Timings Matter: Standard Compliant IEEE 802.11 Channel Access for a Fully Software-based SDR Architecture

Bastian Bloessl[†]
bloessl@ccs-labs.org

Andre Puschmann[*]
andre.puschmann@tu-ilmenau.de

Christoph Sommer[†]
sommer@ccs-labs.org

Falko Dressler[†]
dressler@ccs-labs.org

[†] Distributed Embedded Systems Group, University of Paderborn, Germany
[*] Integrated Communication Systems Group, Ilmenau University of Technology, Germany

We present a solution for enabling standard compliant channel access for a fully software-based Software Defined Radio (SDR) architecture. With the availability of a GNU Radio implementation of an Orthogonal Frequency Division Multiplexing (OFDM) transceiver, there is substantial demand for standard compliant channel access. It has been shown that implementation of CSMA on a host PC is infeasible due to system-inherent delays. The common approach is to fully implement the protocol stack on the FPGA, which makes further updates or modifications to the protocols a complex and time consuming task. We take another approach and investigate the feasibility of a fully software-based solution and show that standard compliant broadcast transmissions are possible with marginal modifications of the FPGA. We envision the use of our system for example in the vehicular networking domain, where broadcast is the main communication paradigm. We show that our SDR solution exactly complies with the IEEE 802.11 Distributed Coordination Function (DCF) as well as Enhanced Distributed Channel Access (EDCA) timings. We were even able to identify shortcomings of commercial systems and prototypes.

I. Introduction

Software Defined Radios (SDRs) have become one of the most powerful tools when it comes to experimental and proof-of-concept solutions of new wireless technologies [16]. Even more importantly, the use of SDR proved hugely beneficial to general wireless networking research as well. The main advantage is that freely programmable radios provide access to all data including the physical layer and allow to study current protocol standards in greater detail, to study new protocol variants, or even to study completely new protocols.

In this paper,[1] we are interested in the IEEE 802.11 protocol family, particularly in Orthogonal Frequency Division Multiplexing (OFDM)-based physical layers. These are defined in IEEE 802.11a/g – and also IEEE 802.11p, which has been designed for Inter-Vehicle Communication (IVC) [2, 14].

In the automotive domain, SDRs technology helps to develop a new era of future proof radios. A completely software driven radio can be updated to next generation communication technologies and protocols via simple firmware updates. Given the rather long product cycles in the automotive industry, this may be the only option to keep track with changing standards or just enable such changes [11].

Following [17], we can distinguish two kinds of SDR architectures based on how the *physical layer* is implemented: First of all, we have what we'd like to call *software-only* solutions. Here, the most common approach is to use a system like GNU Radio where the complete physical layer is implemented on a general purpose processor, i.e., a host PC [9]. This approach gives the user the best flexibility and also allows even newcomers to the field to quickly set up the entire communication system. On the other hand, this architecture does not allow to quickly react on received signals since streaming the samples to the host PC and decoding on a CPU running a non-real-time operating system introduces significant delays and non-determinism expressed in delay variations [18]. Therefore, conceptually simple tasks like conforming to Carrier Sense Multiple Access (CSMA) timing constraints become infeasible.

[1] An earlier version of the paper was published in [6]

The second approach is what we call *hardware* solutions, where the physical layer is implemented directly on a Field-Programmable Gate Array (FPGA) [15] or very close to the hardware as with Digital Signal Processors (DSPs). Using this architecture, timings are deterministic and delay requirements of modern wireless standards can be met, however, reprogramming the system becomes complex and time consuming. Another drawback is that an implementation of a wireless standard is specific to an SDR platform.

Given the strong need to support SDR-based solutions especially in the IVC application domain, we investigated the aforementioned limitations in greater detail.

In many applications, for example in the vehicular domain using the IEEE 802.11p protocol, broadcasting is the main communication paradigm. Considering broadcasting only, we have less timing constraints. In particular, we do not have to cope with dependent transmissions like Acknowledgement Frames (ACKs) or Request/Clear To Send Frames (RTS/CTS). Relevant timings are therefore Clear Channel Assessment (CCA) and the CSMA backoff mechanism.

In previous work, we developed a GNU Radio-based transceiver system for use with the Ettus N210 SDR [7, 8]. We now extended this work to also perform standard compliant channel access with minor modifications of the FPGA, while maintaining all benefits of the software based architecture. We further implemented Enhanced Distributed Channel Access (EDCA) functionality, which is part of the IEEE 802.11e amendment. Our solution is particularly useful for research in the vehicular networking context, but also implements IEEE 802.11a/g functionality and, thus, can also be used for WiFi.

To verify our channel access mechanism, we conducted an extensive set of timing measurements and verified fair share with IEEE 802.11a/g/p Commercial Off-The-Shelf (COTS) hardware based on a standard Atheros chipset. We were not only able to show that out SDR solution exactly complies with the IEEE 802.11 Distributed Coordination Function (DCF) and EDCA timings but were also able to identify shortcomings of the ATH5K driver and the Cohda MK2 system.

Our main contributions can be summarized as follows:

- We present a way to ensure standard compliant carrier sensing for broadcast packets for a software-only SDR platform.
- We make all the code available[2] as Open Source.
- Our CSMA implementation supports both the standard DCF as well as the EDCA QoS extensions.
- We tested and verified the compliance with Commercial Off-The-Shelf (COTS) hardware and IEEE 802.11p prototypes.

II. Related Work

It is well known that meeting the latency requirements of state of the art wireless standards is among the biggest problems for SDR platforms. Especially the challenges of implementing the IEEE 802.11 protocol have been well studied in the literature. In this context, we are interested in CSMA implementations on different SDR architectures.

The possible design space for architectures spans three subgroups: hardware-based, software-based, and hybrid solutions. Hardware-based solutions implement the entire physical and MAC layer on top of programmable hardware, such as FPGAs or DSPs. Their main advantage is the offered performance and deterministic timing, but they are expensive (since the hardware needs to be more powerful) and programming is complex and time consuming. The WARP [15] platform is a prominent example that follows this architecture. Recently, the project has released a *IEEE 802.11 Reference Design* providing a standard compliant FPGA implementation of the IEEE 802.11g standard. This design implements the DCF, i.e., does not support QoS extensions of IEEE 802.11e, but is able to meet all timing constraints posed by the standard.

In contrast to hardware-based solutions, software-only approaches offer a high degree of flexibility. With software approaches, the whole physical layer is implemented on a host PC and only some signal processing tasks like sample rate conversion or channel filters are implemented on the FPGA. The fundamental challenges are system-inherent latencies between the host PC and the SDR as well as computational performance and non-deterministic delays introduced by running the physical layer on a non-real-time operating system [17]. Therefore, conventional architectures with USB or Gigabit Ethernet interfaces are not able to meet the latency requirements of the IEEE 802.11 [18].

As a consequence, the Sora project [19] has combined a radio front-end that connects to the host com-

[2] `http://www.wime-project.net/`

puter via the high-speed low-latency PCIe bus. This ensures that the timing requirements of IEEE 802.11 protocol can be satisfied, even though the physical and MAC layer are implemented on the host. Sora uses highly optimized threads, look up tables, process scheduling, and caching of premodulated ACKs to meet the timing requirements. However, the high degree of optimization and the fact that the Sora platform is not Open Source limits its application for prototyping new protocol variants.

In [12], the authors present a complete IEEE 802.11 DCF implementation in software, running on the host PC. The authors show that the implementation exceeds the CSMA timings as defined by the standard by three orders of magnitude and, thus, the implementation is not standard compliant.

Hybrid approaches aim at combining advantages of both hardware and software strategies. The idea is to implement time critical functionality in hardware to achieve deterministic timing, while leaving the non time critical system parts in software for higher flexibility. The split functionality architecture of Nychis et al. [17] was among the first functional implementations of the hybrid concept. Their prototype featured a GNU Radio implementation of a CSMA MAC that realizes carrier sensing, backoff processing and dependent packet processing inside the FPGA of an first generation Ettus USRP. Recently, [10] presented a similar architecture for an embedded USRP. However, both approaches do not aim to implement standard compliant channel access, since both do not transmit IEEE 802.11 frames, implement a very simplified backoff algorithm and consequently are not interoperable with COTS hardware.

In this paper, we follow the split functionality approach and present a new architecture that allows to implement standard compliant channel access for IEEE 802.11 broadcast transmissions.

III. IEEE 802.11 Wireless LAN

Since its initial release in 1997, the IEEE 802.11 Wireless LAN (WLAN) standard [4] has grown to encompass several access technologies and many different amendments. In the context of this paper, we refer to IEEE 802.11 as only the subset that is directly relevant for our system. Referring to these sections as the standard amendments that introduced them, these are

- IEEE 802.11a, the specification of an OFDM physical layer,
- IEEE 802.11e, the MAC layer amendment defining QoS enhancements for access prioritization,
- IEEE 802.11p, which specifies amendments for operation in the 5.9 GHz Dedicated Short Range Communications (DSRC) band; in particular, specification of the Outside the Context of a BSS (OCB) mode and associated QoS parameters.

Furthermore, we restrict the discussion to the broadcast case, but have an in-depth look. In general, IEEE 802.11 specifies a contention based channel access algorithm, the well known CSMA/CA, along with an EDCA procedure that adds support for multiple Access Categories (ACs) to provide QoS functionalities.

IEEE 802.11 channel access works as follows: Whenever a station is not sending, it senses the channel to determine whether the medium is busy. Carrier sensing is divided in virtual and physical carrier sensing and the channel is declared busy if either method senses it busy.

Virtual carrier sensing relies on the *duration* field of overheard frames. The sender of a frame may set the duration field to a timeslot during which the channel is virtually busy. This timeslot covers the duration of the frame and all its dependent transmissions, like ACKs following a data frame or the actual transmission following RTS/CTS.

Physical carrier sensing is divided in preamble detection and energy detection. The preamble consists of a repeating pattern that can be recognized even for low energy signals.

Once a frame is detected, the receiver tries to decode the signal field that follows the preamble. The signal field is encoded in the most robust modulation and coding scheme and contains the length and encoding of the following data. If the receiver is able to decode the signal field, it senses the channel busy for the duration of the frame (which can be derived from the data in the signal field), even if the data can not be decoded or even if the energy level drops. The second variant of physical carrier sensing is energy detection, where the channel is declared busy if the received power exceeds a given threshold. According to the standard, the medium has to be sensed busy for power levels above −65 dBm.

The output of the carrier sensing module is used by the CSMA state machine that decides when a frame may be sent. Considering broadcasts, a station may transmit immediately if the channel has been observed to be idle for the duration of an Arbitration Inter Frame Space (AIFS). The length of the AIFS depends on the AC of the frame and is shorter for higher priorities. It is defined as an integer multiple of *slots* (AIFSN) and based on the Short Interframe Space (SIFS) as $\text{AIFS}[\text{AC}] = \text{SIFS} + \text{AIFSN}[\text{AC}] \cdot \text{slot time}$.

Parameter	Value	Reference
slot time	13 µs	[4, Table 18-17]
SIFS	32 µs	[4, Table 18-17]
aCW_{min}	15	[4, Table 18-17]
TXOP	0	[4, Table 8-106]

Table 1: Selected CSMA timing parameters.

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

Table 2: Default EDCA parameters used for all our experiments in this paper [4, Table 8-106].

Table 1 gives an overview of relevant timings.

Any station that unsuccessfully tries to access the channel enters a random backoff period, i.e., it delays sending the frame. The length of this backoff period, measured in an integer multiple of *slots*, is chosen uniformly in the interval $[0; \text{CW}]$. The upper limit CW of this contention window starts at CW_{min} and increases for every unsuccessful (e.g., collided) transmission. Like the AIFS, the value of the contention window depends on the AC of a frame. CW_{min} is very small for high priority frames and larger for lower priority frames. Thus, a node's MAC waiting for a free channel for a packet with high priority (hence, short AIFS and, on average, lower number of backoff slots) will most likely be able to access the channel sooner than another station's MAC that wants to send a low priority frame.

Furthermore, each AC corresponds to a separate MAC layer queue, each with its own backoff timer, that compete for channel access. If multiple queues are trying to access the channel at the same time, the conflict is resolved internally using a *virtual collision* mechanism, where the frame with higher priority is sent and the other queue enters a backoff. Note that this is the sole possibility of a detected collision, since broadcasts are not acknowledged, collisions are generally not detected and the congestion window remains constant.

Once a frame is sent, the device enters a post-TX backoff that works similar to the normal backoff procedure. The post-TX backoff ensures that the device does not capture the channel and starts sending packets spaced by AIFSs once it won the contention, potentially causing starvation of other devices. This mechanism is crucial in order to guarantee fairness.

In general, the 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 the IEEE WAVE [3] and ETSI ITS G5 [5] standard agree on parameters listed in Table 2. Consequently, we employ this parameter set for all presented measurements.

Apart from ACs, IEEE 802.11e also introduced the concept of Transmission Opportunity (TXOP) limits. A TXOP limit is specified by the AC and defines a time for which a station can occupy the channel once it won the contention and is allowed to access the medium. TXOPs were introduced to solve the well known *rate anomaly* of IEEE 802.11 [13].

This anomaly arises since the normal DCF provides fairness on packet level and not on time level, i.e., a station with a bad connection has to use a more robust encoding, resulting in a longer frame and, thus, occupies the channel for a longer time. Therefore, a single station with a bad connection can considerably degrade network performance. However, IEEE 802.11p sets the TXOP of all ACs to zero, effectively disabling the mechanism and falling back to packet based fairness.

Finally, a notable feature introduced by IEEE 802.11p is its novel operation mode, the Outside the Context of a BSS (OCB) mode, in which no authentication or association is performed by the MAC sublayer. Instead, stations transmit (and receive) frames with a wildcard BSSID value and, thus, avoid the need for any signaling prior to exchanging information, supporting the high dynamics and short contact times of vehicular networks.

IV. Concept and Implementation

In the following, we detail our implementation, which extends our Open Source IEEE 802.11a/g/p OFDM WiFi transceiver [7, 8] with CSMA functionality. The transceiver is implemented based on GNU Radio on the software side and on the Ettus N210 on hardware side.

An overview of the system can be seen in Figure 1, where *WIME* depicts our physical layer implementation of the WiFi standard. We follow the split functionality approach and move time critical functionality, i.e., CCA and backoff logic into the FPGA while keeping all physical layer processing in software, maintaining all advantages of a software-only SDR platform. The most important components of our testbed

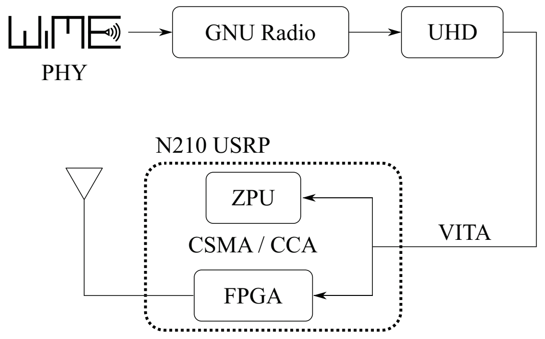

Figure 1: Overview of the system architecture.

Component	Type
GNU Radio	Version 3.7
UHD	Version 003 006 001
SDR	Ettus Research N210 revision 4
Daughterboard	XCVR2450
Xilinx ISE	Version 12.3

Table 3: Relevant components of our testbed.

are listed in Table 3. While we base our implementation on the most recent versions of GNU Radio and the USRP Hardware Driver (UHD), we used an older version of the Xilinx ISE to compile the FPGA image because we experienced timing problems when using the most recent version.

To implement CSMA functionality we had to extend all layers of the system, i.e., our transceiver (implemented based on GNU Radio), the UHD (used to interface the SDR), the FPGA image of the SDR, and the firmware of the ZPU soft core running on the FPGA.

At first, we created a new GNU Radio block that provides four inputs for the ACs. The sole functionality of the new block is to tag the data packets with CSMA metadata, i.e., AIFS and the random number of backoff and post-TX backoff slots. Hence, all random numbers are generated on the host. The tags are propagated through the transmit chain until they reach the *USRP sink* block that orchestrates the SDR through the UHD. Here, the CSMA parameters are extracted from the annotated tags and added to the VITA 49 header to make them accessible from within the FPGA. The VITA 49 packet format [20] is used to transport samples between the host and the SDR.

On the FPGA, the CSMA parameters are used to configure the CSMA state machine. The samples are buffered until the state machine triggers their transmission. Note that currently we maintain a single queue for all frames on the host side as opposed to one queue per AC. Multi-queue support would require to implement queues on the FPGA, which adds functionality, but does not pose further timing challenges. Also, the available memory for such queues on the FPGA is very limited.

Finally, we extended the firmware of the ZPU soft core to provide a control interface to set data that does not change per packet, like slot time and the CCA threshold used for energy detection.

IV.A. Clear Channel Assessment

Since virtual carrier sensing is not relevant in the broadcast case due to the lack of dependent transmission we only have to consider physical carrier sensing, i.e., preamble and energy detection. We limited our implementation to energy detection, since with preamble detection we have to demodulate and decode at least the signal field on the FPGA. This would however, require considerable functionality on the FPGA, including frame detection, synchronization, demodulation, and a Viterbi decoder. Having these physical layer algorithms in hardware would contradict the software only approach and exceed the resources of the used Ettus N210.

For energy detection, we pipe all samples from the RX 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 that we implemented on the ZPU softcore. For our tests and evaluations we used a moving average of eight samples, corresponding to a time window of 0.8 µs at 10 MHz.

IV.B. CSMA State Machine

When a frame is to be transmitted, its samples are transfered to the SDR and buffered in memory until the CSMA state machine triggers its transmission. Each frame is annotated with its AIFS duration and random variables for backoff and post-TX backoff. An overview of the state machine is depicted in Figure 2. 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 sensed busy while in the AIFS&GO state, the normal backoff procedure starts by switching to the AIFS state, otherwise the frame is sent immediately.

We stay in AIFS until the medium remains free for an AIFS without interruption. Once the medium

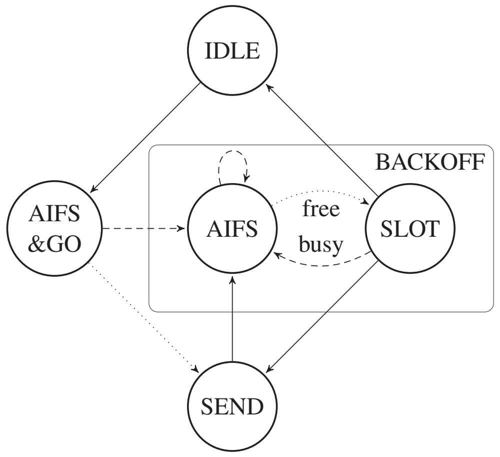

Figure 2: CSMA state machine implemented on the FPGA that controls frame transmission.

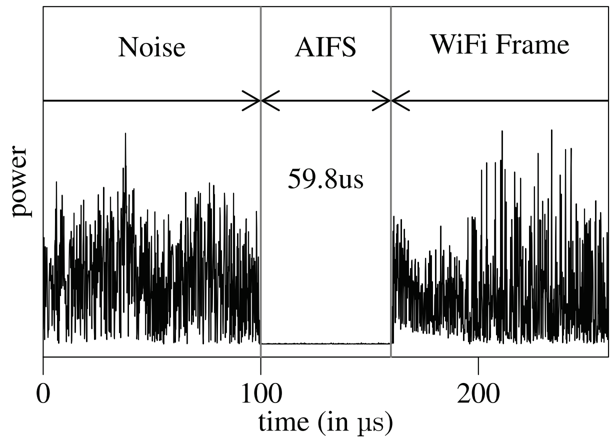

Figure 3: Power measurements to verify AIFS timing and to determine channel access delay.

was free for AIFS, 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. 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. However, since we also backoff even though no frame might be buffered, we added a check just before sending and switch back to IDLE if this is not the case. Hence, the transition from SLOT back to IDLE.

V. Evaluation

Verification and evaluation of our implementation has been performed in three steps: correct energy threshold for CCA, timings between consecutive packets, and interoperability with commercial products and prototypes.

V.A. Energy Threshold for CCA

For energy detection, we have to set a threshold that defines the power level at which the channel is sensed busy. In our case, the threshold does not define an absolute power level, but is expressed in the raw values that are output by the A/D converter – 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. We created an application to monitor the output of the signal power module on the FPGA. With this monitoring application we set the threshold between the power level of the noise floor and a frame transmission.

To investigate the timing of our implementation, we used a third SDR to monitor the power level of the channel over time. We synchronized the clock of the monitoring device with clock of the device that performs channel sensing. This way we prevent a relative clock drift between the devices so that the sampling frequencies of the device that monitors the power levels and the signal strength module of the device that performs carrier sensing are in sync. Furthermore, we set all backoffs to zero so that the channel is accessed deterministically after the AIFS period.

In a first experiment, we used a second SDR to block the channel so that the frame transmission is blocked reliably. The results of this experiment are depicted in Figure 3. During the first 100 µs, the channel is blocked with random noise. Furthermore, we see that frame transmission is delayed even after the channels 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 well explained by 1 µs RX-TX turn around time of the MAX2829 transceiver IC [1] that is used on our RF frontend 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 with different interframe 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 1.8 µs can be compensated by subtracting it from the AIFS, resulting in a more precise timing. We did, however, not compensate for that, since CCA delay and RX-TX turnaround delays are already considered in the standard and are part of the calculations for the slot time.

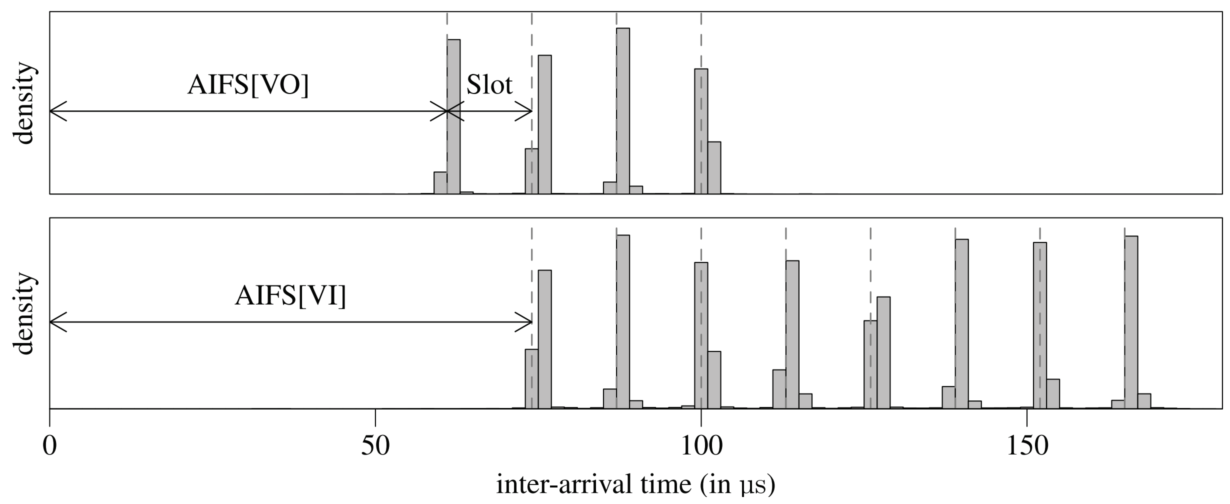

Figure 4: Distribution of the inter-arrival time when using the ACs for *voice* and *video*.

V.B. Inter-arrival Time

As a next step we tested the basic CSMA functionality. A convenient way to do that 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 – where the post-TX backoff lasts for AIFS plus a random number of slots between 0 and CW_{min}. Thus, ideally, the inter-arrival times are assumed to be discrete and equally distributed over the CW range.

As receiver we used a Linux PC with a Unex DCMA-86P2 IEEE 802.11p-capable card. This card is based on an Atheros chipset that is supported by the *ath5k* Linux driver. Since there is no IEEE 802.11p stack available yet, we had to make some changes to the kernel and the driver to achieve physical connectivity: We extended the regulatory domain with the Intelligent Transportation System (ITS) channels in the 5.9 GHz band and, thus, allowed the card to tune to those frequencies. Furthermore, we had to switch to *half rate* mode, i.e., switch from 20 MHz to 10 MHz bandwidth.

In order to measure the inter-arrival time, we extended the RX 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 measurements with the *voice* and *video* ACs can be seen in Figure 4.

Each histogram is based on more than 30 000 frames and, thus, samples of the measured inter-arrival times. 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 limited clock resolution of the Linux PC that we used to measure the inter-arrival time.

We can clearly see that the card waits for the mandatory AIFS duration plus a random number of slots. This verifies the slot time, the AIFS duration, the CW_{min} setting, and shows that the number of backoff slots is approximately uniform as expected.

Additionally, 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).

V.C. Interoperability

In a final experiment, we verified interoperability in terms of fairness with IEEE 802.11p prototypes. We started with the Cohda Wireless MK2, which has been used for major field trials in the U.S. and in Europe.[3] It provides an IEEE 802.11p radio implemented on an FPGA that ships with all the firmware and software of a complete IEEE WAVE stack; we used firmware revision 4.0.14615. To assert that we configured the device correctly, we conducted the same measurements as for the SDR. The results for the MK2 are plotted in Figure 5 (top plot).

Clearly, the distribution does not correspond with the expected results. It turned out that the Cohda MK2 does not implement the post-TX correctly and sends consecutive packets deterministically after the AIFS period. We configured different ACs and observed dif-

[3] `http://www.cohdawireless.com/`

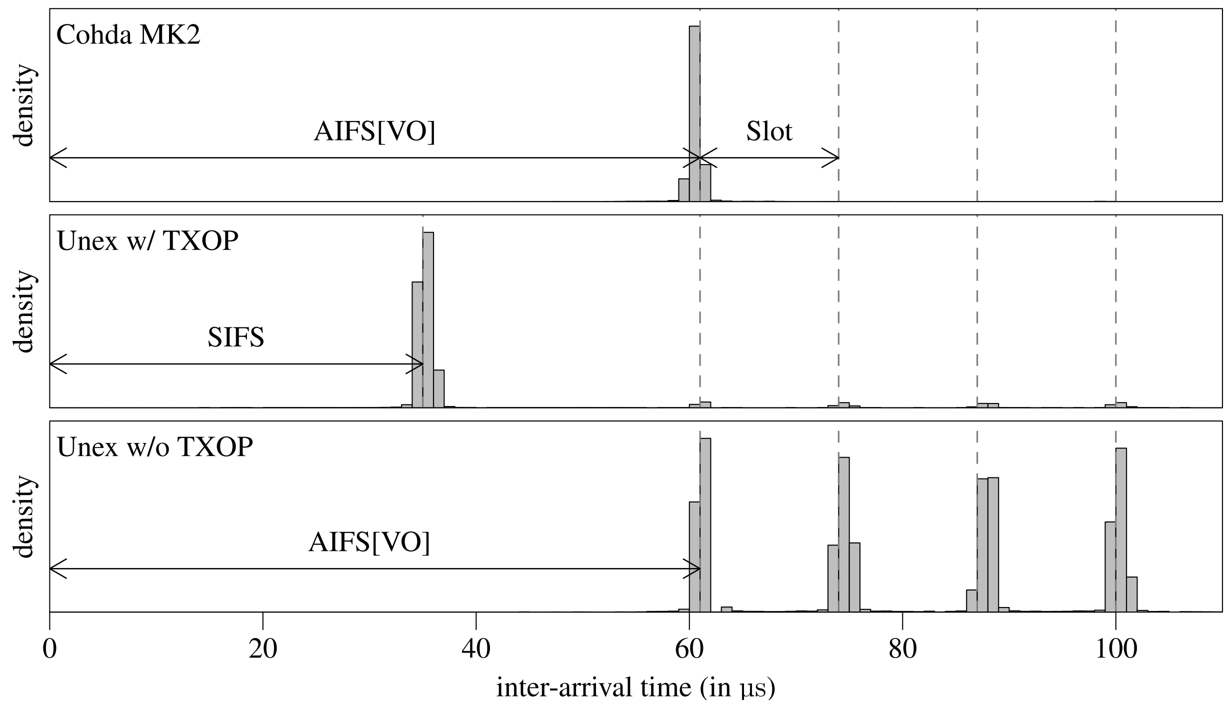

Figure 5: Distribution of inter-arrival times of Cohda, MK2, and Unex card with and without TXOP.

ferent AIFS, indicating that the QoS queues are indeed used and parameterized correctly. However, the post-TX backoff did not work for any ACs. The bug has been reported and confirmed by Cohda Wireless, but deemed impossible to fix. Since the post-TX backoff is a crucial part of the CSMA mechanism, especially when it comes to a saturated channel and since we wanted to saturated the channel for our measurements, we excluded the MK2 from additional experiments.

Instead, we switched to the Unex cards that we already used for our initial measurements. However, for the fairness test physical connectivity is not sufficient: we need standard compliant and correctly parameterized MAC functionality. This required further modifications of the *ath5k* driver. We had to instantiate and configure the QoS queues, set the slot time, and the SIFS duration.

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 the WiFi card operates in monitor mode, the Radiotap header allows to annotate metadata (like modulation and coding scheme and signal power) to a frame. We exploited a field that is currently not used on the TX side to signal the AC to the kernel, where we put the packet in the corresponding queue.

Since we made major modifications to the driver, we first validated the changes with measurements of the inter-arrival time. At first, we observed the distribution depicted in the center of Figure 5 and realized that the driver sets TXOPs for certain ACs by default. That means that when a device wins contention, it uses the channel for the time period configured as TXOP and sends packets spaced by SIFS during that time. Only after a TXOP, the device will trigger the post-TX backoff, indicated by the small bars at the slot boundaries in the plot. The ratio between packets sent after SIFS and sent after a post-TX backoff is controlled by the duration of the TXOP limit.

Following these tests, 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 5. Also for the other ACs we observed the correct distribution. With these tests we know that the AC categories are working and that the AIFS, SIFS, slot time, and CW_{min} are configured correctly.

With the validated Unex devices, we are able to conduct fairness measurements as a final step towards ensuring the correctness of the implemented algorithm. We use one PC with a Unex card as monitoring device that logs all frames and (in the first setup) saturate the channel with an SDR and a Unex and (in the second setup) with two Unex devices. The average throughput over time of both configurations is shown in Figure 6: we observe perfect fairness in both cases.

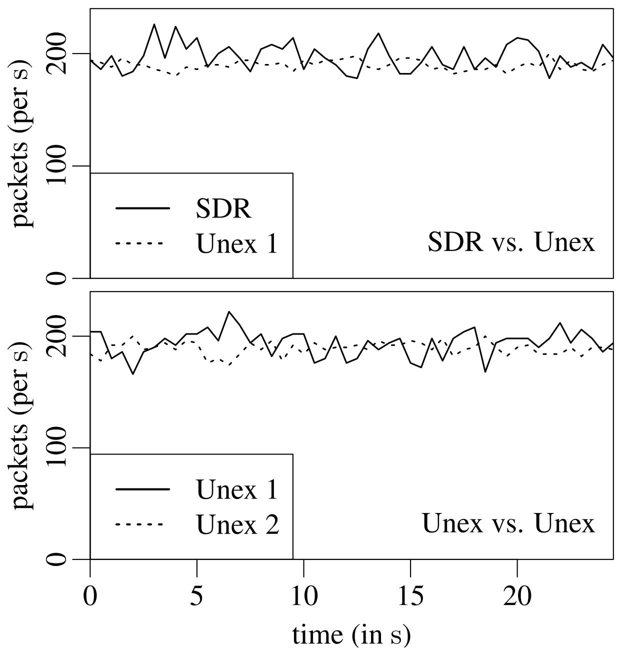

Figure 6: Throughput / fairness when saturating the channel with two devices (Unex and SDR).

VI. Conclusion

We presented a method that allows for standard compliant channel access for broadcast transmissions for a software-based SDR architecture. Our implementation follows the split functionality approach, where only time critical functionality such as carrier sensing and CSMA logic is implemented in hardware. All remaining physical layer processing remains implemented as software on a host PC.

This architecture preserves the ease and flexibility of a software implementation but at the same time allows for standard compliant channel access for broadcast transmissions. To demonstrate this, we studied the broadcast case in great detail and presented an Open Source implementation, which has been validated with an extensive set of measurements. This highlights the feasibility of the approach and shows that our implementation is able to meet all timing requirements of the standard. Furthermore, we showed its interoperability with IEEE 802.11p prototypes, as its transmissions occupy exactly an equal fair share of the wireless channel.

We believe that this work is particularly helpful in the vehicular context where broadcast is the primary communication paradigm. The availability of our implementation extends interoperability and standard conformance of SDRs from physical layer up to MAC and application layer.

References

[1] MAX2828/MAX2829 Single-/Dual-Band 802.11a/b/g World-Band Transceiver ICs. Datasheet Rev 0, Maxim Integrated, October 2004.

[2] Wireless Access in Vehicular Environments. Draft Standard P802.11p/D10.0, IEEE, January 2010.

[3] IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Multi-channel Operation. Std 1609.4, IEEE, February 2011.

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

[5] Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band. EN 302 663 V1.2.1, ETSI, July 2013.

[6] 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)*, pages 57–63, Maui, HI, September 2014. ACM.

[7] 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)*, pages 9–16, Hong Kong, China, August 2013. ACM.

[8] B. Bloessl, M. Segata, C. Sommer, and F. Dressler. Towards an Open Source IEEE 802.11p Stack: A Full SDR-based Transceiver in GNURadio. In *5th IEEE Vehicular Networking Conference (VNC 2013)*, pages 143–149, Boston, MA, December 2013. IEEE.

[9] E. Blossom. GNU Radio: Tools for Exploring the Radio Frequency Spectrum. *Linux Journal*, (122), June 2004.

[10] 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*, 2014. to appear.

[11] F. Dressler, H. Hartenstein, O. Altintas, and O. K. Tonguz. Inter-Vehicle Communication - Quo Vadis. *IEEE Communications Magazine*, 52(6):170–177, June 2014.

[12] 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*, pages 1–5, Venice, Italy, October 2010. IEEE.

[13] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance Anomaly of 802.11b. In *22nd IEEE Conference on Computer Communications (INFOCOM 2003)*, volume 2, pages 836–843, San Francisco, CA, March 2003. IEEE.

[14] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil. Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions. *IEEE Communications Surveys and Tutorials*, 13(4):584–616, November 2011.

[15] A. Khattab, J. Camp, C. Hunter, P. Murphy, A. Sabharwal, and E. W. Knightly. WARP: A Flexible Platform for Clean-Slate Wireless Medium Access Protocol Design. *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R)*, 12(1):56–58, January 2008.

[16] J. Mitola. The Software Radio Architecture. *IEEE Communications Magazine*, 33(5):26–38, May 1995.

[17] 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)*, pages 91–105, Boston, MA, April 2009. USENIX.

[18] T. Schmid, O. Sekkat, and M. B. Srivastava. An Experimental Study of Network Performance Impact of Increased Latency in Software Defined Radios. In *2nd ACM International Workshop on Wireless Network Testbeds, Experimental evaluation and Characterization (WiNTECH'07)*, pages 59–66, Montréal, Québec, Canada, September 2007. ACM.

[19] K. Tan, J. Zhang, J. Fang, H. Liu, Y. Ye, S. Wang, Y. Zhang, H. Wu, W. Wang, and G. M. Voelker. Sora: High Performance Software Radio Using General Purpose Multi-core Processors. *Communications of the ACM*, 54(1):99–107, January 2011.

[20] VCO. VITA Radio Transport (VRT). Std 49.0, VITA, 2009.