most transmissions did not pose great challenges on the receive algorithms, so that the overall differences between the algorithms were small. If, however, we focus on larger frames or higher speeds, the differences become larger, and the limitations of simple algorithms become more prevalent. To allow fellow researchers to further evaluate the trace with different algorithms, we release our trace data, photos, and a description of the field test on our website (cf. Figure 5.10).⁴

⁴<https://www.wime-project.net/>

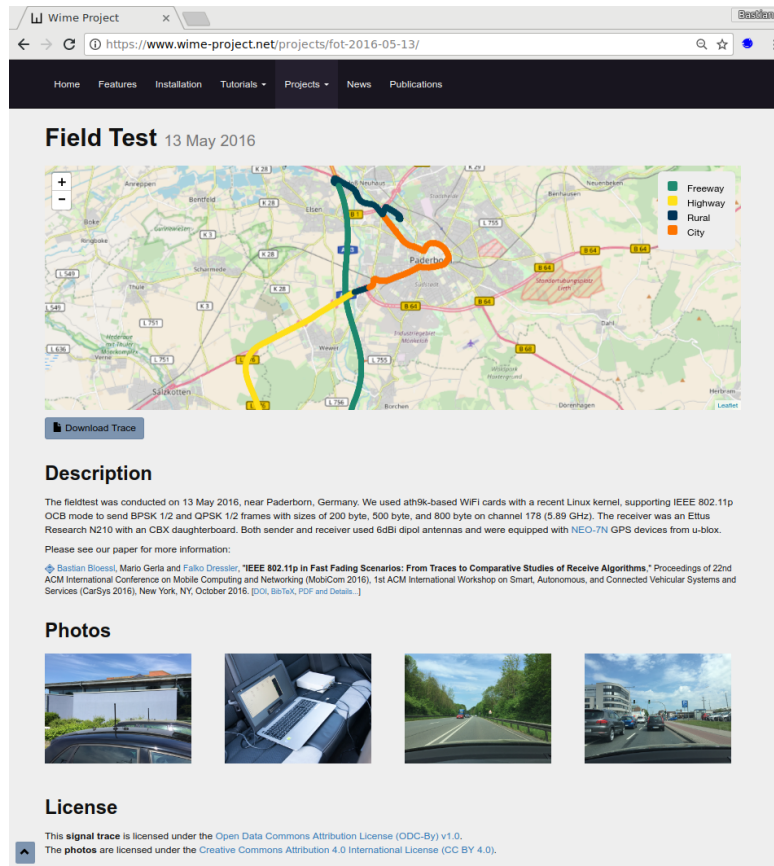


Figure 5.10 – We share the trace data and details of the experiment on our website.

5.2 Impact of Intra-Technology Interference

In our previous experiments, we focused on the performance of a single link. In this study, we go a step further and also consider the impact of interference. More specifically, we use results from our SDR-based testbed to validate a PHY simulation model that is used in popular network simulators. Apart from the relevance of the results themselves, the study also shows how SDR can bridge between Electrical Engineering and Computer Science. We characterize PHY performance in a simple scenario (typically the domain of electrical engineers) and use the results to derive a more abstract model that can be used for macroscopic network simulations (typically the domain of computer scientists). Such network simulators are great tools to design, test, and evaluate VANET applications, as they allow experimenting with protocols in a fast, cheap, and reproducible manner [75]. The realism and accuracy of the simulators depend to a large degree on the quality of their simulation models.

Especially the model of the PHY is crucial for the fidelity of the results, as it decides whether a given combination of signal, interference, and noise allow reception of the frame [129].

Given the importance of the model, it is unfortunate that there is a dispute within the community on how interference should be modeled – or if it can be modeled accurately at all [130], [131]. Popular network simulators, like *ns-3* and *Veins*, assume that interference can be treated similar to noise. The rationale behind that approach might be easy to understand if we keep in mind that the simulation model was adapted from readily available WLAN models operating on the 2.4 GHz Industrial Scientific and Medical (ISM) band. On this band, we can find a large range of interference sources, like microwave ovens, cordless telephones, and Zig-Bee transceivers. It is, therefore, hard to characterize interference in detail [132]. Considering different interference sources, or even the combination of different interference sources, would extremely complicate simulation models. Given this complexity, a pragmatic and practical solution was to treat all kinds of interference like noise. However, in that regard, IEEE 802.11p is special as it uses dedicated spectrum in the 5.9 GHz band. Interference is, therefore, limited to intra-technology interference, i.e., interference from other IEEE 802.11p devices. This might cast doubt on earlier assumptions.

Some researchers argue that there is, in fact, a large difference in the impact of interference and noise. A popular example is the work of Fuxjaeger and Ruehrup [114]. Experimenting with frame capturing, the authors noticed that the Device Under Test (DUT), an off-the-shelf IEEE 802.11p card, could cope better than expected with interference. The authors compared their results to the NIST error model, a state-of-the-art model used by many network simulators. This model is empirically validated with off-the-shelf WLAN cards but only with AWGN [120]. Comparing the measurement results to the error rate calculated by the simulator, the authors concluded that noise must have a more detrimental impact than OFDM interference. Network simulators would, therefore, produce overly pessimistic results in interference scenarios. Contrasting this line of thought, one could even argue in the opposite direction: Compared to noise, an OFDM signal could create more spotted interference exactly at the subcarrier frequencies and might, therefore, be worse than a similar level of noise. These examples show that the relation between noise and interference is, at least, not trivial.

To show the fundamental difference between the two, we created an IEEE 802.11p OFDM signal and added, in one case, white noise and, in another case, a second IEEE 802.11p signal. We calibrated the power levels of the interfering signals to an average power of one. These signals degrade the signal quality of the original frame by shifting the subcarrier constellations away from their ideal constellation points. This deviation is, however, very different for noise and OFDM interference. For the

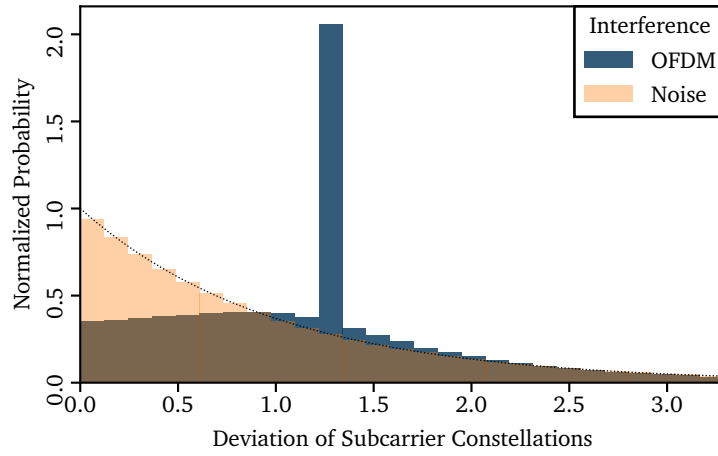


Figure 5.11 – Histograms, showing the impact of OFDM interference and a similar level of noise on the subcarrier constellations of an IEEE 802.11p frame. The histograms are normalized so that integration over the bins yields unity.

noise, it is well known that the deviation follows a complex normal distribution. The power of the deviation is, therefore, negative exponentially distributed as indicated by the light colored histogram in Figure 5.11. The known actual distribution is overlaid by the dashed line.

A similar histogram for OFDM interference is depicted by the darker color. The most important observation is that the distributions of noise and OFDM interference are very different. In particular, OFDM interference leads to a pronounced peak. It turns out that we have to differentiate between two cases: If the OFDM symbols of the two frames are aligned so that the FFT does not cross symbol boundaries, the ideal subcarrier constellation points get shifted according to the constellation of the interfering frame. In our example, we considered an interfering QPSK frame with a constant amplitude, which leads to a constant deviation that is visible by the spike in the distribution. (Note that the peak of the OFDM interference is slightly above 1, since only 52 of the 64 subcarriers are used.) If the FFTs are not aligned, the deviation spreads out over about the same interval as the noise. These simulations clearly show the fundamental differences between noise and OFDM interference. The open question is whether these differences at a very low layer also lead to differences in PHY performance.

When consulting the available literature, we find that there are, on the one hand, papers that deal with PHY performance in different channels but without interference [12], [13]. On the other hand, there are papers that consider interference but do not target PHY performance in general [114], [115]. Instead, they focus on characterizing the capture effect, i.e., the special case where a frame is interfered by

a high power frame. In that case, the receiver might cancel reception of the initial frame and switch over to the high power frame to avoid losing both frames.

Here, we contribute to reconciling these different viewpoints on the impact of noise and OFDM interference on IEEE 802.11a/g/p WLANs in the following aspects:

- we conduct an extensive set of PHY simulations to investigate these questions;
- we systematically cross-validate these simulations with empirical results and over-the-air experiments using both lab equipment and off-the-shelf hardware and find that, on a macroscopic scale, the impact of noise can be modeled by OFDM interference and vice versa;
- based on these findings, we illustrate a way of constructing a testbed for the controlled creation and evaluation of interference scenarios using only off-the-shelf WLAN cards.

5.2.1 Simulative Evaluation

To understand the impact of intra-technology interference and noise on IEEE 802.11p PHY performance, we set up simulations with our GNU Radio-based IEEE 802.11p transceiver implementation. In our simulations, we send 546 Byte QPSK- $\frac{1}{2}$ frames that are interfered either by noise or by another IEEE 802.11p frame. Interference starts with a delay of 122 μ s (corresponding to 31 % of the frame time). The parameters of this and the following experiments are summarized in Table 5.2. The parameters are chosen to allow crosschecking results and are similar to those employed by Fuxjaeger and Ruehrup [114]. To make the simulations as realistic as possible, we slightly resampled one frame to introduce sample, phase, and frequency offsets. Furthermore, we varied the alignment of OFDM symbols between the original and the interfering frame. In every simulation run, we sent 100 frames per run and made 80 runs per configuration.

Since we want to isolate the effect of interference, we set a high Signal to Noise Ratio (SNR) (over 40 dB) at the start of the frame, i.e., during the first 122 μ s. Given the high SNR at frame start, the performance is determined by the power ratio of the

Table 5.2 – Most relevant parameters of our simulations and measurements.

Parameter	Value
PSDU Size	546 Byte
MCS	QPSK- $\frac{1}{2}$
Bandwidth	20 MHz
Channel	178 (5.89 GHz)
Delay	122 μ s

interfered part of the frame. In the following, we always denote this power ratio as the *Signal to Interference Ratio (SIR)* of the frame, even though we alternate between adding noise and OFDM interference.

Figure 5.12 illustrates the frame delivery ratios for various SIRs. The error bars indicate confidence intervals at a confidence level of 95 %. The results show that frame delivery ratios for noise and for OFDM are very similar. The sole difference is that OFDM shows a slightly more stretched curve. Looking at individual runs, we notice that this is because, with OFDM interference, we can clearly differentiate cases where the OFDM symbols were closely aligned (i.e., the FFT windows overlap) to when they were not. Still, with regard to packet level simulations, the results indicate that it is reasonable to treat noise and interference as similar.

To further validate our simulation results, we compared them to the NIST error model [120], which is used in network simulators, like *ns-3* or *Veins*. Compared to our simulations, the NIST model predicts a sharper transition from no reception to reception (data not shown); this is in line with observations when the model was validated against commercial cards [120].

5.2.2 Experimental Evaluation

The simulations suggest that noise and interference may be treated as similar in network simulators. To back up this important result, we set up measurements in our lab. Since we already used an SDR implementation in our simulations, we can run the same software together with a radio front end (we employed an Ettus Research B210 with a 9 dBi ECOM9-5500 dipole antenna) to transmit over the air. Like in the simulations, we generated the signal plus interference and transmitted the combined

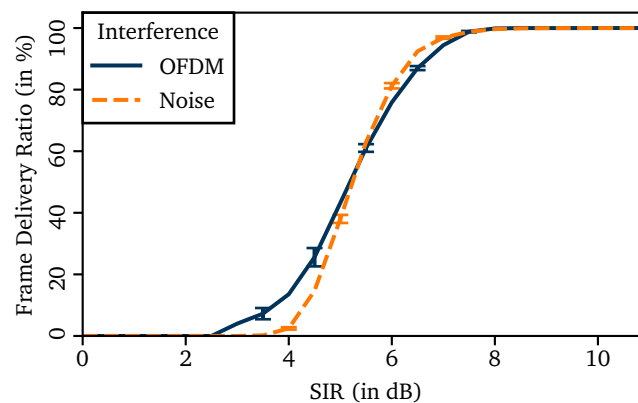


Figure 5.12 – GNU Radio simulations of the frame delivery ratio of an OFDM frame that is interfered by OFDM frames or a similar level of noise. (Reproduced from [24], © 2017 IEEE.)

signal. The mixed signal is sent with high gain to minimize the impact of thermal noise. With this approach, we know the SIR very precisely.

To assert that the effect of noise and interference is not specific to our SDR implementation, we used two off-the-shelf WLAN cards as receivers: a Unex DCMA-86P2, supported by the Linux *ath5k* driver; and a Netgear WNDA3200, supported by the Linux *ath9k_htc* driver. Especially the results from the Unex DCMA-86P2 are interesting, since the card is specifically designed for VANETs and was already used in many field tests [83]–[86]. We modified both drivers to tune to the 5.9 GHz band allocated for C-ITS. Given current limitations of the *ath9k_htc* driver, we performed the measurements at 20 MHz channel bandwidth. Like in the simulations, we sent 546 Byte QPSK- $\frac{1}{2}$ frames and delayed the interfering signal by 122 μ s. The experiments were conducted on channel 178 at 5.89 GHz. Using a spectrum analyzer, we made sure that this channel was vacant in our environment. Again, we repeated the experiment 80 times, sending 100 frames per run.

Figure 5.13 shows the frame delivery ratio for the Unex card. Again, we can see that the performance under OFDM interference and noise is very similar, backing up our simulation results. For the *ath9k_htc* card, the error curves were slightly shifted, but the curves for noise and interference were also identical (data not shown). In sum, different receivers exhibited a different overall performance but behaved similar under noise and OFDM interference. The performance differences between chips might also explain the deviating conclusions of Fuxjaeger and Ruehrup [114]. The authors compared the error model of the network simulator directly with interference measurements. It is, therefore, possible that their particular receiver just did not match the error model; it does not necessarily imply that OFDM interference has a fundamentally different impact on PHY performance.

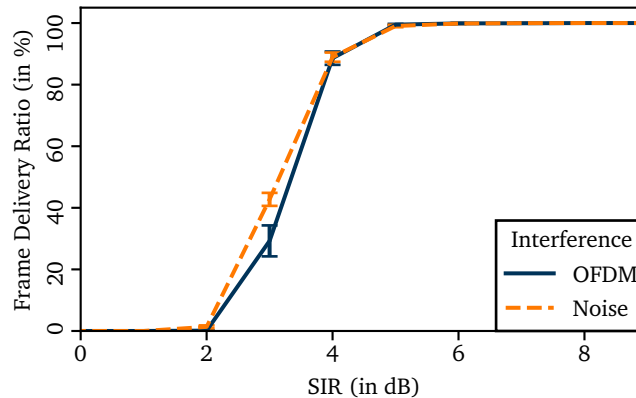


Figure 5.13 – Frame delivery ratio of a Unex DCMA-86P2 card when receiving frames that are interfered by another OFDM frame or a similar level of noise. (Reproduced from [24], © 2017 IEEE.)

5.2.3 Off-the-Shelf Testbed

Finally, we were curious to see whether we could replace the SDR-based testbed with a simpler version that is based on cheap off-the-shelf WLAN cards. A limitation of such a testbed would be that it could only consider OFDM interference, since normal WLAN cards are restricted to sending WLAN frames. Still, such a testbed would be cheaper and easier to set up, yielding a platform that is more accessible for researchers. Furthermore, the results from another testbed provide an additional data point that can be used to cross-check of our previous simulations and measurements.

Figure 5.14 shows an overview of our off-the-shelf testbed. To create interference scenarios in a controlled and reproducible manner, we used a Netgear WNDA3200 USB WLAN dongle and flashed it with a custom firmware that is based on the work of Vanhoef and Piessens [133]. Developed in the security context, the firmware modifies the dongle to emit signals in response to transmissions, i.e., it transforms the WLAN card into a reactive jammer. In a nutshell, the firmware constantly checks whether the transceiver is about to receive a frame and, if this is the case, interrupts reception and sends an interfering frame.

Since the data frame and the interfering frame are sent from different transmitters, we do not immediately know the SIR at the receiver. We, therefore, precede every interference experiment with a measurement run, where we sent frames in a similar configuration but one after the other, i.e., without interference. To ease these measurement runs, we extended the reactive jammer to wait for a configurable delay between sensing the frame transmission and transmission of the jam signal. Adjusting this delay with micro second resolution, we can, on the one hand, create different interference scenarios and, on the other hand, easily conduct the measurement runs. If we delay the jam signal long enough to be sent after the frame, the receiver can log frames from both the regular transmitter and the jammer, allowing us to calculate the SIR based on the received signal strength, which is annotated in the *Radiotap* header of received frames. Like Pei and Henderson [120], we conducted experiments with different transmit gains and found that, at least, the relative power levels reported

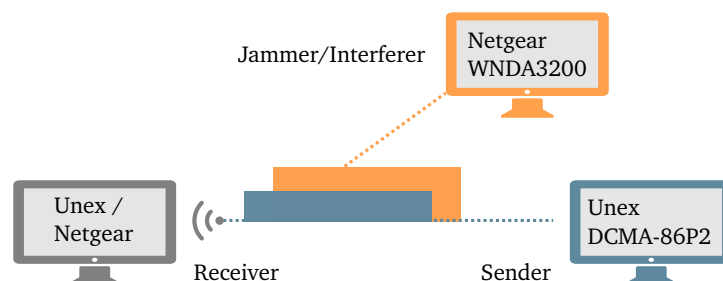


Figure 5.14 – Overview of our off-the-shelf testbed. (Reproduced from [24], © 2017 IEEE.)

by the cards are very accurate. We noticed that the signal strength reported for an interference-free frame and a jammed frame are similar, which implies that the values reported by the card are estimated at the start of the frame, rather than a running average over the whole frame duration. We made sure that the SNR is very high (above 40 dB) at parts that are not interfered. This way, we made sure that the frame delivery ratio is only determined by the SIR of the interfered part of the frame.

To validate our setup, we started by reproducing the experiments of Fuxjaeger and Ruehrup [114] and Lee et al. [115]. Both papers show that a stronger frame is captured starting from a SIR of 8 dB and is captured reliably with a SIR above 12 dB. We configured different SIRs and measured the delivery ratio of the interfering frame. The corresponding graph is shown in Figure 5.15; it matches amazingly well. Again, the error bars correspond to the confidence intervals with a confidence level of 95 %. While these results were already well established, the experiment serves, in this context, as a basic validation of our setup.

After this preliminary step, we configured the testbed to recreate the setup from previous measurements, i.e., we sent QPSK- $\frac{1}{2}$ frames on channel 178 and delayed the jammed frame by 122 μ s. By adjusting the transmit power of the sender, we configured different SIRs and measured frame delivery ratios. Figure 5.16 compares the results of these measurements against the SDR measurements reported in the previous section. The error bars, again, indicate the confidence intervals at a confidence level of 95 %. The most important observation is that the curves are very similar, which backs up our previous results and validates the operation of the testbed. To make sure that we do not merely observe the specific characteristics of one particular receiver (or receiver model), we conducted similar measurements with a Netgear WNDA3200. We confirmed that, also with this setup, the frame

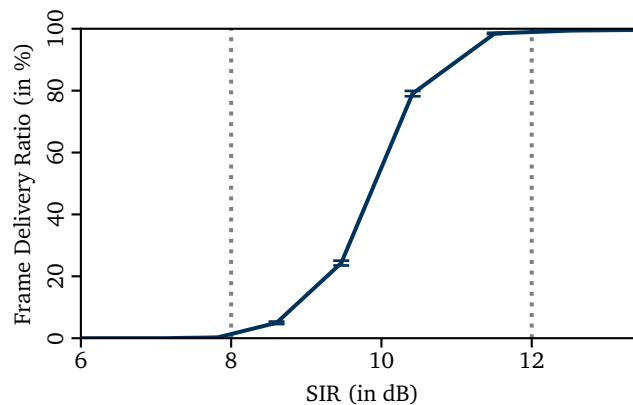


Figure 5.15 – To validate our setup, we reproduce results from the literature and measure the success rate of frames captures at different SIRs.

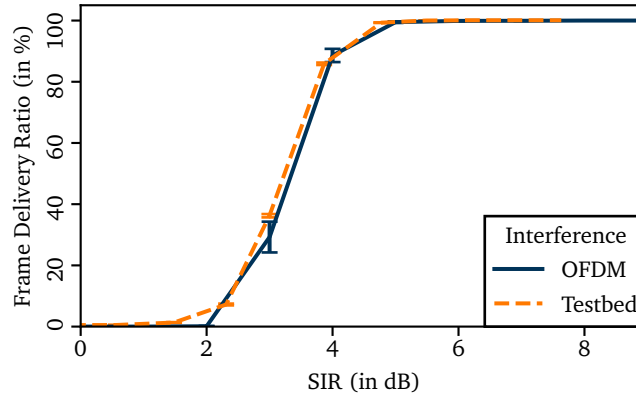


Figure 5.16 – Frame delivery ratio of a Unex DCMA-86P2 under OFDM interference created with an SDR and our testbed. (Reproduced from [24], © 2017 IEEE.)

delivery ratio of the SDR experiments and our testbed matches very well (data not shown).

While this is very motivating as far as developing a cheap and effective testbed goes, the testbed setup is not without drawbacks. Since the jammer only *reacts* to transmissions, we cannot create arbitrary interference situations, as there is always a delay between the original frame and the interfering frame. Using a spectrum analyzer, we measured the delay at the receiver to be $122\ \mu\text{s}$, which corresponds to about 15 OFDM symbols at 10 MHz bandwidth. Yet, considering that the testbed does not rely on SDRs (like [114]) and does not require complex synchronization and split interference domains (like [115]) – and considering that it uses real off-the-shelf WLAN hardware – it might provide an interesting avenue for future research. To allow experimenting with our testbed, we make the firmware with our modifications available on our website.⁵

5.2.4 Conclusion

To understand the impact of noise and interference on IEEE 802.11p, we conducted detailed PHY simulations with our SDR implementation and compared them to over-the-air measurements in two different testbeds. While we did not explore the full parameter space with all MCSs, frame sizes, and interference situations, our experiments produced coherent results and a subset could even be cross-checked with the literature. Overall, these results strongly suggest that intra-technology interference and noise have a similar impact on the frame delivery rate of IEEE 802.11p networks. The consequences for the network community are positive in the sense that the commonly adopted simplification of network simulators (i.e., to treat

⁵<https://www.wime-project.net/>

noise like interference) seems reasonable. Apart from that, the result has another interesting implication for constructing testbeds for IEEE 802.11 devices. If noise and interference have a similar impact on the performance, we might be able to build more accessible physical layer testbeds that are based on normal WLAN cards.

5.3 The Scrambler Attack

Previous experiments already showed the applicability of our SDR-based IEEE 802.11 a/g/p transceiver for physical layer studies. Here, we go one step further and show how the information that is only accessible with an SDR can be used to develop an attack on the location privacy of vehicles. Apart from the novel attack vector, the work also highlights the flexibility of our tool by showing an exemplary use-case in a different context.

It is beyond dispute that C-ITS have many advantages. Yet, serious privacy concerns still remain [134]. By overhearing unencrypted periodic beacon messages of vehicles, it is possible for operators of networks of Roadside Units (RSUs) to track drivers through the network and reveal their locations [135]. To counter this, both the European and the US system dictate the use of a Public Key Infrastructure (PKI) employing pseudonymous identifiers, which have to be signed by a certificate authority and cannot be linked to the real identity of a driver by anyone else [136], [137]. Location privacy is then achieved by frequently changing these pseudonyms (along with all other identifying information such as the Medium Access Control (MAC) address). The goal is to make it impossible to track vehicles by linking messages with different pseudonyms to each other.

Privacy is becoming a critical concern also in the normal WLAN domain. The use of privacy preservation techniques has, for example, been integrated into new Apple products where MAC addresses are randomized during the active probing for new base stations. This trend will continue with new wearable devices. In the scope of this thesis, however, we primarily focus on vehicular networks as a base technology. Attack vectors to still link messages despite changing identifiers usually include the exploitation of PHY characteristics, e.g., unique features of the electromagnetic waveform emitted by a particular transceiver. However, such approaches need to rely on potentially fragile features of the channel or the hardware. Thus, they are unlikely to work well in highly dynamic vehicular networks.

In this thesis, we reveal and discuss a novel attack vector based on data contained directly in the PHY: the scrambler state. Scrambling, despite its name, is not related to network security but is an important process to improve wireless communication performance. An attack exploiting this mechanism becomes possible by employing an SDR rather than off-the-shelf hardware, as these transceiver chips do not disclose

the necessary information. Conversely, an SDR allows an attacker free access to the complete physical layer frame. We identified a robust passive fingerprinting technique based on non-random initial scrambler states that can be exploited to considerably degrade the location privacy of vehicles and, thus, their drivers.

Using our SDR-based IEEE 802.11a/g/p transceiver, we were able to gain access to this information and reverse engineer the scrambler algorithms of current IEEE 802.11p prototypes and off-the-shelf hardware. We present our investigation of the weakness of these algorithms as well as the results of an extensive simulation study of best/worst case scenarios of an attacker attempting to track vehicles across an intersection and through blind spots in radio coverage. This allows us to give a quantitative indication of the impact of the presented attack vector.

5.3.1 Related Work

Due to their large success, IEEE 802.11 networks gained much interest from the research community. When these devices become mobile, as it is the case with WLAN-enabled mobile phones and vehicular networks, the preservation of location privacy is a non-trivial challenge. On the PHY, several attack vectors to track users' mobility have been identified, and countermeasures have already been discussed [138]. Most of these attacks on IEEE 802.11 networks can be directly applied to IEEE 802.11p networks, however, their feasibility in highly dynamic environments such as vehicular networks has to be reconsidered.

Furthermore, characteristic distortions of the physical waveforms of the signal can be exploited to re-identify a user. These distortions can be introduced by the wireless channel [139] or by imperfections and variations of the analog part of the hardware [140], [141]. Klein, Temple, and Mendenhall [142] present a method to identify WLAN devices with the help of characteristic features of the preamble by using a sophisticated signal analyzer using a static setup in a shielded chamber. However, in highly mobile networks, like VANETs, where the signal is greatly influenced by effects of the wireless channel, these specific characteristics might be difficult to detect. Ureten and Serinken [143] show how the transient phase of IEEE 802.11b network cards, another feature of the physical waveform, can be exploited to identify a device with an accuracy of up to 98 %. During the transient phase, i.e., immediately after the network card switched to transmit mode, the signal has a characteristic shape, which is induced by powering up transmit components like amplifiers. Even though this approach is very reliable in static scenarios, its practical exploitation in vehicular environments has yet to be shown.

Kohno, Broido, and Claffy [144] show how unique clock drift characteristics of a device can be exploited using timestamps of TCP packets. This attack might also be applicable in VANETs as periodic beacon messages include a millisecond

timestamp. However, vehicles in VANETs are equipped with GPS receivers that allow us to derive the time with very high precision, possibly limiting the applicability of this method. Another fingerprinting technique is the utilization of features of higher layers like protocol and traffic characteristics. Franklin et al. [145] show how small, vendor-specific implementation details can be used to identify the used hardware. A limitation of these methods is that they do not identify a specific user but disclose the model or vendor of the hardware. The scrambler attack presented in this thesis is able to specifically identify a unique user with high probability, because even though the same hardware might be used by different users, their state differs.

5.3.2 Description of the Attack Vector

Frame encoding in IEEE 802.11a/g/p OFDM PHY is a rather involved process. However, for our attack, only the scrambler is relevant. With IEEE 802.11, the binary payload of a frame is scrambled just before forward error correction is applied. The scrambler *xors* the input data with a pseudo-random sequence, generated by a linear feedback shift register as depicted in Figure 5.17. This produces an uncorrelated binary sequence with uniformly distributed bits, maximizing the entropy and, thus, information content.

In OFDM systems, the scrambler has another advantage besides maximizing the information content of the input data: Since the scrambler generates a different pseudo-random sequence per frame, the same data payload is mapped to different binary sequences and thus, physical signals. This is desirable, especially with OFDM, since certain bit patterns are mapped to disadvantageous waveforms with very high Peak-to-Average Power Ratio (PAPR) [146]. Without a scrambler, the same payload would always generate the same physical waveform. Therefore, certain payloads could experience systematically higher packet error rates.

The output of the scrambler depends only on its initial seed. According to the standard, the scrambler should be seeded randomly with a nonzero value for every frame. More precisely, it states: “When transmitting, the initial state of the scrambler shall be set to a pseudo-random nonzero state” [59, Section 17.3.5.5].

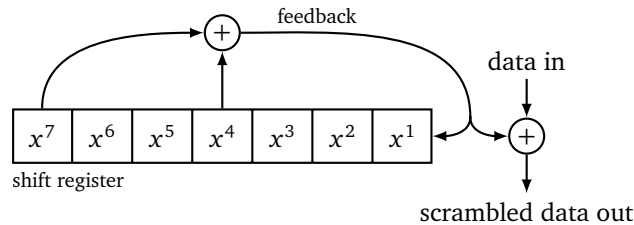


Figure 5.17 – Schematic overview of the IEEE 802.11 scrambling algorithm. (Reproduced from [27], © 2015 IEEE.)

The receiver can reverse this process to decode the packet since the transmitter prefixes the payload with seven zeros. With these seven scrambled zeros, the receiver can reconstruct the initial state of the linear feedback register.

The seeding of the scrambler piqued our interest, as it might allow for an attack vector: If it was possible to correlate scrambler state seeds across multiple messages, this would allow for the re-identification of devices, and thereby drivers. Furthermore, such an attack would be highly robust against channel variations and completely bypass all currently considered privacy preserving mechanisms.

5.3.3 Applicability to Current Hardware

With our SDR-based testbed, we are able to investigate how different vendors implemented the pseudo-random seeding of the scrambler. Since IEEE 802.11p networks are not deployed, there are no commercial consumer systems available yet. Instead, experimentation is done with either very expensive prototype systems or adapted WLAN cards. We use our receiver to investigate the scrambler of IEEE 802.11p devices of either category.

First, we examine the Cohda Wireless MK2, a well-known prototype system, which has been used for major field trials in the US and Europe. The MK2 is an ARM-based PC with an IEEE 802.11p radio implemented on a Field-Programmable Gate Array (FPGA) that ships with all the firmware and software of a complete IEEE 1609 Wireless Access in Vehicular Environment (WAVE) enabled On-Board Unit (OBU) or RSU; we used firmware revision 4.0.14615. As second device, we used the Unex DCMA-86P2 miniPCI card, which is based on the Atheros AR5413 WLAN chip. With Atheros being one of the market leaders for WLAN cards, this can be regarded as very representative off-the-shelf hardware. The card is supported by the standard Linux kernel; we used the Linux 3.9.0 kernel with a modified *ath5k* driver that allowed us to set the bandwidth to 10 MHz and to tune to the C-ITS frequencies in the 5.9 GHz band.

We found that both devices implement a very simple – and most notably, a fully deterministic – algorithm to seed the scrambler. Our experiments revealed that the MK2 has a freewheeling scrambler in the sense that the state is not reset at all but is running from frame to frame without reinitializing its state (as a side effect, this also means that if the packet size is a multiple of the cycle length of the scrambler, the seed does not change at all). The Unex card uses a simple counter, i.e., the initial scrambler seed is incremented by one for each frame that is sent. We tested different scenarios (e.g., we set the card to monitor and ad hoc mode, we generated cross-traffic that the card overheard) to make sure that no external parameter has an impact on either of the scramblers investigated.

Obviously, both algorithms allow for a trivial re-identification of consecutive frames from one card and, thus, to re-identify vehicles, even if MAC addresses (or pseudonyms) are changed between two frames. Since we assumed that also WLAN cards use over simplistic algorithms, we conducted initial experiments with commercial devices like a MacBook Air and indeed found suspicious behavior like network beacons with constant initial scrambler seed. An in-depth investigation of WLAN devices is, however, out of the scope of this thesis. In fact, since the publication of this work, another group extended our work and, indeed, found deterministic seeding also in other consumer products [147].

5.3.4 Implications for Privacy

The standards' definition of the scrambler is problematic in terms of privacy as it does not clearly state how the pseudo-random sequence should be derived. From a communications perspective, it is sufficient to change the scrambler values per frame. This seems to have led to the situation that most vendors employ very simple algorithms, not considering possible implications for location privacy.

The most important privacy protection in vehicular networks is the use of pseudonyms that are changed according to a pseudonym changing strategy. To ensure location privacy and untraceability, messages sent by the same vehicle but with different pseudonyms must not be linkable to each other. If an eavesdropping attacker is able to use transmitted scrambler values to link messages regardless of their pseudonymous identifier, this privacy measure is circumvented and rendered useless. For example, in the case of a Unex card, which increments the scrambler value by one per frame, an attacker overhearing frames $\binom{A}{10}, \binom{A}{11}, \binom{B}{12}, \binom{B}{13}$, with $\binom{P}{n}$ being a frame with pseudonym P and scrambler state n , is (with all but certainty) able to identify A and B as being the same entity.

Also in more complex scenarios where an attacker put up several receivers but is not able to fully overhear all network traffic, non-random scrambler values can be used to still link messages with different pseudonyms. If the attacker is able to guess the amount of messages sent by a vehicle when it was not within the transmission range it can predict the scrambler values and then re-identify the vehicle. This attack becomes especially feasible when vehicles use static (or a discrete set of) beaconing frequencies, and in the case of Cohda devices, use messages of the same length.

5.3.5 Evaluation of Impact

To obtain a quantitative indication of the impact of our attack on the location privacy of drivers in vehicular networks, we conducted an extensive set of simulations using the Veins framework [148]. We extended the framework so that vehicles either

used a simulated IEEE 802.11p radio from Cohda, Unex, or one that uses correctly implemented pseudo-random scrambler values.

5.3.5.1 Simulation Setup

To be able to accurately gauge the impact of the scrambler attack, we investigate two challenging scenarios. Instead of the usual straight, fully covered stretch of freeway, we investigate a large urban intersection where vehicles can turn (Figure 5.18) as well as a 3 km stretch of 3-lane freeway with a large blind spot (Figure 5.19). We generated vehicular mobility in both scenarios using the microscopic traffic simulator SUMO and kept the number of vehicles constant throughout the simulation: for every vehicle that left the scenario a new one of a random preset type with a new, random route was inserted.

As we believe that the scrambler attack is able to circumvent privacy protection on the MAC layer and higher layers, such as pseudonym changes, we investigated a best-case scenario for privacy: vehicles used a new pseudonym for each message, making it impossible to map messages based on any upper layer identifier. Also, vehicles emitted beacons with a frequency of only 1 Hz – the lowest possible beacon frequency according to the ETSI family of standards [149] – which represents the best

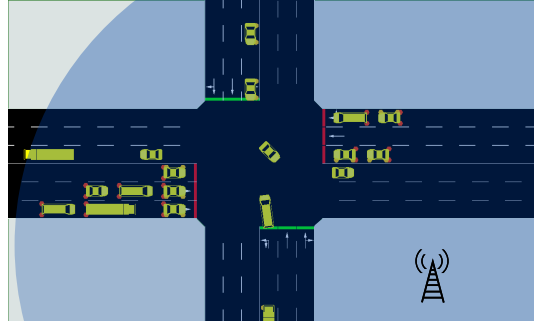


Figure 5.18 – Fully covered crossing scenario. (Reproduced from [27], © 2015 IEEE.)

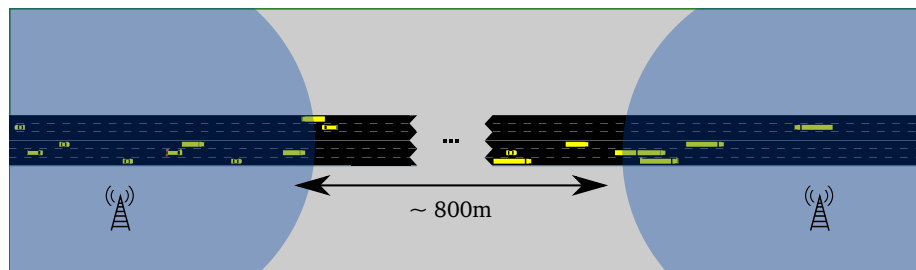


Figure 5.19 – Freeway scenario with blind spot in between attacker receivers. (Reproduced from [27], © 2015 IEEE.)

case in terms of location privacy. The PHY was simulated using two-ray-interference path-loss with a transmission power of 20 mW, leading to a theoretical transmission range of about 600 m. All parameters are summarized in Table 5.3.

5.3.5.2 Attacker Model

The attacker in our simulation deployed (connected) receivers along the road and overhears periodic broadcasts of vehicles to extract information like heading, position, and speed. We, furthermore, assume that the attacker knows that different vendors use characteristic scrambler algorithms.

In the intersection scenario (cf. Figure 5.18), the theoretical transmission range allowed the attacker to receive packets from vehicles approaching and leaving the intersection and on the intersection itself. A vehicle is considered tracked when it was possible to fully recreate the distinct path of a vehicle over the intersection from receiving the first packet until receiving the last packet. Note, as in our simulation the attacker is not omniscient but uses a radio receiver, she can experience packet loss and therefore lose track of a vehicle or associate an overheard beacon with the wrong vehicle.

In the freeway scenario (cf. Figure 5.19), the attacker was not able to fully cover the whole scenario but placed two receivers along the freeway with a blind spot of 800 m between them. Here, a vehicle is considered tracked if it was possible to track its path from entering the transmission range of the first receiver and leaving the transmission range of the second one.

To perform the actual tracking, we deployed an enhanced correlation tracking algorithm: When trying to associate received beacons with existing vehicle tracks, it accounted for physical limits of vehicular movement in terms of acceleration, speed, and heading and consecutively used Edmond's maximum weighted matching algorithm [150] to find the best association hypothesis. This tracking method is computationally inexpensive and has been shown to be very effective[151].

Table 5.3 – Simulation Parameters. (Reproduced from [27], © 2015 IEEE.)

Parameter	Setting
Framework	extended Veins
Scenarios	Intersection, Freeway
PHY/MAC	IEEE 802.11p/IEEE 1609.4
Transmission Power	20 mW
Radio Sensitivity	−89 dBm
Beacon Frequency	1 Hz
Simulation Time	300 s
Repetitions	50

To understand the impact of the scrambler attack, we compared this already advanced tracking algorithm with a variation that also exploited information about scrambler states: For each sequence of consecutively received beacons that the tracking algorithm deemed likely to be from the same vehicle (e.g., due to correlation of position, speed, etc.), it tried to infer which IEEE 802.11p device the vehicle might be using (by correlating the beacons' scrambler states). For vehicles where this succeeded, the tracking algorithm was then able to extrapolate future scrambler states and use this information to rule out potential associations of beacons and vehicles, limiting the number of candidate tuples and thus easing tracking.

No other information was used for both tracking mechanisms. For example, the attacker did not exploit the beacon delay to determine which vehicle sent which beacon, as this could be easily prevented by distributing beacon events uniformly over the beacon period.

5.3.5.3 Results

Figure 5.20a shows our results for the intersection scenario. In this and the following plots, the error bars correspond to the confidence intervals for a confidence level of 95 %. The figure shows the somewhat worrisome picture that it is almost impossible to confuse an attacker with pseudonym changes when (almost) all messages can be overheard. This confirms earlier findings [152], suggesting that in these cases privacy can only be achieved by not sending any packets. Although the tracking probability was already above 98 %, the usage of additional scrambler information could increase these values even more.

The results for the freeway scenario are shown in Figure 5.20b. The blind spot between the two receivers made it considerably harder (dotted, orange line) for the attacker to track vehicles. Dynamics in the mobility of vehicles such as lane changing, overtaking, or varying velocities lead to wrong associations of beacons to vehicles on the attacker side. We observed that the mobility generated by SUMO seemed to be more dynamic as one would expect; to confuse a 'normal tracking' attacker in real life, the gap between the receivers would likely have to be wider. Congestion setting in at the highest vehicle density caused a slight increase in tracking probability, due to fewer lane changes and passing maneuvers.

When the attacker used additional scrambler information to track vehicles the situation completely changed (solid, blue line): We observe that the gap between the two receivers only marginally influenced the capability to track vehicles. Approximating the number of beacons presumably sent by a vehicle while driving in the uncovered section of the freeway, the attacker is able to guess a number of possible scrambler values. Using this technique, we obtained tracking probabilities of over 95 %, almost reaching the level of the fully covered intersection scenario.

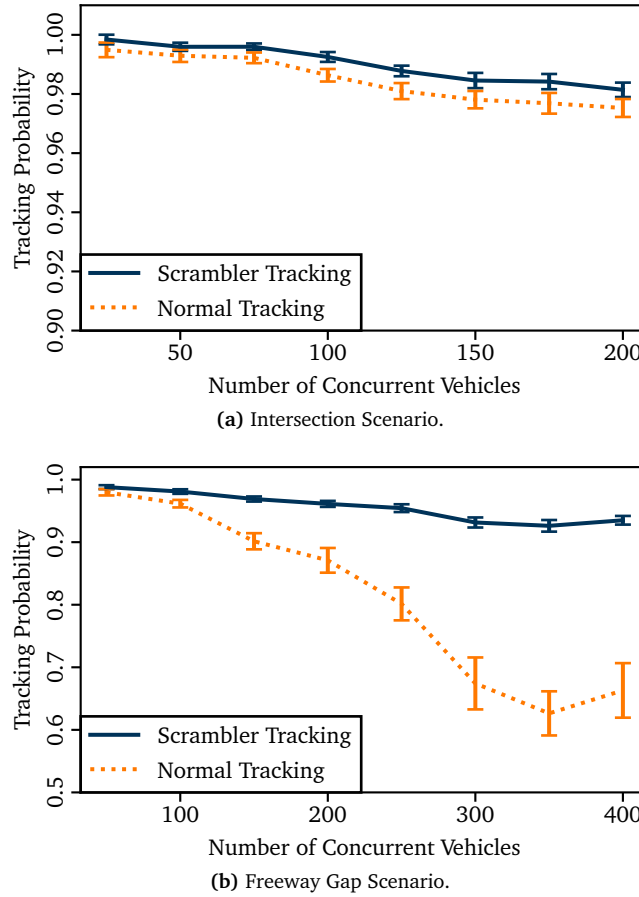


Figure 5.20 – Impact of the Scrambler Attack in different scenarios. (Reproduced from [27], © 2015 IEEE.)

This shows that even on a very busy freeway with interrupted radio coverage the scrambler attack allowed the attacker to effectively circumvent any higher layer privacy protection and track a large portion of vehicles. From this, we conclude that also random silent times [153] (a privacy measure that is likely to be used in the final system [154]) can be rendered ineffective by non-random scramblers.

To fully illustrate the crucial requirement of unpredictable scrambler values, we analyzed the results for the freeway scenario deeper, showing the tracking probability differentiated by the type of IEEE 802.11p radio (Figure 5.21). As can be seen, location privacy cannot be achieved using a predictable scrambler – the attacker was able to track almost every vehicle using the Cohda or the Unex radio. Even the vehicles using a random scrambler (dashed, teal line) suffer from the now smaller number of vehicles possibly confusing an attacker. Their probability of being tracked is considerably higher than it was when scrambler values were not exploited to obtain information (dotted, orange line in Figure 5.20b). This again underlines

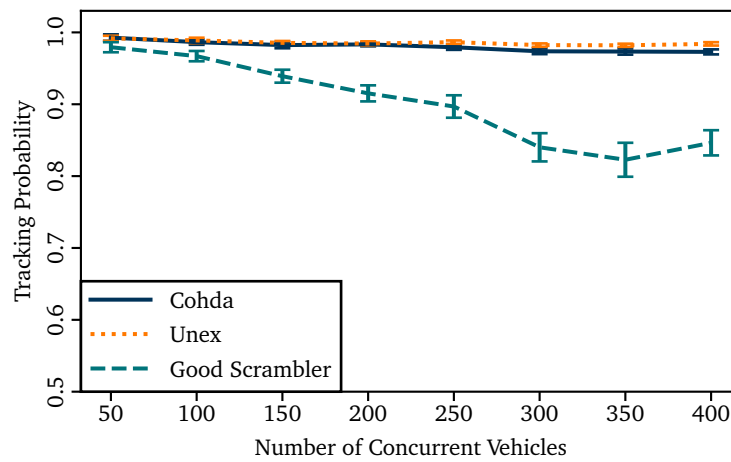


Figure 5.21 – Tracking probability depending on the type of IEEE 802.11p radio using the scrambler tracking. (Reproduced from [27], © 2015 IEEE.)

the necessity to address this problem and not allow for a circumvention of higher layer privacy measures.

5.3.6 Countermeasures

The most obvious solution is to employ a cryptographic pseudo-random number generator, possibly seeded by the large number of entropy sources in a vehicle (e.g., time when the vehicle was started, sensor values like fuel level and tire pressure, or meta data of communication like noise level, number of neighbors, and received data). Another solution is the deployment of constant network-wide scrambler values, however, this could possibly degrade network communication performance [146]. Then, an eavesdropper cannot derive the state of the internal pseudo-random number generator based on a single packet and very hard from a sequence of packets.

For the Cohda MK2, this change would be straightforward to implement since it does not rely on a transceiver chip but implements all logic on reconfigurable FPGAs. Therefore, it should be possible to fix the scrambling algorithm with a firmware update of the prototype. For off-the-shelf hardware, the picture is different. Because vendors do not provide detailed information about their hardware design, it is hard to tell where certain functionalities are implemented and if this solution can be achieved with a driver or firmware update. In the worst case, the scrambling algorithm is implemented in hardware and cannot be fixed, which means that the chip would have to be replaced.

5.3.7 Conclusion

We identified a novel attack vector on the location privacy of vehicles that leverages and exploits over-simplistic implementations of the pseudo-random number generators used by an integral component of all normal WLAN and IEEE 802.11p radio transceivers. This component is the scrambler, which is crucial to ensure good performance at the PHY. Sequences of the scrambler state can be predicted by overhearing a single packet, making it possible for an eavesdropper to associate different pseudonymous messages with the same sender. This passive, undetectable attack can be considered a physical layer attack and, therefore, no higher-layer privacy protection mechanism such as the use of pseudonyms can compensate for it. In contrast to existing lower-layer attacks, however, it is extremely robust, as it makes use of data rather than signal characteristics.

To show how severely location privacy can be degraded by our attack, we conducted an extensive set of simulations. Even in scenarios where vehicles traveled through sections where an adversary was not able to overhear messages, it was possible to reliably track vehicles. The results highlight the importance to use cryptographic pseudo-random number generators – not to increase the performance of the system but to preserve the location privacy of drivers. We see our results as a first step towards enabling privacy protection mechanisms on a large scale.

Chapter 6

Conclusion

Today, as autonomous cars are about to become a reality, we have to set the tracks for the next generation of transportation. It is without doubt that future vehicles will use wireless communication to exchange information directly with each other and maybe infrastructure nodes to form a vehicular network. The ability to communicate will not be a mere add-on but a true game changer, allowing us to evolve from *autonomous* to *cooperative* driving. The implications of this step will be enormous and open up a wide spectrum of novel applications that will make transportation safer, more efficient, and more comfortable than ever before. While these Cooperative Intelligent Transportation Systems (C-ITS) will rely on multiple communication technologies, it is likely that IEEE 802.11p, i.e., automotive Wireless LAN (WLAN), will play a central role. Apart from the fact that the technology is readily available, IEEE 802.11p bears additional advantages as it does not rely on infrastructure nodes, allows direct communication, and has free-to-use dedicated spectrum allocated. Together, these advantages make the technology well-suited for safety-related applications, which are regarded as one of the main drivers for C-ITS.

Given their huge potential, vehicular networks attracted attention from both industry and academia. In the focus of many studies, they developed into a major technology. Today, their characteristic challenges are well understood and regulatory bodies in Europe, the US, and Japan are finalizing the corresponding standards. At this stage of the development process, reproducible measurements and field tests were identified as important next steps to bring the technology forward. By building prototypes, we can identify weaknesses in system design or provide the ultimate proof-of-concept. Also when reaching out to the general public, a working field test makes a strong argument for readiness and feasibility of the technology. This thesis can be seen as a contribution to this line of research. It detailed the design, implementation, and evaluation of the first Open Source IEEE 802.11a/g/p transceiver for a general purpose Software Defined Radio (SDR) framework. Using

this architecture, the physical layer (PHY) is implemented completely on the PC and does not rely on any hardware- or software-specific features. This unique property allows our transceiver to be used in each step of the research process, from simulations to measurements in the lab or in the field. It provides a holistic research and development framework that bridges the gap between theory and practice.

To establish our transceiver as a credible tool for research, we evaluated it extensively through simulations and real-world experiments. Designed specifically for Vehicular Ad Hoc Networks (VANETs), we tested it with both normal WLAN cards and IEEE 802.11p prototypes. Apart from PHY performance, we also studied the computational demands of individual components to make sure that it can operate in real-time on a normal PC. We have seen that this is, indeed, the case, even for the most challenging scenarios. While the PHY was the main focus of our work, we further explored the possibilities and limitations of our SDR platform. By implementing selected time-critical functionality on the Field-Programmable Gate Array (FPGA) of the radio, we realized Automatic Gain Control (AGC) and standard compliant channel access for broadcast transmissions without giving up the advantages of a General Purpose Processor (GPP)-based SDR. To demonstrate our transceiver's applicability for experiments that go beyond simulations and lab measurements, we conducted two field tests. The first compared the performance of different IEEE 802.11p prototypes; the second compared the performance of different receive algorithms. While the first showed that our transceiver provides reasonable performance in a realistic environment, the latter backed up results from the literature, highlighting the shortcomings of simple algorithms that are typically used in WLAN cards. The high mobility and the resulting time variability of the channel ask for advanced receivers that can cope with dynamics of VANETs.

To expand on the performance characterization of the PHY, we used our SDR transceiver to compare the impact of noise and intra-technology interference on IEEE 802.11p. With this study, we addressed a dispute in the community as to whether noise or interference has a more detrimental impact. This disagreement is unfortunate, considering that a detailed understanding of these effects is needed to derive realistic and accurate PHY models for network simulations. Given the popularity of network simulations for research on vehicular networks, it is important that their results are based on solid ground. In the study, we used our IEEE 802.11p implementation for both simulations and over-the-air measurements. Here, we showed how to complement simulation results with real measurements, following the research cycle described in the introduction. Together, the results provide a consistent picture, increasing the confidence in our findings. Even though we did not explore the full parameter space, our results are positive in the sense that they support the assumptions that are adopted by most network simulators.

Finally, we highlighted the flexibility of our prototype by studying a totally different aspect of VANETs: With the ability to look into all steps of the decoding process, our SDR-based transceiver allowed us to identify a novel attack on the location privacy of vehicles. The attack exploits the scrambler seed, i.e., information from the PHY that is not available from normal devices. When inspecting the seeds, we noticed, that, at least, the devices in our study did not seed the scrambler pseudo-randomly (as mandated by the standard) but used a deterministic pattern. Through network simulations, we showed that these predictable patterns can be exploited by a passive eavesdropper to track vehicles even in challenging scenarios with partial network coverage. We believe that the scrambler seed is a particularly interesting attack vector. On the one hand, it is robust, since it does not use potentially fragile features of the channel or the analog RF circuitry but uses binary data from the PHY that every receiver derives during the decoding process. On the other hand, it is a physical layer attack that cannot be mitigated by typical privacy-preserving mechanism, like Medium Access Control (MAC) address randomization or pseudonyms. In fact, the scrambler is such an integral part of the PHY that most devices will probably implement it in hardware, suggesting that the attack cannot easily be fixed through firmware updates.

We think that the applications of our transceiver do not end here and hope that it will continue to serve as a tool that helps to better understand the performance of WLAN for automotive applications. We believe that the insights and lessons learned from experiments are invaluable and allow us to design more reliable vehicular networks. And, in the end, this might be the deciding factor for the general acceptance of the technology, especially with regard to safety-critical applications.

List of Abbreviations

AC	Access Category
ACK	Acknowledgement Frame
ADC	Analog-to-Digital Converter
AGC	Automatic Gain Control
AIFS	Arbitration Inter-Frame Space
ASIC	Application-Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BPSK	Binary Phase-Shift Keying
BSS	Basic Service Set
BSSID	Basic Service Set Identifier
C-ITS	Cooperative Intelligent Transportation System
C-V2X	Cellular Vehicle-to-Everything
CA	Certificate Authority
CAM	Cooperative Awareness Message
CCA	Clear Channel Assessment
CCH	Control Channel
CCK	Complementary Code Keying
CRC	Cyclic Redundancy Check
CSMA	Carrier Sense Multiple Access
CTS	Clear To Send
DAC	Digital-to-Analog Converter
DCC	Decentralized Congestion Control
DCF	Distributed Coordination Function
DENM	Decentralized Environmental Notification Message
DIFS	DCF Interframe Space
DSP	Digital Signal Processor
DSRC	Dedicated Short Range Communications
DSSS	Direct-Sequence Spread Spectrum
DUT	Device Under Test
ECU	Engine Control Unit

EDCA	Enhanced Distributed Channel Access
ERP	Extended Rate PHY
ETSI	European Telecommunications Standards Institute
FCS	Frame Check Sequence
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FPGA	Field-Programmable Gate Array
GCDC	Grand Cooperative Driving Challenge
GPL	General Public License
GPP	General Purpose Processor
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRC	GNU Radio Companion
GSM	Global System for Mobile Communications
IFS	Interframe Space
IoT	Internet of Things
ISI	Intersymbol Interference
ISM	Industrial Scientific and Medical
IVC	Inter-Vehicle Communication
LLC	Logical Link Control
LMS	Least Mean Squares
LNA	Low Noise Amplifier
LS	Least Squares
LTE	Long-Term Evolution
LUT	Lookup Table
MAC	Medium Access Control
MANET	Mobile Ad Hoc Network
MCS	Modulation and Coding Scheme
mmW	Millimeter Wave
MPDU	MAC Protocol Data Unit
MTU	Maximum Transmission Unit
NHTSA	National Highway Traffic Safety Administration
NIC	Network Interface Card
OBU	On-Board Unit
OCB	Outside the Context of a BSS
OFDM	Orthogonal Frequency-Division Multiplexing
OOT	Out-Of-Tree
PAPR	Peak-to-Average Power Ratio
PDU	Protocol Data Unit
PER	Packet Error Rate

PHY	physical layer
PKI	Public Key Infrastructure
PN	pseudonoise
PSDU	PHY Service Data Unit
PSK	Phase-Shift Keying
PyBombs	Python Build Overlay Managed Bundle System
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QPSK	Quadrature Phase-Shift Keying
RF	Radio Frequency
RSS	Received Signal Strength
RSU	Roadside Unit
RTS	Request To Send
RTS/CTS	Request/Clear To Send Frame
SCH	Service Channel
SDR	Software Defined Radio
SIFS	Short Interframe Space
SIMD	Single Instruction Multiple Data
SIR	Signal to Interference Ratio
SNR	Signal to Noise Ratio
STA	Spectral Temporal Averaging
TDMA	Time-Division Multiple Access
TXOP	Transmission Opportunity
UDP	User Datagram Protocol
UHD	USRP Hardware Driver
UMTS	Universal Mobile Telecommunications System
USRP	Universal Software Radio Peripheral
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
VANET	Vehicular Ad Hoc Network
VGA	Variable Gain Amplifier
VLC	Visible Light Communication
VOLK	Vectorized Library of Kernels
WARP	Wireless Open-Access Research Platform
WAVE	Wireless Access in Vehicular Environment
WLAN	Wireless LAN
WSM	Wave Short Message
WSMP	Wave Short Message Protocol

List of Figures

1.1 Our SDR transceiver can be used throughout the whole research process, allowing us to overcome the usual disconnect between theory and practice.	4
2.1 Schematic time-frequency resource usage of a single-carrier and a corresponding multi-carrier scheme.	17
2.2 Spectra of adjacent OFDM subcarriers.	18
2.3 Comparison of an IEEE 802.11a/g frame (left) and a corresponding IEEE 802.11p frame (right) in time-frequency domain.	20
2.4 Comparison of the channel's coherence time (solid, orange line) and the duration of QPSK- $\frac{1}{2}$ frames of different sizes (horizontal lines).	21
2.5 Frame duration depending on the payload size for all MCSs. The dashed horizontal line corresponds to the coherence time of the channel for a relative speed of 100 km/h.	21
3.1 Schematic overview of an ideal Software Radio.	29
3.2 Schematic overview of a direct-conversion Software Defined Radio that up- or down-converts the complex baseband signal to a carrier frequency f_c	30
3.3 Screenshot of GNU Radio Companion, a tool to setup and configure GNU Radio flow graphs.	35
4.1 Screenshot of the transmitter in GNU Radio Companion.	45
4.2 WLAN frames comprise a short and a long training sequence for synchronization; a signal field, containing information about the length and encoding of the frame; and the data symbols, carrying the actual payload. (Reproduced from [23], © 2018 IEEE.)	47
4.3 Typical course of the autocorrelation coefficient during frame start at different SNR levels. Frame detection is triggered once the autocorrelation coefficient exceeds a predefined threshold (dotted line).	49

4.4	Screenshot of the receiver in GNU Radio Companion. The signal for the power p , the autocorrelation a , and the autocorrelation coefficient c are annotated.	50
4.5	Cross-correlation of a frame with the known pattern of the long preamble, as calculated by the receiver to determine OFDM symbol alignment.	51
4.6	Delivery ratio of 435 Byte BPSK- $1/2$ frames at different sensitivity levels.	55
4.7	Screenshot of the transceiver in GNU Radio Companion.	56
4.8	Screenshot of the transceiver's graphical user interface while decoding a QPSK frame. (Reproduced from [31].)	57
4.9	Delivery ratio of 435 Byte frames over an AWGN channel.	58
4.10	Frame delivery ratio between two devices that are connected via cable. The frame size is 133 Byte. (Reproduced from [30], © 2013 IEEE.)	61
4.11	Comparison of the receive performance of a commercial device and our SDR implementation. (Reproduced from [33] with permission, © 2013 ACM.)	63
4.12	Delivery ratio of 95 Byte frames that are sent from a Cohda Wireless MK2 and received with our SDR implementation. (Reproduced from [33] with permission, © 2013 ACM.)	64
4.13	CPU utilization of individual signal processing blocks.	66
4.14	Distribution of the computational load with a fully saturated channel of 64-QAM- $3/4$ encoded frames.	68
4.15	Conceptual overview of our CSMA implementation. (Reproduced from [28] with permission.)	70
4.16	CSMA state machine implemented on the FPGA that controls frame transmission. (Reproduced from [28] with permission.)	72
4.17	Power measurements to verify AIFS timing and to determine channel access delay. (Reproduced from [23], © 2018 IEEE.)	74
4.18	Distribution of the inter-arrival time when using the ACs for <i>Voice</i> and <i>Video</i> . (Reproduced from [23], © 2018 IEEE.)	75
4.19	Inter-arrival time of a Cohda Wireless MK2 (top) and a Unex card with (middle) and without (bottom) TXOP set. (Reproduced from [28] with permission.)	76
4.20	Interoperability test between an SDR and a Unex card (top) and two Unex cards (bottom), showing that each device gets a fair share of the channel. (Reproduced from [28] with permission.)	77
4.21	Power over time for a low-power frame that got amplified by AGC (left) and a high-power frame that got attenuated (right). The power is normalized to the reference power level that the AGC tries to adjust to. (Reproduced from [23], © 2018 IEEE.)	80

4.22	Frame delivery ratio with fixed receive gains and AGC. (Reproduced from [26], © 2015 IEEE.)	81
5.1	Setup of our field test. (Reproduced from [23], © 2018 IEEE.) . . .	87
5.2	Delivery ratio of frames sent from an MK5 and an SDR, using another MK5 as reference receiver. (Reproduced from [23], © 2018 IEEE.)	88
5.3	Received IEEE 802.11p frames per device. (Reproduced from [23], © 2018 IEEE.)	89
5.4	Cumulative number of packets received over time. (Reproduced from [23], © 2018 IEEE.)	89
5.5	Picture of the measurement setup.	94
5.6	GPS trace of the measurement, showing the various environments. (The map data is © OpenStreetMap contributors, the path is reproduced from [25] with permission.)	94
5.7	Performance comparison of receive algorithms in different environments. (Reproduced from [25] with permission.)	95
5.8	Impact of frame size on the receive performance of the selected algorithms. (Reproduced from [25] with permission.)	95
5.9	Impact of speed on the receive performance of the selected algorithms. (Reproduced from [25] with permission.)	96
5.10	We share the trace data and details of the experiment on our website.	97
5.11	Histograms, showing the impact of OFDM interference and a similar level of noise on the subcarrier constellations of an IEEE 802.11p frame. The histograms are normalized so that integration over the bins yields unity.	99
5.12	GNU Radio simulations of the frame delivery ratio of an OFDM frame that is interfered by OFDM frames or a similar level of noise. (Reproduced from [24], © 2017 IEEE.)	101
5.13	Frame delivery ratio of a Unex DCMA-86P2 card when receiving frames that are interfered by another OFDM frame or a similar level of noise. (Reproduced from [24], © 2017 IEEE.)	102
5.14	Overview of our off-the-shelf testbed. (Reproduced from [24], © 2017 IEEE.)	103
5.15	To validate our setup, we reproduce results from the literature and measure the success rate of frames captures at different SIRs.	104
5.16	Frame delivery ratio of a Unex DCMA-86P2 under OFDM interference created with an SDR and our testbed. (Reproduced from [24], © 2017 IEEE.)	105
5.17	Schematic overview of the IEEE 802.11 scrambling algorithm. (Reproduced from [27], © 2015 IEEE.)	108

5.18 Fully covered crossing scenario. (Reproduced from [27], © 2015 IEEE.)	111
5.19 Freeway scenario with blind spot in between attacker receivers. (Reproduced from [27], © 2015 IEEE.)	111
5.20 Impact of the Scrambler Attack in different scenarios. (Reproduced from [27], © 2015 IEEE.)	114
5.21 Tracking probability depending on the type of IEEE 802.11p radio using the scrambler tracking. (Reproduced from [27], © 2015 IEEE.)	115

List of Tables

2.1	Channel access parameters for IEEE 802.11p.	15
4.1	PHY parameters of the supported OFDM modes.	44
4.2	Our transceiver supports all MCSs. (Reproduced from [23], © 2018 IEEE.)	45
4.3	WLAN cards and IEEE 802.11p prototypes used in our interoperability tests. (Reproduced from [23], © 2018 IEEE.)	59
4.4	The most relevant components of our test system for lab measurements. (Reproduced from [30], © 2013 IEEE.)	61
4.5	Most relevant components of our setup. (Reproduced from [28] with permission.)	70
4.6	Channel access parameters for the different access categories [59, Table 9-138]. (Reproduced from [23], © 2018 IEEE.)	73
5.1	Hardware setup and most relevant parameters used in the field test. (Reproduced from [25] with permission.)	92
5.2	Most relevant parameters of our simulations and measurements. . .	100
5.3	Simulation Parameters. (Reproduced from [27], © 2015 IEEE.) . .	112

Bibliography

- [1] M. Campbell, M. Egerstedt, J. P. How, and R. M. Murray, “Autonomous Driving in Urban Environments: Approaches, Lessons and Challenges,” *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 368, no. 1928, Sep. 2010. DOI: 10.1098/rsta.2010.0110.
- [2] S. Singh, “Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey,” National Center for Statistics and Analysis (NHTSA), Tech. Rep. DOT HS 812 115, Feb. 2015.
- [3] ETSI, “Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions,” ETSI, TR 102 638 V1.1.1, Jun. 2009.
- [4] S. Joerer, B. Bloessl, M. Segata, C. Sommer, R. Lo Cigno, A. Jamalipour, and F. Dressler, “Enabling Situation Awareness at Intersections for IVC Congestion Control Mechanisms,” *IEEE Transactions on Mobile Computing*, vol. 15, no. 7, pp. 1674–1685, Jun. 2016. DOI: 10.1109/TMC.2015.2474370.
- [5] 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. DOI: 10.1109/MCOM.2011.5762815.
- [6] R. Stahlmann, A. Tomatis, R. German, and D. Eckhoff, “Multi-hop for GLOSA Systems: Evaluation and Results from a Field Experiment,” in *9th IEEE Vehicular Networking Conference (VNC 2017)*, Torino, Italy: IEEE, Nov. 2017, pp. 175–178. DOI: 10.1109/VNC.2017.8275617.
- [7] S. Santini, A. Salvi, A. S. Valente, A. Pescapè, M. Segata, and R. Lo Cigno, “A Consensus-based Approach for Platooning with Inter-Vehicular Communications and its Validation in Realistic Scenarios,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 3, pp. 1985–1999, Mar. 2017. DOI: 10.1109/TVT.2016.2585018.

- [8] B. Gallagher, H. Akatsuka, and H. Suzuki, "Wireless Communications for Vehicle Safety: Radio Link Performance and Wireless Connectivity Methods," *IEEE Vehicular Technology Magazine*, vol. 1, no. 4, pp. 4–24, Dec. 2006. DOI: 10.1109/MVT.2006.343641.
- [9] L. Cheng, B. E. Henty, R. Cooper, D. D. Stancil, and F. Bai, "A Measurement Study of Time-Scaled 802.11a Waveforms Over The Mobile-to-Mobile Vehicular Channel at 5.9 GHz," *IEEE Communications Magazine*, vol. 46, no. 5, pp. 84–91, May 2008. DOI: 10.1109/MCOM.2008.4511654.
- [10] 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. DOI: 10.1109/mcom.2007.4378329.
- [11] 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. DOI: 10.1109/ICC.2014.6884223.
- [12] 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. DOI: 10.1109/tvt.2011.2164428.
- [13] 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.
- [14] T. Kella, "Decision-Directed Channel Estimation for Supporting Higher Terminal Velocities in OFDM Based WLANs," in *IEEE Global Telecommunications Conference (GLOBECOM 2003)*, San Francisco, CA: IEEE, Dec. 2003, pp. 1306–1310. DOI: 10.1109/GLOCOM.2003.1258449.
- [15] 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, California: ACM, Sep. 2008, pp. 1–6. DOI: 10.1145/1410043.1410045.
- [16] J. Nuckelt and T. Kürner, "MRC Performance Benefit in V2V Communication Systems in Urban Traffic Scenarios," in *6th European Conference on Antennas and Propagation (EUCAP)*, Prague, Czech Republic: IEEE, Mar. 2012, pp. 2311–2315. DOI: 10.1109/EuCAP.2012.6206330.
- [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. DOI: 10.1109/MCOM.2014.6829960.

- [18] 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. DOI: 10.1145/1143549.1143746.
- [19] W. H. Tuttlebee, *Software Defined Radio: Enabling Technologies*. John Wiley & Sons, 2003.
- [20] 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, 59–67, Jan. 2016. DOI: 10.1109/mcom.2016.7378427.
- [21] 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. DOI: 10.1109/ivs.2016.7535356.
- [22] V. Iyer, V. Talla, B. Kellogg, S. Gollakota, and J. R. Smith, "Inter-Technology Backscatter: Towards Internet Connectivity for Implanted Devices," in *ACM SIGCOMM 2016*, Florianopolis, Brazil: ACM, Aug. 2016, pp. 356–369. DOI: 10.1145/2934872.2934894.
- [23] B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "Performance Assessment of IEEE 802.11p with an Open Source SDR-based Prototype," *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, pp. 1162–1175, May 2018. DOI: 10.1109/TMC.2017.2751474.
- [24] B. Bloessl, F. Klingler, F. Missbrenner, and C. Sommer, "A Systematic Study on the Impact of Noise and OFDM Interference on IEEE 802.11p," in *9th IEEE Vehicular Networking Conference (VNC 2017)*, Torino, Italy: IEEE, Nov. 2017, pp. 287–290. DOI: 10.1109/VNC.2017.8275633.
- [25] 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. DOI: 10.1145/2980100.2980104.
- [26] 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. DOI: 10.1109/INFCOMW.2015.7179325.

- [27] 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. DOI: 10.1109/ICCNC.2015.7069376.
- [28] 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. DOI: 10.1145/2643230.2643240.
- [29] 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," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 18, no. 3, pp. 81–90, Jul. 2014. DOI: 10.1145/2721896.2721913.
- [30] B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "Towards an Open Source IEEE 802.11p Stack: A Full SDR-based Transceiver in GNU Radio," in *5th IEEE Vehicular Networking Conference (VNC 2013)*, Boston, MA: IEEE, Dec. 2013, pp. 143–149. DOI: 10.1109/VNC.2013.6737601.
- [31] B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "A GNU Radio Based Receiver Toolkit for IEEE 802.11a/g/p," in *19th ACM International Conference on Mobile Computing and Networking (MobiCom 2013), 5th Wireless of the Students, by the Students, for the Students Workshop (S3 2013), Demo Session*, Miami, FL: ACM, Oct. 2013.
- [32] B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "Decoding IEEE 802.11a/g/p OFDM in Software using GNU Radio," in *19th ACM International Conference on Mobile Computing and Networking (MobiCom 2013), Demo Session*, Miami, FL: ACM, Oct. 2013, pp. 159–161. DOI: 10.1145/2500423.2505300.
- [33] 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. DOI: 10.1145/2491246.2491248.

- [34] B. Bloessl and F. Dressler, “mSync: Physical Layer Frame Synchronization Without Preamble Symbols,” *IEEE Transactions on Mobile Computing*, 2018, available online. DOI: 10.1109/TMC.2018.2808968.
- [35] M. Nabeel, B. Bloessl, and F. Dressler, “Efficient Receive Diversity in Distributed Sensor Networks using Selective Sample Forwarding,” *IEEE Transactions on Green Communications and Networking*, 2017, to appear. DOI: 10.1109/TGCN.2017.2780196.
- [36] F. Klingler, G. S. Pannu, C. Sommer, B. Bloessl, and F. Dressler, “Field Testing Vehicular Networks using OpenC2X,” in *15th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys 2017), Poster Session*, Niagara Falls, NY: ACM, Jun. 2017, pp. 178–178. DOI: 10.1145/3081333.3089322.
- [37] M. Nabeel, B. Bloessl, and F. Dressler, “Selective Signal Sample Forwarding for Receive Diversity in Energy-Constrained Sensor Networks,” in *IEEE International Conference on Communications (ICC 2017)*, Paris, France: IEEE, May 2017. DOI: 10.1109/ICC.2017.7996320.
- [38] M. Nabeel, B. Bloessl, and F. Dressler, “Low-Complexity Soft-Bit Diversity Combining for Ultra-Low Power Wildlife Monitoring,” in *IEEE Wireless Communications and Networking Conference (WCNC 2017)*, San Francisco, CA: IEEE, Mar. 2017. DOI: 10.1109/WCNC.2017.7925504.
- [39] M. Nabeel, B. Bloessl, and F. Dressler, “On Using BOC Modulation in Ultra-Low Power Sensor Networks for Wildlife Tracking,” in *IEEE Wireless Communications and Networking Conference (WCNC 2016)*, Doha, Qatar: IEEE, Apr. 2016, pp. 848–853. DOI: 10.1109/WCNC.2016.7564858.
- [40] M. Segata, B. Bloessl, S. Joerer, C. Sommer, M. Gerla, R. Lo Cigno, and F. Dressler, “Towards Communication Strategies for Platooning: Simulative and Experimental Evaluation,” *IEEE Transactions on Vehicular Technology*, vol. 64, no. 12, pp. 5411–5423, Dec. 2015. DOI: 10.1109/TVT.2015.2489459.
- [41] B. Bloessl and F. Dressler, “mSync - Frames without Preambles,” in *21st ACM International Conference on Mobile Computing and Networking (MobiCom 2015), 4th ACM Software Radio Implementation Forum (SRIF 2015), Poster Session*, Paris, France: ACM, Sep. 2015, pp. 11–11. DOI: 10.1145/2801676.2801678.
- [42] B. Bloessl and F. Dressler, “mSync in Action,” in *21st ACM International Conference on Mobile Computing and Networking (MobiCom 2015), 7th Wireless of the Students, by the Students, for the Students Workshop (S3*

- 2015), *Demo Session*, Paris, France: ACM, Sep. 2015, pp. 24–24. DOI: 10.1145/2801694.2801698.
- [43] F. Dressler, B. Bloessl, M. Hierold, C.-Y. Hsieh, T. Nowak, R. Weigel, and A. Koelpin, “Protocol Design for Ultra-Low Power Wake-Up Systems for Tracking Bats in the Wild,” in *IEEE International Conference on Communications (ICC 2015)*, London, UK: IEEE, Jun. 2015, pp. 6345–6350. DOI: 10.1109/ICC.2015.7249335.
- [44] M. Segata, S. Joerer, B. Bloessl, C. Sommer, F. Dressler, and R. Lo Cigno, “PLEXE: A Platooning Extension for Veins,” in *6th IEEE Vehicular Networking Conference (VNC 2014)*, Paderborn, Germany: IEEE, Dec. 2014, pp. 53–60. DOI: 10.1109/VNC.2014.7013309.
- [45] M. Segata, B. Bloessl, S. Joerer, C. Sommer, M. Gerla, R. Lo Cigno, and F. Dressler, “Towards Inter-Vehicle Communication Strategies for Platooning Support,” in *7th IFIP/IEEE International Workshop on Communication Technologies for Vehicles (Nets4Cars 2014-Fall)*, Saint-Petersburg, Russia: IEEE, Oct. 2014, pp. 1–6. DOI: 10.1109/Nets4CarsFall.2014.7000903.
- [46] S. Joerer, B. Bloessl, M. Huber, A. Jamalipour, and F. Dressler, “Assessing the Impact of Inter-Vehicle Communication Protocols on Road Traffic Safety,” in *20th ACM International Conference on Mobile Computing and Networking (MobiCom 2014)*, *6th Wireless of the Students, by the Students, for the Students Workshop (S3 2014)*, Maui, HI: ACM, Sep. 2014, pp. 21–23. DOI: 10.1145/2645884.2645885.
- [47] S. Joerer, B. Bloessl, M. Huber, A. Jamalipour, and F. Dressler, “Simulating the Impact of Communication Performance on Road Traffic Safety at Intersections,” in *20th ACM International Conference on Mobile Computing and Networking (MobiCom 2014)*, *Demo Session*, Maui, HI: ACM, Sep. 2014, pp. 287–289. DOI: 10.1145/2639108.2641737.
- [48] S. Joerer, B. Bloessl, M. Segata, C. Sommer, R. Lo Cigno, and F. Dressler, “Fairness Kills Safety: A Comparative Study for Intersection Assistance Applications,” in *25th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2014)*, Washington, D.C.: IEEE, Sep. 2014, pp. 1442–1447. DOI: 10.1109/PIMRC.2014.7136395.
- [49] M. Segata, B. Bloessl, C. Sommer, and F. Dressler, “Towards Energy Efficient Smart Phone Applications: Energy Models for Offloading Tasks into the Cloud,” in *IEEE International Conference on Communications (ICC 2014)*, Sydney, Australia: IEEE, Jun. 2014, pp. 2394–2399. DOI: 10.1109/ICC.2014.6883681.

- [50] S. Joerer, M. Segata, B. Bloessl, R. Lo Cigno, C. Sommer, and F. Dressler, "A Vehicular Networking Perspective on Estimating Vehicle Collision Probability at Intersections," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 4, pp. 1802–1812, May 2014. DOI: 10.1109/TVT.2013.2287343.
- [51] M. Segata, B. Bloessl, S. Joerer, F. Dressler, and R. Lo Cigno, "Supporting Platooning Maneuvers through IVC: An Initial Protocol Analysis for the Join Maneuver," in *11th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2014)*, Obergurgl, Austria: IEEE, Apr. 2014, pp. 130–137. DOI: 10.1109/WONS.2014.6814733.
- [52] M. Segata, B. Bloessl, S. Joerer, C. Sommer, R. Lo Cigno, and F. Dressler, "Vehicle Shadowing Distribution Depends on Vehicle Type: Results of an Experimental Study," in *5th IEEE Vehicular Networking Conference (VNC 2013)*, Boston, MA: IEEE, Dec. 2013, pp. 242–245. DOI: 10.1109/VNC.2013.6737623.
- [53] 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, Nov. 2012, pp. 25–32. DOI: 10.1109/VNC.2012.6407441.
- [54] B. Bloessl, S. Joerer, F. Mauroner, and F. Dressler, "Low-Cost Interferer Detection and Classification using TelosB Sensor Motes," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, no. 4, pp. 34–37, Oct. 2012. DOI: 10.1145/2436196.2436215.
- [55] B. Bloessl, S. Joerer, F. Mauroner, and F. Dressler, "Low-Cost Interferer Detection and Classification using TelosB Sensor Motes," in *18th ACM International Conference on Mobile Computing and Networking (MobiCom 2012), Poster Session*, Istanbul, Turkey: ACM, Aug. 2012, pp. 403–405. DOI: 10.1145/2348543.2348595.
- [56] B. Bloessl, S. Joerer, N. Nordin, C. Sommer, and F. Dressler, "SaFIC: A Spectrum Analysis Framework for Interferer Classification in the 2.4 GHz Band," in *31st IEEE Conference on Computer Communications (INFOCOM 2012), Demo Session*, Orlando, FL: IEEE, Mar. 2012.
- [57] G. R. M. Garratt, *The Early History of Radio From Faraday to Marconi*, ser. History of Technology. IET, Jun. 1994.
- [58] X. Wang, S. Mao, and M. X. Gong, "An Overview of 3GPP Cellular Vehicle-to-Everything Standards," *GetMobile: Mobile Computing and Communications*, vol. 21, no. 3, pp. 19–25, Sep. 2017. DOI: 10.1145/3161587.3161593.

- [59] IEEE, “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” IEEE, STD 802.11-2016, Dec. 2016.
- [60] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, “Performance Anomaly of 802.11b,” in *22nd IEEE Conference on Computer Communications (INFOCOM 2003)*, vol. 2, San Francisco, CA: IEEE, Mar. 2003, pp. 836–843.
- [61] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2005.
- [62] Z. Wang, X. Ma, and G. B. Giannakis, “OFDM or Single-Carrier Block Transmissions?” *IEEE Transactions on Communications*, vol. 52, no. 3, pp. 380–394, Mar. 2004. DOI: 10.1109/TCOMM.2004.823586.
- [63] P. Alexander, D. Haley, and A. Grant, “Cooperative Intelligent Transport Systems: 5.9-GHz Field Trials,” *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1213–1235, Jul. 2011. DOI: 10.1109/JPROC.2011.2105230.
- [64] W. Viriyasitavat, M. Boban, H.-M. Tsai, and A. V. Vasilakos, “Vehicular Communications: Survey and Challenges of Channel and Propagation Models,” *IEEE Vehicular Technology Magazine*, vol. 10, no. 2, pp. 55–66, Jun. 2015. DOI: 10.1109/MVT.2015.2410341.
- [65] J. A. Fernandez, D. D. Stancil, and F. Bai, “Dynamic Channel Equalization for IEEE 802.11p Waveforms in the Vehicle-to-Vehicle Channel,” in *48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Monticello, IL: IEEE, Sep. 2010, pp. 542–551. DOI: 10.1109/ALLERTON.2010.5706954.
- [66] 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. DOI: 10.1109/vnc.2015.7385549.
- [67] IEEE, “IEEE Guide for Wireless Access in Vehicular Environments (WAVE) - Architecture,” IEEE, STD 1609.0-2013, Mar. 2014. DOI: 10.1109/IEEESTD.2014.6755433.
- [68] IEEE, “IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Multi-channel Operation,” IEEE, STD 1609.4-2016, Jan. 2016.
- [69] D. Eckhoff, N. Sofra, and R. German, “A Performance Study of Cooperative Awareness in ETSI ITS G5 and IEEE WAVE,” in *10th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2013)*, Banff, Canada: IEEE, Mar. 2013, pp. 196–200. DOI: 10.1109/WONS.2013.6578347.
- [70] ETSI, “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, Jul. 2011.

- [71] ARIB, “700 MHz Band Intelligent Transport Systems,” ARIB, STD T109-v1.3, Jul. 2017.
- [72] J. Heinovski, F. Klingler, F. Dressler, and C. Sommer, “Performance Comparison of IEEE 802.11p and ARIB STD-T109,” in *8th IEEE Vehicular Networking Conference (VNC 2016)*, Columbus, OH: IEEE, Dec. 2016, pp. 1–8. DOI: 10.1109/VNC.2016.7835923.
- [73] ETSI, “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.
- [74] ETSI, “Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specification of Decentralized Environmental Notification Basic Service,” ETSI, EN 302 637-3 V1.2.1, Sep. 2014.
- [75] 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. DOI: 10.1007/978-3-319-15497-8_13.
- [76] 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.
- [77] 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. DOI: 10.4108/eai.24-8-2015.2261106.
- [78] M. Dohler, R. W. Heath, A. Lozano, C. B. Papadias, and R. A. Valenzuela, “Is the PHY Layer Dead?” *IEEE Communications Magazine*, vol. 49, no. 4, pp. 159–165, Apr. 2011.
- [79] H. Rakouth, P. Alexander, A. Brown Jr., W. Kosiak, M. Fukushima, L. Ghosh, C. Hedges, H. Kong, S. Kopetzki, R. Siripurapu, et al., “V2X Communication Technology: Field Experience and Comparative Analysis,” in *FISITA World Automotive Congress*, Beijing, China: Springer, Nov. 2012, pp. 113–129. DOI: 10.1007/978-3-642-33838-0_10.
- [80] 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. DOI: 10.1109/JSAC.2011.110103.

- [81] 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. DOI: 10.1109/ITST.2011.6060156.
- [82] 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. DOI: 10.1109/VNC.2012.6407446.
- [83] 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. DOI: 10.1016/j.vehcom.2014.04.001.
- [84] 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. DOI: 10.1016/j.trc.2014.05.006.
- [85] 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. DOI: 10.1109/TVT.2013.2292519.
- [86] 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. DOI: 10.1109/TITS.2012.2189882.
- [87] 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. DOI: 10.1109/VNC.2016.7835955.
- [88] J. Mitola, "Software Radios: Survey, Critical Evaluation and Future Directions," *IEEE Aerospace and Electronic Systems Magazine*, vol. 8, no. 4, pp. 25–36, Apr. 1993. DOI: 10.1109/62.210638.
- [89] J. Mitola, "The Software Radio Architecture," *IEEE Communications Magazine*, vol. 33, no. 5, pp. 26–38, May 1995. DOI: 10.1109/35.393001.
- [90] S. Shakkottai, T. S. Rappaport, and P. C. Karlsson, "Cross-Layer Design for Wireless Networks," *IEEE Communications Magazine*, vol. 41, no. 10, pp. 74–80, Oct. 2003. DOI: 10.1109/MCOM.2003.1235598.

- [91] Z. Li, T. Braun, and D. C. Dimitrova, "A Passive WiFi Source Localization System based on Fine-grained Power-based Trilateration," in *16th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2015)*, Boston, MA: IEEE, Jun. 2015. DOI: 10.1109/wowmom.2015.7158147.
- [92] H. Qiu, K. Psounis, G. Caire, K. M. Chugg, and K. Wang, "High-Rate WiFi Broadcasting in Crowded Scenarios via Lightweight Coordination of Multiple Access Points," in *17th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2016)*, Paderborn, Germany: ACM, Jul. 2016, pp. 301–310. DOI: 10.1145/2942358.2942372.
- [93] F. Quitin, M. M. U. Rahman, R. Mudumbai, and U. Madhow, "A Scalable Architecture for Distributed Transmit Beamforming with Commodity Radios: Design and Proof of Concept," *IEEE Transactions on Wireless Communications*, vol. 12, no. 3, pp. 1418–1428, Mar. 2013. DOI: 10.1109/TWC.2013.012513.121029.
- [94] M. Schulz, F. Gringoli, D. Steinmetzer, M. Koch, and M. Hollick, "Massive Reactive Smartphone-Based Jamming using Arbitrary Waveforms and Adaptive Power Control," in *10th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec'17)*, Boston, MA: ACM, Jul. 2017, pp. 111–121. DOI: 10.1145/3098243.3098253.
- [95] M. Schulz, D. Wegemer, and M. Hollick, "Nexmon: Build Your Own Wi-Fi Testbeds With Low-Level MAC and PHY-Access Using Firmware Patches on Off-the-Shelf Mobile Devices," in *23rd ACM International Conference on Mobile Computing and Networking (MobiCom 2017)*, *11th ACM International Workshop on Wireless Network Testbeds, Experimental evaluation and Characterization (WiNTECH 2017)*, Snowbird, UT: ACM, Oct. 2017, pp. 59–66. DOI: 10.1145/3131473.3131476.
- [96] J. Kumagai, "Radio Revolutionaries," *IEEE Spectrum*, vol. 44, no. 1, pp. 28–32, Jan. 2007. DOI: 10.1109/MSPEC.2007.273037.
- [97] 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. DOI: 10.1145/1374512.1374532.
- [98] V. G. Bose, "Design and Implementation of Software Radios using a General Purpose Processor," PhD Thesis, Department of Electrical Engineering and Computer Science, Boston, MA, Apr. 1999.

- [99] 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.
- [100] T. Schmid, O. Sekkat, and M. B. Srivastava, "An Experimental Study of Network Performance Impact of Increased Latency in Software Defined Radios," in *13th ACM International Conference on Mobile Computing and Networking (MobiCom 2007), 2nd ACM International Workshop on Wireless Network Testbeds, Experimental evaluation and Characterization (WiNTECH 2007)*, Montréal, Canada: ACM, Sep. 2007, pp. 59–66.
- [101] 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 *19th ACM International Conference on Mobile Computing and Networking (MobiCom 2013), 8th International Workshop on Wireless Network Testbeds Experimental evaluation and Characterization (WiNTECH 2013)*, Miami, FL: ACM, Sep. 2013.
- [102] M. Braun, J. Pendlum, and M. Ettus, "RFNoC: RF Network-on-Chip," in *6th GNU Radio Conference*, Boulder, CO: GNU Radio Foundation, Sep. 2016.
- [103] J. Kim, S. Hyeon, and S. Choi, "Implementation of an SDR System Using Graphics Processing Unit," *IEEE Communications Magazine*, vol. 48, no. 3, pp. 156–162, Mar. 2010. DOI: 10.1109/MCOM.2010.5434388.
- [104] E. Blossom, "GNU Radio: Tools for Exploring the Radio Frequency Spectrum," *Linux Journal*, no. 122, Jun. 2004.
- [105] 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. DOI: 10.1002/9781119057246.ch2.
- [106] 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. DOI: 10.1145/1866739.1866760.
- [107] M. Robert, Y. Sun, T. Goodwin, H. Turner, J. H. Reed, and J. White, "Software Frameworks for SDR," *Proceedings of the IEEE*, vol. 103, no. 3, pp. 452–475, Mar. 2015. DOI: 10.1109/JPROC.2015.2391176.
- [108] P. D. Sutton, J. Lotze, H. Lahlou, S. A. Fahmy, K. E. Nolan, B. Özgül, T. W. Rondeau, J. Noguera, and L. E. Doyle, "Iris: An Architecture for Cognitive Radio Networking Testbeds," *IEEE Communications Magazine*, vol. 48, no. 9, pp. 114–122, Sep. 2010. DOI: 10.1109/MCOM.2010.5560595.

- [109] T. W. Rondeau, N. McCarthy, and T. O'Shea, "SIMD Programming in GNU Radio: Maintainable und User-Friendly Algorithm Optimization with VOLK," in *SDR-WInnComm 2013*, Washington, DC: Wireless Innovation Forum, Jan. 2013, pp. 101–110.
- [110] 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.
- [111] T. Vilches and D. Dujovne, "GNURadio and 802.11: Performance Evaluation and Limitations," *IEEE Network*, vol. 28, no. 5, 27–31, Sep. 2014. DOI: 10.1109/mnet.2014.6915436.
- [112] 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.
- [113] 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. DOI: 10.1145/2998373.2998443.
- [114] P. Fuxjaeger and S. Ruehrup, "Validation of the NS-3 Interference Model for IEEE802.11 Networks," in *8th IFIP Wireless and Mobile Networking Conference (WMNC 2015)*, Munich, Germany: IEEE, Oct. 2015, pp. 216–222. DOI: 10.1109/wmnc.2015.40.
- [115] J. Lee, W. Kim, S.-J. Lee, D. Jo, J. Ryu, T. Kwon, and Y. Choi, "An Experimental Study on the Capture Effect in 802.11a Networks," in *13th ACM International Conference on Mobile Computing and Networking (MobiCom 2007), 2nd ACM International Workshop on Wireless Network Testbeds, Experimental evaluation and Characterization (WiNTECH 2007)*, Montreal, Canada: ACM, Sep. 2007, pp. 19–26.
- [116] T. Schmidl and D. Cox, "Robust frequency and timing synchronization for OFDM," *IEEE Transactions on Communications*, vol. 45, no. 12, pp. 1613–1621, 1997. DOI: 10.1109/26.650240.
- [117] 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. DOI: 10.1109/GLOCOM.2003.1258308.

- [118] E. Sourour, H. El-Ghoroury, and D. McNeill, "Frequency Offset Estimation and Correction in the IEEE 802.11a WLAN," in *60th IEEE Vehicular Technology Conference (VTC2004-Fall)*, Los Angeles, CA: IEEE, Sep. 2004, pp. 4923–4927. DOI: 10.1109/VETECF.2004.1405033.
- [119] F. Tosato and P. Bisaglia, "Simplified Soft-Output Demapper for Binary Interleaved COFDM with Application to HIPERLAN/2," in *IEEE International Conference on Communications (ICC 2002)*, New York, NY: IEEE, May 2002, pp. 664–668.
- [120] G. Pei and T. R. Henderson, "Validation of OFDM Error Rate Model in ns-3," Boeing Research & Technology, Seattle, WA, Tech. Rep., 2010.
- [121] J. R. Gutierrez-Agullo, B. Coll-Perales, and J. Gozalvez, "An IEEE 802.11 MAC Software Defined Radio Implementation for Experimental Wireless Communications and Networking Research," in *IFIP Wireless Days Conference 2010*, Venice, Italy: IEEE, Oct. 2010, pp. 1–5.
- [122] A. Puschmann, M. A. Kalil, and A. Mitschele-Thiel, "A Flexible CSMA based MAC Protocol for Software Defined Radios," *Frequenz: Journal of RF-Engineering and Telecommunications*, vol. 6, no. 66, pp. 261–268, Oct. 2012. DOI: 10.1515/freq-2012-0048.
- [123] P. Di Francesco, S. McGettrick, U. K. Anyanwu, J. C. O'Sullivan, A. B. MacKenzie, and L. A. DaSilva, "A Split MAC Approach for SDR Platforms," *IEEE Transactions on Computers*, vol. 64, no. 4, pp. 912–924, Apr. 2015. DOI: 10.1109/TC.2014.2308197.
- [124] ETSI, "Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band," ETSI, EN 302 663 V1.2.1, Jul. 2013.
- [125] Maxim Integrated, "MAX2828/MAX2829 Single-/Dual-Band 802.11a/b/g World-Band Transceiver ICs," Maxim Integrated, Datasheet Rev 0, Oct. 2004.
- [126] ETSI, "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.
- [127] A. Kwoczek, Z. Raida, J. Láčák, 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. DOI: 10.1109/VNC.2011.6117107.
- [128] Cohda Wireless, "DSRC Field Trials," Cohda Wireless, White Paper, Aug. 2017.

- [129] M. Takai, J. Martin, and R. Bagrodia, "Effects of Wireless Physical Layer Modeling in Mobile Ad Hoc Networks," in *2nd ACM International Symposium on Mobile Ad hoc Networking and Computing (MobiHoc 2001)*, Long Beach, CA: ACM, Oct. 2001, pp. 87–94. DOI: 10.1145/501426.501429.
- [130] R. Patidar, S. Roy, T. R. Henderson, and A. Chandramohan, "Link-to-System Mapping for ns-3 Wi-Fi OFDM Error Models," in *Workshop on NS-3 (WNS3 2017)*, Porto, Portugal: ACM, 2017, pp. 31–38. DOI: 10.1145/3067665.3067671.
- [131] C. Shahriar, M. L. Pan, M. Lichtman, T. C. Clancy, R. McGwier, R. Tandon, S. Sodagari, and J. H. Reed, "PHY-Layer Resiliency in OFDM Communications: A Tutorial," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 292–314, 2015. DOI: 10.1109/COMST.2014.2349883.
- [132] S. Papanastasiou, J. Mittag, E. G. Strom, and H. Hartenstein, "Bridging the Gap between Physical Layer Emulation and Network Simulation," in *IEEE Wireless Communications and Networking Conference (WCNC 2010)*, Sydney, Australia: IEEE, Apr. 2010, pp. 1–6. DOI: 10.1109/WCNC.2010.5506341.
- [133] M. Vanhoef and F. Piessens, "Advanced Wi-Fi Attacks Using Commodity Hardware," in *30th Annual Computer Security Applications Conference (ACSAC 2014)*, New Orleans, LA: ACM, Dec. 2014, pp. 256–265. DOI: 10.1145/2664243.2664260.
- [134] D. Eckhoff and C. Sommer, "Driving for Big Data? Privacy Concerns in Vehicular Networking," *IEEE Security & Privacy*, vol. 12, no. 1, pp. 77–79, Feb. 2014. DOI: 10.1109/MSP.2014.2.
- [135] M. Gruteser and B. Hoh, "On the Anonymity of Periodic Location Samples," in *Security in Pervasive Computing*, Boppard, Germany: Springer, Apr. 2005, pp. 179–192.
- [136] IEEE, "IEEE Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages," IEEE, STD 1609.2-2016, Jan. 2016.
- [137] ETSI, "Intelligent Transport Systems (ITS); Security; Trust and Privacy Management," ETSI, TS 102 941 V1.1.1, Jun. 2012.
- [138] B. Danev, D. Zanetti, and S. Čapkun, "On Physical-Layer Identification of Wireless Devices," *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, pp. 1–28, Nov. 2012. DOI: 10.1145/2379776.2379782.
- [139] N. Patwari and S. K. Kasera, "Robust Location Distinction Using Temporal Link Signatures," in *13th ACM International Conference on Mobile Computing and Networking (MobiCom 2007)*, Montréal, Canada: ACM, Sep. 2007, pp. 111–122.

- [140] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless Device Identification with Radiometric Signatures," in *14th ACM International Conference on Mobile Computing and Networking (MobiCom 2008)*, San Francisco, CA: ACM, Sep. 2008, pp. 116–127. DOI: 10.1145/1409944.1409959.
- [141] M. Edman and B. Yener, "Active Attacks Against Modulation-based Radiometric Identification," Rensselaer Polytechnic Institute, Department of Computer Science, Tech. Rep. 09-02, Aug. 2009.
- [142] R. W. Klein, M. A. Temple, and M. J. Mendenhall, "Application of Wavelet-Based RF Fingerprinting to Enhance Wireless Network Security," *Journal of Communications and Networks*, vol. 11, no. 6, pp. 544–555, Dec. 2009.
- [143] O. Ureten and N. Serinken, "Wireless Security Through RF Fingerprinting," *Canadian Journal of Electrical and Computer Engineering*, vol. 32, no. Winter, pp. 27–33, 2007. DOI: 10.1109/CJECE.2007.364330.
- [144] T. Kohno, A. Broido, and K. Claffy, "Remote Physical Device Fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, May 2005. DOI: 10.1109/TDSC.2005.26.
- [145] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. V. Randwyk, and D. Sicker, "Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting," in *15th USENIX Security Symposium*, Vancouver, Canada: USENIX, Jul. 2006, pp. 167–178.
- [146] D.-W. Lim, S.-J. Heo, and J.-S. No, "An Overview of Peak-to-Average Power Ratio Reduction Schemes for OFDM Signals," *Journal of Communications and Networks*, vol. 11, no. 3, pp. 229–239, Jun. 2009. DOI: 10.1109/JCN.2009.6391327.
- [147] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens, "Why MAC Address Randomization is not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms," in *11th ACM Asia Conference on Computer and Communications Security (ASIACCS 2016)*, Xi'an, China: ACM, May 2016, pp. 413–424. DOI: 10.1145/2897845.2897883.
- [148] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 3–15, Jan. 2011. DOI: 10.1109/TMC.2010.133.
- [149] ETSI, "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service," ETSI, EN 302 637-2 V1.3.0, Aug. 2013.

- [150] J. Edmonds, "Maximum Matching and a Polyhedron with 0, 1-vertices," *Journal of Research of the National Bureau of Standards*, vol. 69B, no. 1-2, pp. 125–130, Jan. 1965.
- [151] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Artech House Boston, 1999.
- [152] B. Wiedersheim, Z. Ma, F. Kargl, and P. Papadimitratos, "Privacy in Inter-Vehicular Networks: Why simple pseudonym change is not enough," in *7th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2010)*, Kranjska Gora, Slovenia, Feb. 2010.
- [153] L. Huang, K. Matsuura, H. Yamane, and K. Sezaki, "Enhancing Wireless Location Privacy Using Silent Period," in *IEEE Wireless Communications and Networking Conference (WCNC 2005)*, New Orleans, LA, Mar. 2005.
- [154] SAE, "Dedicated Short Range Communications (DSRC) Message Set Dictionary," SAE, STD J2735-200911, Nov. 2009.