Enhancing the Performance of Random Access MAC Protocols for Low-cost SDRs

André Puschmann
Technische Universität Ilmenau
Ilmenau, Germany
andre.puschmann@tu-ilmenau.de

Paolo Di Francesco
CTVR / The Telecommunications Research Centre
Trinity College, Dublin, Ireland
pdifranc@tcd.ie

Mohamed A. Kalil
Technische Universität Ilmenau
Ilmenau, Germany
mohamed.abdrabou@tu-ilmenau.de

Luiz A. DaSilva
CTVR / The Telecommunications Research Centre
Trinity College, Dublin, Ireland
dasilval@tcd.ie

Andreas Mitschele-Thiel
Technische Universität Ilmenau
Ilmenau, Germany
mitsch@tu-ilmenau.de

ABSTRACT

Software Defined Radio (SDR) is a technology which facilitates experimentation and the practical realization of novel wireless communication systems. Especially low-cost SDRs, however, experience high communication delays due to the connection between the radio hardware and the host computer. This delay hinders the implementation of Medium Access Control (MAC) protocols. In Carrier Sense Multiple Access (CSMA) based protocols, especially the Clear Channel Assessment (CCA) as well as the subsequent channel access phase are subject to strict temporal constraints. In this paper, we present two strategies that address both issues and aim to enhance the performance and efficiency of CSMA protocols implemented on low-cost SDRs. The first approach employs a dedicated spectrum sensing engine as a CCA agent for the SDR. The second strategy optimizes the frame transmission path inside the SDR. Experimental results indicate that both strategies have a positive impact on reducing the slot time parameter of the CSMA MAC.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Design, Experimentation, Measurement, Performance, Verification

WiNTECH'13, September 30, 2013, Miami, Florida, USA.

ACM A978-1-4503-2364-2/13/09 ...$15.00.
http://dx.doi.org/10.1145/2505469.2505481

Keywords

Software defined radio; MAC; CSMA; USRP; Iris; Performance analysis; Latency

1. INTRODUCTION

Software Defined Radio (SDR) is a key technology for future multi-mode terminals for digital broadcasting and mobile communications. SDRs are also a building block for frequency-agile and cognitive radios (CR). Experimental research on SDRs and CRs has relied on a number of hardware and software platforms. Prominent examples are the *Universal Software Radio Peripheral* (USRP) and the *Wireless Open-Access Research Platform* (WARP) as well as *GNU Radio* and *Iris* on the software side. To date, these experiments have primarily focused on environments where a single link is established or, at most, two competing links coexist in an interference channel. The vast majority of experimental work on SDR platforms has dealt with the physical layer, while higher layer design impose significant research and engineering challenges due to increasing processing delay [1, 2].

The core of the problems associated with inexpensive SDRs are mainly twofold: (1) the limitation of general purpose computers in terms of signal processing performance and (2) the connection of the same to the RF part of the radio such as the USRP. In practice, this hinders the realization of *Carrier Sense Multiple Access* (CSMA) based protocols with IEEE 802.11-like [3] functionality, as the latencies lead to outdated sensing information and delayed frame transmissions. Moreover, the execution and programming model of the underlying SDR framework often also decrease the performance and efficiency of available MAC implementations.

This fact has motivated researchers to shift the entire MAC layer implementation to reconfigurable hardware such as an FPGA [4]. However, this approach clearly limits the flexibility and convenience of developing and experimenting with new protocols. Another approach, called *split-functionality*, has been developed by *Nychis et al.* to move only the time-critical aspects, such as the *Clear Channel Assessment*

(CCA) and the backoff mechanism, closer to the RF hardware [2]. A similar architectural design has been taken in [5] and the same concept has been generalized and then proved in an embedded platform. While the approach proposed in [2] and [5] provides a good trade-off between flexibility and achievable performance, the realization and any further adaptation of the time-critical functions still requires a high amount of low-level modification. In addition, this partition of MAC functionality between hardware and software makes the system architecture more complex and must be carefully planned. Moreover, other performance-critical aspects such as the generation of frames (i.e. payload framing and modulation) is still carried on the host computer. We therefore argue that the transmit path on the host side also needs to be modified, even – or, especially – if the CCA and transmission logic is hardware-assisted in order to gain the full benefits from it.

The contribution of this paper can be summarized as follows. First, we employ a dedicated spectrum sensing engine as a CCA agent for the SDR that is fully controllable in software. The IMEC sensing engine [6] is used for this purpose. It provides a low-cost, low-power, high-performance solution for spectrum sensing. This allows us to study a hardware-assisted CCA mechanism independently from the transmission logic. In this work, the latter remains on the host computer, which differentiates our analysis from [2] and [7].

Second, we propose a strategy that optimizes the frame transmission path inside the SDR framework. We call this technique *premodulation*. The main idea of this concept is to move the frame to be sent as close as possible to the RF part of the radio while at the same time allowing software to precisely control the actual transmission.

We demonstrate and evaluate the two aforementioned techniques. We describe the details of the implementation, which is based on a CSMA MAC protocol developed earlier by us [8]. Experimental results indicate that both strategies have a positive impact in reducing the slot time parameter of the CSMA MAC and mitigate the negative effects of limited signal processing capabilities.

The reminder of this paper is organized as follows: In section 2 we introduce fundamentals of the MAC and the sensing engine. In section 3 we describe the first approach, the hardware-assisted CCA agent, followed by the transmit path optimization in section 4. A detailed evaluation of both solutions is presented in Section 5.

2. BASICS

We start this section with a description of a CSMA MAC protocol. Then, we present the Iris SDR framework and the IMEC sensing engine. Both are part of the FP7 project *Cognitive Radio Experimentation World* (CREW) [9].

2.1 CSMA MAC protocol

This section briefly summarizes the basic functionality of CSMA MAC protocols. With millions of devices around the world, probably the most popular representative of such a protocol is the IEEE 802.11 standard [3]. To reduce the probability of a frame collision during communication, nodes that have data frames to send first have to contend for the channel. This is usually achieved through a mechanism called CCA which needs to be executed prior to sending a frame. Only if the medium is found to be idle for a certain amount of time, a so called *Distributed Inter Frame Space* (DIFS), the node starts transmitting the frame.

Table 1: Protocol parameter according to the IEEE 802.11b PHY specification [3].

Slot time	SIFS	DIFS
20 µs	10 µs	50 µs

Fundamental protocol parameters that impact the transmission time of a frame - and thus the achievable performance - are *slot time* and *Short Inter Frame Space* (SIFS). Slot time is a hardware-dependent parameter that includes all physical and MAC layer delays, including the CCA time, the transceiver switch-over time and the MAC processing time. Independently from the physical layer configuration, DIFS is calculated by the following equation: $DIFS = 2 \times Slot\ time + SIFS$. Table 1 summarizes the most important parameter defined in the standard.

As shown in [10], the total transmission time of a data frame comprises the transmission time of the data frame itself, $t_{tr} = L/R$, where L is the length of the frame in bits and R is the data rate, a constant overhead, plus the time spent during the contention phase if multiple nodes (N) are active in the network:

$$T = t_{tr} + t_{ov} + t_{cont}(N) \tag{1}$$

The constant overhead is defined as:

$$t_{ov} = DIFS + t_{pr} + SIFS + t_{pr} + t_{ack} \tag{2}$$

and comprises twice the preamble and header transmission time t_{pr}, i.e. for data and acknowledgement frames, the SIFS period, and the transmission time of the acknowledgement t_{ack}. The efficiency of the protocol can be written as the transmission time of the data frame divided by the total transmission time:

$$\eta = \frac{t_{tr}}{T} \tag{3}$$

To illustrate the impact of the slot time parameter, we have plotted the protocol efficiency of the IEEE 802.11 protocol operating at a rate of 2 Mbit/s under various slot time configurations while keeping the SIFS parameter constant. As it can be seen from Figure 1, reducing the slot time has an important impact on the efficiency and thus on the achievable throughput of a communication system.

2.2 Iris

Iris [11, 12] is an open-source software framework for designing reconfigurable, component-based SDRs. XML files are used to define the components a radio consists of and how they are connected with each other. Upon start, the Iris core application loads the configuration file and constructs the radio flow graph by connecting input and output ports as specified by the user. The architecture defines multiple building blocks which may be used to describe an entire radio. The core blocks of the Iris architecture can be briefly described as follows:

- *Components:* Components are self-contained entities implementing a discrete radio function such as a filter,

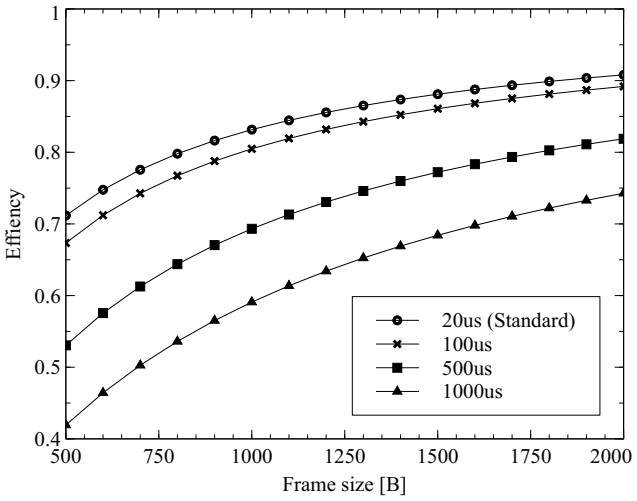


Figure 1: Efficiency of the IEEE 802.11 protocol under various slot time configurations at a data rate of 2 Mbit/s.

modulator or even an entire MAC protocol. They are characterized by a set of input and/or output ports which are used to connect the components with one another.

- *Engines*: Engines are an abstract concept to encapsulate one or more components within a radio. An engine defines how the specific part of the flow-graph under its regime is executed, how data is passed between components and how the reconfiguration is organized. A radio may consist of one or more engines.

- *Controllers:* During runtime, controllers are the managing entities of a radio. They are capable of reconfiguring all component parameters, as well as changing the structure of the entire radio by inserting or deleting components and links. Controller activity is usually triggered by events sent out by components. A controller, however, may also trigger a command which can be received by another component, thus allowing light weight inter-component communication.

2.3 Imec sensing engine

The *IMEC Sensing Engine* (SE) [6] is a versatile sensing solution, consisting of a reconfigurable analog frontend (the WARP analog frontend in this configuration) and a digital frontend capable of performing various sensing operations. The reconfigurability of both components allows for both wide band spectrum sensing and narrow band coarse energy detection. The SE can be connected to a PC via an USB interface which is used to configure the device (i.e. downloading firmware, setting center frequency and bandwidth). To allow for low-latency communication between SE and host PC, the former can be connected to the parallel port through *General Purpose Input Output* (GPIO) pins.

We have programmed the SE such that for every 32 time domain samples, the sensing processor of the digital frontend calculates the signal power present in the selected frequency band. The energy level is compared against a programmable threshold and the result is written to one of the GPIO pins. Since these calculations are very straightforward and performed completely in hardware, the SE allows for fast sensing operations with low latency.

A photograph of the SE with a WARP frontend mounted on top of it can be seen in Figure 2.

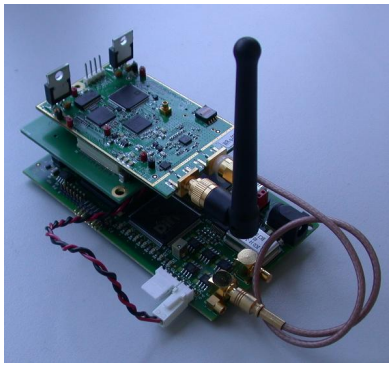


Figure 2: Photograph of the IMEC Sensing Engine with WARP analog frontend [9].

3. HARDWARE-ASSISTED CCA AGENT

As outlined above, the main reasons that hinder the practical realisation of CSMA MAC protocols on low-cost SDR are significant processing delays that lead to outdated sensing information and delayed frame transmissions.

To mitigate the negative effects of an energy detection algorithm implemented in software on the host computer, we explored the impact of a dedicated sensing device, i.e. the IMEC SE, that executes the CCA mechanism in hardware. The SE is an RF receiver that complements the actual RF transceiver of the SDR, i.e. the USRP2 in our case. In contrast to the USRP2, which uses an Ethernet interface, a GPIO pin of the SE is directly connected to one of the data pins of the parallel port of the host computer. For this purpose, we have equipped our SDR hosts with PCI express parallel port adapters.

To interface the SE with the CSMA MAC, we have developed an Iris component that controls the operation of the external hardware device. The component configures the SE during the initialization phase of the radio flow graph and reacts upon sensing requests issued from the CSMA core component during runtime. The interaction between MAC core and sensing component is explained in detail in [8].

From a SDR performance analysis point of view, the main advantage of having an external, hardware-assisted CCA mechanism is that this solution allows us to study its impact and relationship to the transmission delay independently.

4. TRANSMIT PATH OPTIMIZATION

This section describes an optimized algorithm for efficient frame transmission in software radios. We first describe how frames are usually processed to motivate our optimization strategy. We then describe the proposed idea and discuss some implementation details.

4.1 Frame Processing

The exchange of data frames in CSMA-based protocols consists of a number steps. In most implementations they

are carried out sequentially. If a node has a frame to send it first needs to determine the current state of the channel. It therefore needs to receive and transfer the incoming samples from the RF hardware to the host in order to execute the CCA algorithm. If the channel is found to be idle, the outgoing data can be modulated and sent to the RF hardware, which will finally transmit the frame. Figure 4(a) shows a simplified Tx flow chart (without backoff functionality) which is used in current CSMA implementations.

However, strictly following this sequence causes a severe performance drop due to the low signal processing capacity of conventional PC hardware. This is because the frame needs to be modulated before it is actually sent out. This takes a relatively large amount of time and increases the latency between the time when MAC observes a transmission opportunity and the actual time at which it accesses the medium.

Figure 3 shows a continuous uni-directional frame transmission between two SDR nodes that has been captured with a real-time spectrum analyser. In the experiment, the transmitter was sending a (relatively) large data frame to the receiver and was waiting until it received a (relatively) small acknowledgment frame before sending the next frame. A large gap of around 10 ms between acknowledgement and the following data frame can be clearly observed. This is mainly due to the sequential execution that leads to a high frame generation delay. The same experiment has been repeated after the optimization. The results will be described in Section 5.3.

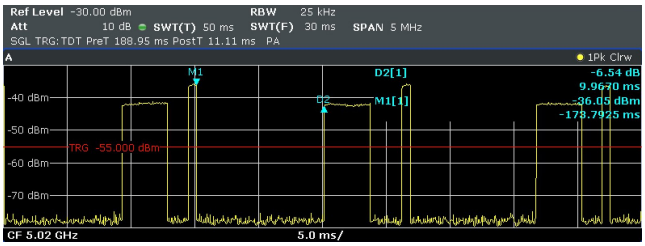


Figure 3: Snapshot of a continuous, uni-directional frame transmission (data/acknowledgement pairs) between two legacy SDRs captured using the Rhode & Schwarz FSVR spectrum-analyser.

4.2 Optimization Strategy

To overcome the problem described above, we propose to modify the transmit flow graph such that a frame is modulated (and thus ready to be sent) *before* the CCA algorithm is executed. We call this optimization strategy *premodulation*. Effectively, this reduces the time between when the channel has been detected to be free and the actual channel access time. Moreover, we propose to use the time the sender is waiting for an acknowledgement to prefetch and modulate the following data frame in the background.

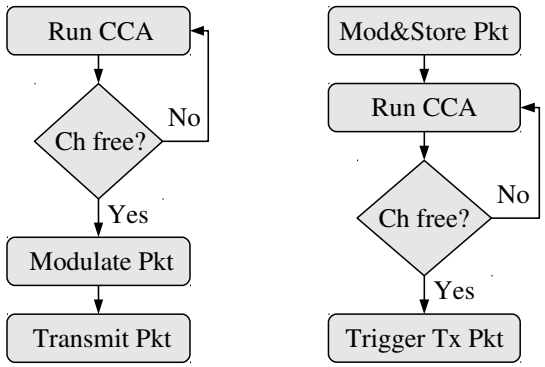
In order to implement the proposed idea in practice, however, the traditionally layered execution model between MAC and physical layer needs to be revised. In other words, the generation and the actual transmission of data frames need to be separated from each other. From the MAC protocol point of view, this requires framing a data frame and sending it down to the physical layer for modulation without transmitting immediately. Consequently, the physical layer would need a buffer to store those frames that are ready to transmit but were not yet approved by the MAC. After finishing the CCA procedure, the MAC would simply trigger the transmission of the frame buffered inside the physical layer.

The main idea of this concept is to move the frame to be sent (represented as a stream of I/Q samples) as close as possible to the RF part of the radio while at the same time allowing software to precisely control the actual transmission. This is similar to the idea of moving the full transmission mechanism including CCA to the FPGA [2], with the main difference that our solution doesn't require any hardware modifications. A similar approach is also employed in commercial wireless network cards, again with the same limitations.

Figure 4(b) shows the decoupling of frame generation and transmission as compared to the legacy solution shown in Figure 4(a) where both steps are sequentially executed.

The advantages of the premodulation mechanism can be summarized as follows:

- *Enabling of high-performance CSMA radios:* The parallel generation and transmission of data frames better suits today's multi-core, multi-threaded architectures used in many SDR setups. It is essential to fully exploit the potential of a hardware-assisted transmission logic because the data flow is no longer slowed down due to the slow generation of frames on the host computer.
- *Efficient processing of transmission errors:* As outgoing frames are buffered inside the physical layer, retransmission can be processed more efficiently. Outgoing frames do not need to be modulated twice[1].
- *Independence of CCA realization:* The premodulation mechanism is independent from how CCA is implemented in the SDR. It can be used with software CCA or hardware CCA (either with an external sensing engine or with a CCA algorithm implemented inside the FPGA of the USRP).



(a) Legacy system without premodulation.

(b) Enhanced system with premodulation.

Figure 4: Comparison of the CCA flow-graph with and without pre-modulation.

[1]Assuming that initial frame and retransmitted frame are identical.

4.3 Implementation Details

As described above, the proposed premodulation technique requires us to separate the generation of frames and their actual transmission. In our radio architecture, such a separation is implemented inside the CSMA MAC protocol core component as well as two additional components: (1) a transmit buffer component called *TxBuffer* that stores the outgoing frame and (2) a software controller called *MacPhyController* that coordinates the interaction between the MAC core and the transmit buffer.

If the MAC has a frame to send, it frames the data frame and sends it down to the physical layer for modulation. Before passing it down to the physical layer, however, the MAC component creates a unique identification key for the frame based on its destination address and sequence number. This key is appended to the frame as metadata that allows to clearly identify the frame even after it has been modulated.

On the way down the transmit flowgraph, the frame traverses the modulator before it eventually reaches the buffer component. If the buffer detects metadata associated with the frame, it will not send the frame immediately but will store it internally inside a data structure that uses the metadata as a key to access the data. Using the exact same key, the MAC can instruct the buffer to transmit a certain frame by using a *transmit event* that is sent to the buffer component via the MacPhyController. As described in section 2.2, controllers are software blocks that allow light weight communication among radio components.

Figure 5 shows the Iris flow graph that includes the components involved in the premodulation mechanism.

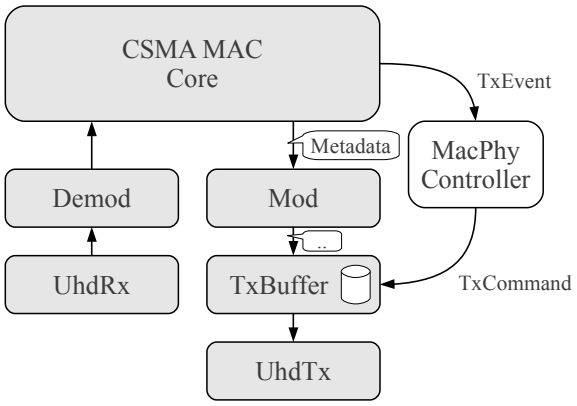


Figure 5: Iris flow graph of CSMA radio with frame premodulation.

4.4 Discussion

The premodulation technique proposed in this paper tries to exploit idle periods (i.e. while the node is waiting for a busy medium to become free or an acknowledgement to arrive) to execute the computationally costly step of modulating an outgoing frame. This is an advantage but may be a disadvantage at the same time. This is because the effectiveness depends on the length of the idle period and the time needed to premodulate the frame. Clearly, if the idle period is shorter than the modulation time, there will still be an unavoidable delay. Therefore, premodulation may be more effective in moderate or high traffic conditions where nodes, on average, have to wait longer before they actually can access the medium. On the other hand, recent advances in multi-core, multi-threaded computer architectures have given rise to justifiable hope of improvements in digital signal processing performance and, therefore, lower modulation time. However, how an optimal ration between idle period length and modulation time may be achieved in a given system design is still an open research question.

The second disadvantage of the premodulation technique is that it doesn't support dependent frames. Dependent frames are frames that are sent in response to received frames and that require certain information from those frames such as a source address or a sequence number. Popular examples for this kind of frames are acknowledgements or *Clear to Send* frames. A possible solution to support acknowledgements could be to defer their transmission on purpose. A protocol could simply acknowledge the second last frame instead of the last. Another possibility is the implementation of a windowing function or block acknowledgement scheme similar to IEEE 802.11e.

5. EVALUATION

In this section, we will evaluate both optimization strategies presented in this paper. First, we describe our measurement setup. As a performance metric, we assess the slot time parameter of the CSMA-based MAC protocol as well as the frame generation delay as a function of the payload length. We compare our measurements with a reference system that runs a CSMA MAC protocol purely implemented in software.

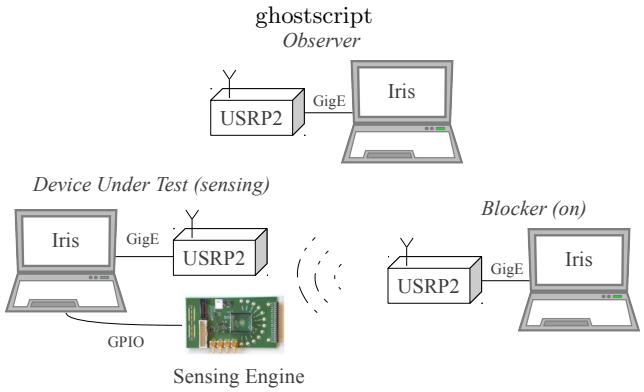
5.1 Methodology and Metrics

Slot time is an important PHY layer parameter which comprises all physical and MAC layer processing delays including the CCA mechanism and air propagation time. We refer the reader to [3] for a detailed explanation. In order to empirically determine the slot time of a given SDR, we adopt an experiment previously described in [8].

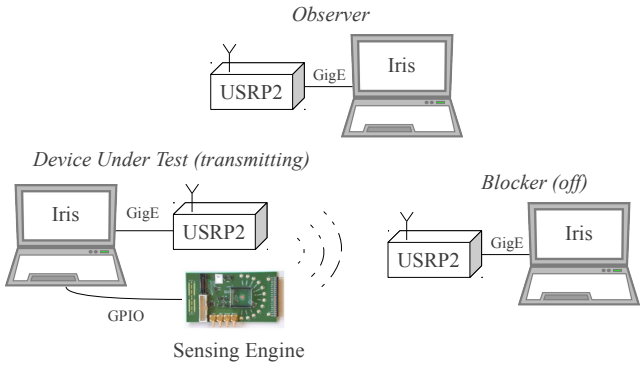
Slot time may be approximated by measuring the delay between the instant at which the channel state turns from busy to idle and the time at which the SDR node actually accesses the channel. To measure that instant, we have used the setup shown in Figure 6, which comprises three SDR nodes. For simulating a busy channel, we used one SDR node (i.e. the blocker) which continuously transmitted a pseudo-random OFDM signal, see Figure 6(a). The device under test (DUT) was configured such that it sent a single frame with a predefined size of 1000 B after the blocker had been turned off, see Figure 6(b). A third monitoring node captured the channel activity during the experiment. The recorded samples were processed offline through a Python script which plotted the received energy level over the experiment execution time. For each configuration, the slot time of the system has been calculated using the following formula: $T_{Slottime} = T_{DUT_{ON}} - T_{BLOCKER_{OFF}}$.

The same setup has also been used to determine the frame generation delay. The plots in Figures 3 and 7 have been created using a *Rohde & Schwarz* real-time spectrum analyser.

All three SDR nodes were identically configured systems with a conventional Ubuntu Linux. They were also equipped with one USRP2 each for transmitting and receiving data. For the experiments with the dedicated sensing engine, two SDR nodes have been equipped with a IMEC SE that we connected via the USB bus and parallel port. The complete



(a) The blocker is occupying the channel.



(b) Device under test is transmitting.

Figure 6: Experimental setup for the empirical determination of the slot time.

radio configurations as well as the parameter settings of the experiment are summarized in table 2.

5.2 Slot time

The results of the slot time measurement are listed in Table 3. They have been obtained during ten runs for each configuration. We compare them against the reference configuration which runs a CSMA protocol and CCA mechanism purely implemented in software without any transmit path optimization (first row in Table 3).

As it can be seen from the results, each of the proposed optimization strategies significantly improves the slot time characteristic of the SDR. Compared to the reference system, the premodulation technique reduced the slot time by a factor of three, down to roughly 1 ms, which is a remarkable value for a purely software-based CSMA solution.

Compared to the reference system, employing a dedicated sensing engine as a CCA agent improved the slot time characteristic of the protocol by a factor of two. By far the biggest improvement, however, has been achieved with the sensing engine as CCA agent and enabled premodulation. Compared to the same system without premodulation, the slot time could be reduced by a factor of roughly ten. Compared to the reference system the slot time could be reduced by a factor of roughly 18. Interestingly, the impact of premodulation (i.e. the improvement factor) is higher when using the hardware CCA compared to the CCA implemented in software.

Table 2: Hardware configuration and system parameters.

Parameter	Value
RF hardware	USRP2 , XCVR2450 and Imec sensing engine
Sampling rate	2 Msamples/s
f_{center}	2.404 GHz
Channel bandwidth	2 MHz
Modulation scheme	QPSK
Subcarriers (data)	64 (44)
Transport protocol	UDP
Frame size	$100 - 6000$ B
CW_{min}	31
CW_{max}	1023
ACK timeout	5 ms

Furthermore, we have compared our measurements to a CSMA implementation that realizes the entire transmission logic in hardware, i.e. inside the FPGA of an USRP E100 [5] [7]. In contrast to hardware-assisted CCA only, this approach has the great advantage that the I/Q samples are already available at the transmitter after the channel has been sensed idle. This configuration therefore has the lowest slot time. However, on the E100 we have still observed a relatively large spread of measurements.

Due to the limited processing capabilities, sample rates above 2 Msamples/s are impractical on the USRP E100. We have therefore ported the CCA code implemented on the E100 to the USRP N210 in order to be able to increase the sample rate. We could now observe a much more stable slot time than on the E100. We have used a slightly different energy detection implementation on the N210 which is a possible explanation for the variations on the E100.

The deterministic behaviour of the FPGA implementation allows to calculate the expected slot time value as a function of the sample rate and the width of the sample window used to determine the channel state plus a hardware dependent overhead to switch between receive and transmit mode:

$$Slot\,time = t_{Sample} \times NumSamples + t_{RxTxTurnaround} \quad (4)$$

Given a sample window width of 64 samples, a sample rate of 5 Msamples/s and a Rx/Tx turnaround time of 1 µs of the *MAX2829* [13] transceiver chip of the XCVR2450 board, we were able to reduce the slot time down to 14 µs.

This is even slightly below the reference value of 20 µs, defined in the IEEE 802.11b standard. However, we have not included these measurement results into table 3 to avoid an unfair comparison with different sampling rates.

5.3 Frame generation delay

To compare our optimized frame processing strategy to a legacy SDR, we have repeated the experiment described in section 4.1 and have again configured the radios such that a saturated transmitter was sending a continuous data stream to the receiver.

Figure 7 shows the radio environment in this scenario as captured with the a real-time spectrum analyser. It can be clearly observed that the gap between an acknowledgement and the following data frame has decreased significantly to

Table 3: Comparison: slot time measurements for different CSMA configurations at a sample rate of 2 Msamples/s (in ms).

Configuration	Min	Avg	Max	Stdev
SW CCA (reference)	3.2	3.4	3.6	0.15
SW CCA w/ premod	0.87	1.09	1.21	0.13
HW CCA	1.8	2.0	2.2	0.15
HW CCA w/ premod	0.14	0.19	0.27	0.04
Full HW on E100	0.059	0.088	0.156	0.04
Full HW on N210	0.039	0.0397	0.041	0.0006

around 1.5 ms, which is almost a factor of 7 lower as compared to the legacy system.

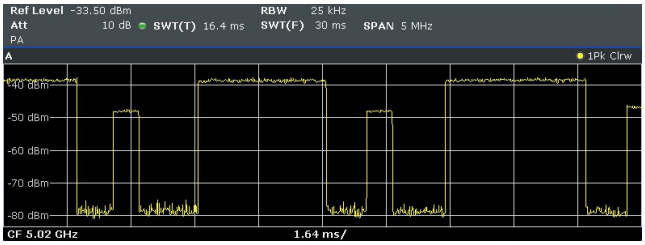


Figure 7: Snapshot of a continuous, uni-directional frame transmission (data/acknowledgement pairs) between two SDRs with frame premodulation captured using the Rhode & Schwarz FSVR spectrum-analyser.

We have further measured the actual data throughput for various payload lengths. For the legacy SDR, we have found that increasing the size of a data frame has not improved the throughput as would have been expected. The reason for that is a dependency between frame size and the size of the gap between consecutive frames. This is because the larger the frame is, the longer the modulation process takes. With the premodulation scheme turned on, we were able to observe improved efficiency and throughput when the payload length is increased. We have measured the frame generation delay with and without premodulation and have plotted the results in Figure 8. As it can be seen from the results, while the frame generation delay increases linearly without premodulation, it remains constant when premodulation is turned on.

6. CONCLUSION

Low-cost SDRs, first and foremost the popular USRP series devices, experience high communication delays due to the connection between the radio hardware and the host computer. This delay hinders the implementation of random access MAC protocols. In this paper, we have presented two strategies that mitigate the negative effects. The first approach employs a dedicated spectrum sensing engine as a hardware-assisted CCA agent and therefore reduces the time a SDR requires to determine the current state of the radio channel.

Moreover, an extensive analysis of current SDR frameworks and CSMA MAC implementations has led to the observation that the programming and data flow mode is one

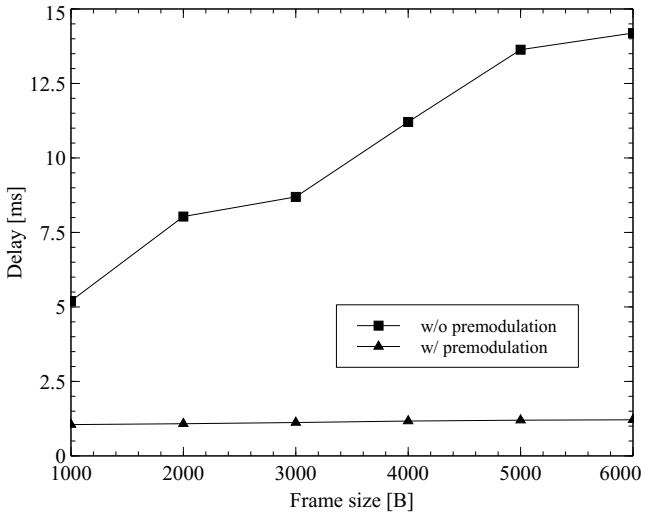


Figure 8: Impact of premodulation scheme on frame generation delay.

of the reasons why today's SDRs show relatively poor performance. This has motivated us to design and implement an improved and more efficient frame transmission scheme. The main idea behind this technique is to parallelize the transmit flow graph and to exploit idle periods at the sender during an ongoing transmission.

Such strategies are not only important for protocols which are entirely implemented in software, but also allow to fully exploit the potential of system designs which realize parts of the MAC in hardware.

We are currently investigating possibilities to reduce the SIFS period which is still a dominating factor in current SDR realizations. Furthermore, we are planning to extend our performance analysis which is currently limited to continuous uni-directional traffic and to include more nodes in a larger network deployment.

Acknowledgment

This study was partially supported by the International Graduate School on Mobile Communications (Mobicom) which is supported by the German Research Foundation (GRK1487), by the EU FP7 under Grant Agreement No 258301 (CREW Project) and by the European Cooperation in Science and Technology under project COST Action IC0902. We would also like to thank Mattias Desmet and Séamas McGettrick for their positive collaboration. Furthermore, we would like to thank our anonymous reviewers for their valuable comments.

7. REFERENCES

[1] Thomas Schmid, Oussama Sekkat, and Mani B Srivastava. An experimental study of network performance impact of increased latency in software defined radios. In *Proceedings of the the second ACM International Workshop on Wireless Network Testbeds Experimental Evaluation and Characterization (WinTECH)*, 2007.

[2] George Nychis, Thibaud Hottelier, Zhuochen Yang, Srinivasan Seshan, and Peter Steenkiste. Enabling

MAC Protocol Implementations on Software-defined Radios. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, April 2009.

[3] Institute of Electrical and Electronics Engineers. *IEEE 802.11 Standard, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, 2007.

[4] J. Ansari, Xi Zhang, A. Achtzehn, M. Petrova, and P. Mähönen. A Flexible MAC Development Framework for Cognitive Radio Systems. In *IEEE Wireless Communications and Networking Conference (WCNC)*, March 2011.

[5] Paolo Di Francesco, Séamas McGettrick, U. K. Anyanwu, J. C. O'Sullivan, A. B. MacKenzie, and L. A. DaSilva. A Split Architecture for Random Access MAC for SDR Platforms. In *8th International Conference on Cognitive Radio Oriented Wireless Networks (CROWNCOM)*, Washington D.C., USA, July 2013.

[6] Sofie Pollin, Lieven Hollevoet, Peter Van Wesemael, Mattias Desmet, Andre Bourdoux, Eduardo Lopez, Frederik Naessens, Praveen Raghavan, Veerle Derudder, Steven Dupont, and Antoine Dejonghe. An integrated reconfigurable engine for multi-purpose sensing up to 6 GHz. In *IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 656–657, May 2011.

[7] Uchenna Kevin Anyanwu. A Reconfigurable Random Access MAC Implementation for Software Defined Radio Platforms. Master's thesis, Virginia Polytechnic Institute and State University, 2012.

[8] André Puschmann, Mohamed A. Kalil, and Andreas Mitschele-Thiel. A Flexible CSMA based MAC Protocol for Software Defined Radios. *Frequenz Journal of RF-Engineering and Telecommunications*, 6:261–268, October 2012.

[9] FP7-CREW Cognitive Radio Experimentation World. http://www.crew-project.eu/.

[10] Martin Heusse, Franck Rousseau, Gilles Berger-Sabbatel, and Andrzej Duda. Performance Anomaly of 802.11b. In *22nd Annual Joint Conference of the IEEE Computer and Communications IEEE Societies (INFOCOM)*, March 2003.

[11] Paul D. Sutton, Jörg Lotze, Hicham Lahlou, Suhaib A. Fahmy, Keith E. Nolan, Baris Özgül, Thomas W. Rondeau, Juanjo Noguera, and Linda E. Doyle. Iris: An Architecture for Cognitive Radio Networking Testbeds. In *IEEE Communications Magazine*, volume 48, pages 114–122, September 2010.

[12] Software Radio Systems Ltd. The Iris project page. http://www.softwareradiosystems.com/redmine/projects/iris.

[13] Maxim Integrated. *MAX2828/MAX2829 Single-/Dual-Band 802.11a/b/g World-Band Transceiver ICs datasheet*, 2004.