An IEEE 802.11 MAC Software Defined Radio Implementation for Experimental Wireless Communications and Networking Research

Juan R. Gutierrez-Agullo, Baldomero Coll-Perales and Javier Gozalvez
Uwicore, Ubiquitous Wireless Communications Research Laboratory, http://www.uwicore.umh.es
University Miguel Hernández of Elche, Avda. de la Universidad, s/n, 03202, Elche, Spain, j.gozalvez@umh.es

Abstract— **Wireless ad-hoc networking will be a fundamental component of emerging technologies such as multi-hop cellular networks, vehicular ad-hoc networks and wireless mesh networks, with most of these technologies based on evolved versions of the 802.11 standard. To achieve their expected benefits, and address their challenges, in terms of energy efficiency, reliability or reconfigurability, advanced cooperative and cross-layer communications and networking techniques need to be investigated. Most of the studies conducted to date are based on analytical and simulation techniques, and the use of commercial hardware equipment usually lacks the flexibility needed to conduct advanced research studies. In this context, this paper presents a fully programmable Software Defined Radio implementation of the IEEE 802.11 MAC that can be used, configured and fully modified, to develop advanced cross-layer communications and networking techniques.**

Keywords-component: software defined radio; IEEE 802.11 Medium Access Control(MAC); hardware prototype

I. INTRODUCTION

Wireless ad-hoc networks [1] can enable the radio connectivity of multiple cooperative nodes. Nodes within the radio range of each other can communicate directly over wireless links, and those that are far apart can use intermediate/cooperative nodes as relays to establish a multi-hop wireless link. For this purpose, multiple networking functions are controlled and managed by the nodes themselves, including the network association process, the multi-hop route establishment, and the regulation to access the shared wireless channel, among others. The interest in wireless ad-hoc networks is continuously increasing, and its potential to improve the communications range and perceived Quality of Service (QoS) has resulted in that ad-hoc networking is a fundamental aspect of emerging technologies such as multi-hop cellular, vehicular ad-hoc and wireless mesh networks [1], among others. The acceptance and capabilities of the IEEE 802.11 standard [2] results in that many of these emerging technologies use evolved versions of 802.11 to support its ad-hoc communications.

The benefits of wireless ad-hoc networks, in terms of capacity improvement, coverage extension, lower infrastructure cost and power saving, have already been widely investigated based on both analytical and simulation studies. However, the ad-hoc performance advantages shown in these studies need to be validated through hardware testbed platforms that eliminate the generally made assumptions and simplifications regarding the real world operating environment. To this end, the research community is moving towards the implementation of wireless ad-hoc functionalities in testbeds using either commercial 802.11 cards provided with open source drivers, or Software Defined Radio (SDR) platforms.

Open source driver platforms use commercial 802.11 cards, and control their functionalities through modifiable drivers such as ath9k. In this open source driver architecture, the physical layer (PHY) is integrated into the chip of the card, and is therefore not accessible to users. The features specified in the MAC layer are logically partitioned in two different modules, based on the time-critical aspect of the tasks to be conducted. The lower MAC module, which operates on the wireless card and depends on the proprietary Hardware Abstraction Layer (HAL), controls the time critical functions. On the other hand, the upper MAC module, which operates on the driver, is responsible for more delay-tolerant control plane functions. Consequently, the ad-hoc protocol implementations and potential communications and networking proposals to be tested are restricted to the upper part of the MAC layer. Although open source driver platforms ensure the compatibility with commercial 802.11 equipments, the limited access to the MAC layer and the nonexistent control of the PHY layer prevent the development of advanced cross-layer communications and networking techniques.

A viable alternative to provide hardware platforms with full access and reconfigurability of communication protocols is SDR. SDR is a class of radio system which uses programmable digital devices to perform the signal processing necessary to transmit and receive baseband information at radio frequency. SDR systems substitute hardware components by software implementations on embedded computing devices and Personal Computers (PCs). The SDR architecture is totally accessible, which provides the sufficient flexibility to facilitate the design of novel communications and networking techniques. In this context, this paper presents a fully programmable SDR implementation of the IEEE 802.11 MAC that can be used, configured, and fully modified, to develop advanced cross-layer communications and networking techniques. The current work is developed in the context of a research project focused on the study of 802.11-based multi-hop cellular

networks based on mobile relays, where for example novel energy-efficient and cross-layer multi-hop routing [3] and mesh networking [4] techniques are being investigated. Once these protocols have been analyzed and optimized through computer simulations, the target is now to validate them through experimental SDR-based hardware platforms, which requires the SDR MAC implementation described in this paper. However, it is important to note that the proposed 802.11 MAC SDR implementation allows the further integration with additional 802.11 amendments such as 802.11 a/g/n, 802.11s for mesh functionalities and 802.11p for vehicular communications.

II. SOFTWARE DEFINED RADIO PLATFORMS

A. Wireless Open-Access Research Platform (WARP)

WARP is a scalable, extendable and programmable wireless community platform created at Rice University [5] for prototyping and researching next-generation wireless networks. WARP uses a Field Programmable Gate Array (FPGA) for digital signal processing, while its multiple daughtercard slots support a wide range of Input or Output (I/O) radio devices. WARP also provides programming tools, application libraries and open-access repositories to handle the hardware, and develop advanced wireless projects. A fast processing of the digital signals is achieved thanks to its embedded Central Processing Unit (CPU), although the evolution of the platform is restrained by the embedded nature of the CPU.

B. Universal Software Radio Peripheral (USRP)

USRP version 1 (USRP1) is a basic SDR platform, developed by Ettus Research, that implements the front-end functionality, and the Analog to Digital (A/D) and D/A conversion on a FPGA. However, the physical layer processing is done on a PC where the USRP1 is plugged. The USRP1 connects to the PC through an Universal Serial Bus version 2.0 (USB 2.0) which restricts the transfer speed to 60MB/sec or 15Msamples/sec, therefore producing a maximal spectral bandwidth of 7.5MHz. Since the 802.11a/b/g standards have wider bandwidth channels (20MHz), USRP1 was not a feasible SDR platform to develop 802.11-compliant ad-hoc testbeds. To overcome this limitation, the USRP2 was released and it replaced the USB interface with a Gigabit Ethernet one that supports simultaneous I/O signals of 50MHz radio frequency bandwidth. USRP2 also provides faster and more precise AD/DA converters, and a FPGA optimized for Digital Signal Processing (DSP) applications. It allows processing complex waveforms at higher sample rates, turning USRP2 into an appropriate platform for researching wireless ad-hoc functionalities. The improvements carried out in the USRP2 platform prevent the compatibility with the USRP1 developments, although an Universal Hardware Driver (UHD) is being developed to solve it.

C. GNU Radio toolkit

GNU Radio is a free collection of signal processing blocks used for building SDR platforms. In fact, GNU Radio is the primary software platform supporting the PC drivers for USRP1 and USRP2. The basic principle behind the use of GNU Radio is to build a Flow-Graph (FG) composed of various signal processing blocks that perform specific radio functionalities. The main limitation of GNU Radio is the flow of information through the software processing blocks. Currently, the FG's processing must be done using a continuous flow of samples from one block to the next one. If a block does not receive a data stream, the FG is stuck until a new stream is ready to be processed. This issue introduces a delay compared with hardware implementations where bits are independently processed [6].

D. Related implementations

There is a growing interest in the research community to design and develop SDR hardware platforms to investigate the potential, performance and operation of advanced ad-hoc communication systems. A practical demonstration of the potential of cooperative ad-hoc mechanisms is shown in [7], where the authors propose a cross-layer mechanism for 802.11 along with a MAC protocol that allows selecting neighboring helper stations for MAC layer forwarding. This work has been conducted under the CoopMAC (Cooperative MAC) project that it is aimed at building an open-source architecture for rapid prototyping of PHY and MAC layer by leveraging existing open-source radio platforms. The project was first based on the USRP1 board, although its USB limitation resulted in a final implementation on WARP. The PHY layer radio platform provides a wideband radio front-end covering the unlicensed frequency bands at 2.4GHz and 5GHz. However the lack of detailed MAC implementations leads to CoopMAC to build a simplified version of the MAC protocol (Carrier Sense Multiple Access, CSMA). Another interesting SDR initiative is Hydra [8], developed by the University of Texas at Austin and Drexel University. Hydra is a multi-hop wireless testbed used to investigate MIMO ad-hoc networking. Hydra nodes consist of a flexible USRP1 front-end and a PC that executes the software-defined MAC and PHY layers. Hydra is one of the major cooperative ad-hoc research initiatives, although its use of USRP1 results in the limitations previously described.

To overcome the USRP1 limitations, while maintaining the flexibility to design ad-hoc communications and networking techniques, BBN Technologies established the Adaptive Dynamic Radio Open-source Intelligent Team (ADROIT) project. The ADROIT project is building an open-source software-defined data radio intended to be controlled by cognitive applications. The ADROIT project implemented the PHY layer of the IEEE 802.11b standard[1] on USRP2, although a single node could only operate as either transmitter or receiver. The MAC layer implementation is yet limited since the only functions available are the formation and fragmentation of the MAC Protocol Data Unit (MPDU) frames, and carrier sensing mechanisms. Another SDR wireless platform implemented using the USRP2 platform is being developed by the FTW research centre to demonstrate that flexible platforms are also capable of generating truly standard-compliant OFDM frames. Indeed, FTW presents in [9] a fully functional IEEE 802.11a/g/p transmitter that represents a very valuable building block for upcoming SDR projects. However, to

[1] The BBN PHY layer available is only able to decode low rate 802.11b data packets at 1 and 2 Mbps.

achieve a fully interactive software-defined 802.11 node, some basic functionalities need yet to be implemented, including the complementary OFDM frame-decoder and the MAC mechanism.

III. 802.11 MAC SDR IMPLEMENTATION

Due to the flexibility offered by the USRP2 SDR platform, the multi-hop communications and networking hardware testbed currently being developed by the authors is based on the USRP2 platform. The previous section has shown that innovative research activities are currently underway to create SDR platforms over which to investigate advanced ad-hoc communications and networking techniques. However it is important to note the lack of a generic 802.11 MAC protocol implementation that can be used independently of the PHY layer. This contribution is then aimed at introducing an 802.11 MAC implementation, with the implemented code being available for the research community at http://www.uwicore.umh.es/mhop.html. In order to ensure its portability and facilitate cross-layer developments, the implemented MAC communicates with other OSI layers through sockets. This allows easily re-using and integrating our MAC implementation with other 802.11-based SDR research testbeds. The Radio Frequency (RF) front-end is the USRP2 board that incorporates a XCVR2450 dual band (2.4-2.5 GHz, 4.9-5.85 GHz) transceiver and a VER2450 antenna.

The PHY layer used is the one developed by FTW [9], although some slight functional modifications have been incorporated through the addition of carrier sensing functions and the capability of transmitting packets of different sizes. The carrier sensing function has been designed following the USRP spectrum sense example provided by GNU Radio, but adjusted to the 802.11 sensing requirements. Using the USRP2 as data source, the captured samples go through a Fast Fourier Transform (FFT) and a complex to squared magnitude block to obtain the signal's Power Spectral Density (PSD). The signal level is then computed though the integration of the PSD. This value is sent to the MAC layer which will decide the channel state (idle or busy). Additionally, the PHY layer is set up to transmit any packet generated at the MAC layer, including data, Request To Send (RTS), Clear To Send (CTS), Acknowledgement (ACK) and Beacon 802.11 frames. For this purpose, the OFDM modulator is reconfigured during the runtime execution when a packet with different number of OFDM symbols is going to be sent, contrary to the original FTW implementation that has a fixed number of OFDM symbols. Since the OFDM receiver is still under development by FTW, the authors emulated its behavior through a queue at the PHY layer that generates packets to simulate their reception and passes them to the MAC layer trough the MAC to PHY Interface (MPI) interface. The implementation of this queue has been necessary to test the correct operation of the MAC layer when a packet is received as it will be shown in the next section.

The implemented MAC layer performs the contention service or Distributed Coordination Function (DCF), which is based on CSMA with Collision Avoidance (CA) technique, to regulate the access to the shared wireless channel. The MAC layer has been fully implemented in Python language following the state machine proposed by Cisco in [10], but adding additional functionalities such as retransmission and fragmentation. The resulting finite-state machine is depicted in Figure 1, where the top of the figure represents the MAC state transition upon receiving a frame from the PHY layer, while the bottom part of the figure illustrates the MAC layer contention processes required to transmit a frame over the wireless channel. As it can be appreciated in the figure, the proposed MAC implements both the carrier sense function to determine whether the channel is idle during a DCF Inter Frame Space period (*WAIT_FOR_DIFS*), and the back off process to avoid packet collisions when several nodes try to send a packet at the same time (*BACKING_OFF*). In addition, the RTS/CTS mechanism is implemented to solve the hidden and exposed terminal problems of wireless networks, and to reserve the channel through the virtual carrier sensing or Network Allocation Vector (NAV). The current implementation also includes the network discovery process of mesh networks. This process is enabled through the periodic (with configurable intervals) broadcast exchange of beaconing messages among neighboring nodes.

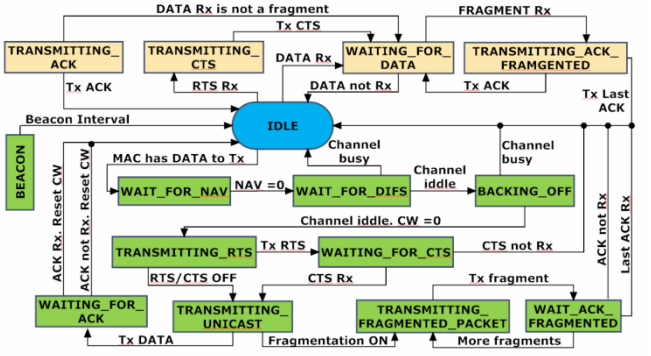

Figure 1. Finite state machine of the implemented 802.11 MAC

Assuming an initial IDLE state, three different options are possible: MAC detects that an upper layer application wants to send a packet (green states in Figure 1), the MAC notices that the PHY layer has a RTS or DATA frame received which needs to be processed (yellow states), or the MAC requires to send a beacon frame. For the first option, the node waits until the NAV value is equal zero. This NAV parameter is constantly updated by reading the duration field of incoming RTS or CTS frames. When the NAV value is equal zero, the station waits a DCF Inter-Frame Spacing (DIFS) time while performing the carrier sensing. If the channel is idle, the MAC switches to BACKING_OFF state. On the other hand, if the channel is not idle, the MAC will return to the IDLE state because another transmission has been detected and it would cause a packet collision. After performing the pseudo-random backoff algorithm, if the channel is not busy, the MAC changes to TRANSMITTING_RTS state, otherwise it will return to IDLE. In TRANSMITTING_RTS state, the node initiates the RTS/CTS protocol. If the packet length is greater than the RTS threshold, the MAC sends the RTS frame and waits for the CTS frame reception in WAITING_FOR_CTS state. Once the CTS frame has been received, the node transmits

the DATA packet and waits for the ACK response of the destination node in the WAITING_FOR_ACK state. If the packet length is greater than the IEEE 802.11 fragmentation threshold, the MPDU will be fragmented and sent in smaller packets. When the ACK arrives, the station switches to IDLE state, the packet is deleted from the upper layer queue and the transmission is marked as correct. When a DATA or RTS frame is received and an IDLE state is assumed, the MAC reception procedure is initiated. After waiting a NAV time (once the channel is idle), the station sends the proper CTS frame and waits for the incoming data. After receiving the DATA frame, independently of whether it is a fragmented packet or not, it generates and sends an ACK frame to the transmitter data station. If the destination station sends only the DATA packet instead of sending a RTS frame, the MAC detects the DATA packet and sends back the proper ACK. Once the packet is acknowledged and defragmented, it is sent to upper layer and the reception is marked as correct.

IV. VALIDATION RESULTS

Although a FPGA or an Application-Specific Integrated Circuit (ASIC) implementation would provide better timing performance, using high-level programming languages (C++ and Python) on a PC makes the system more flexible and easier to change and implement by researchers with no hardware design background. Therefore, there is a trade-off between design flexibility and software processing delay that would be partially solved with more powerful CPUs. In any case, the implemented MAC needs to be validated to demonstrate its compliance with the 802.11 standard. In this context, this section validates the implemented IEEE 802.11 MAC mechanisms using Linux debugging tools and over-the-air measurements at 5GHz (802.11a). The code is executed on a general purpose PC with 2GBytes of RAM and an Intel Core 2 CPU clocked at 2.0GHz. The PC operates under Ubuntu 9.10 with the 2.6.31-16 kernel version and the GNU Radio and Python's versions 3.3 and 2.6.4 respectively. Given the current lack of a receiver PHY layer, the implemented MAC mechanism is validated using the Wireshark packet sniffer, which allows not only to capture the transmitted frames, but also to ensure that the transmitted frames are 802.11 standard-compliant.

A. Timing characterization

The USRP2's sampling rate is faster enough to support the demanding data rates of the 802.11a standard. However, the software processing of the MAC and PHY layers necessary to allow for the capability to fully reconfigure and develop novel cross-layer mechanisms, results in important processing delays that currently prevent reaching the 802.11a commercial performance using the USRP2 SDR-based implementation. However, the USRP2-based platform still represents a unique and invaluable testbed over which to develop and test novel cooperative and cross-layer techniques through comparative SDR-based performance studies. Table I summarizes the characterization of the processing delays incurred by the transmission of a data frame. At the PHY layer, it is possible to differentiate between the *USRP2 signal forming*, which represents the OFDM modulation delay of the packet, and the *PHY processing delay*, which measures the time from the reception of a packet at the PHY layer to its delivery to the OFDM modulator. At the MAC layer, the *Carrier sensing* and the *MAC processing delays* are analyzed. The former represents the delay between the time at which the MAC requests the PHY layer to sense the channel, and the time at which the PHY layer replies with the measured value. The MAC processing delay measures the MAC state transitions until a packet is being transmitted [2] (see Figure 1). The processing delays shown in Table I require scaling synchronization parameters, such as time-slot and DIFS. The processing delays, and consequently the scaled timings, are highly dependent on the processing power capacity of the PC's CPU. Increasing such power will further reduce the characterized delays

TABLE I. PROCESSES DELAY CHARACTERIZATION

USRP2 signal forming	16.665 μs/bit
PHY processing delay	0.942 μs/bit
Socket MAC – PHY delay	1.827 ms
MAC-Data processing delay	103.232 ms
MAC-CS processing delay	20.692 ms

B. Validation trials

A set of over-the-air tests have also been conducted to validate the SDR implementation of the 802.11 MAC presented in this paper. The first one is conducted to validate that the packets generated at the MAC layer are well formed and correctly captured with the sniffer, and consequently 802.11 standard-compliant. For this purpose, an USRP2 node is configured to force the states that generate the transmission of the required packets. The USRP2 communicates using the IEEE 802.11a technology, the frequency is set up at 5.3 GHz (channel 60) and the data transmission is fixed at 6Mbps. Figure 2 shows the eight different packets that are generated at the MAC layer with the MAC SDR implementation, and detected by the wireless sniffer, including fragmented packets (Number –No.- 1 and 2), the acknowledgment of the data transmission (No. 3), the RTS and CTS frames (No. 4 and 5), the beacon frame (No. 6) and the data packet and the retransmitted packet (No. 7 and 8). The capture illustrated in Figure 2 shows that the configuration parameters (frequency, protocol, transmission rate) are correctly detected by the sniffer, which runs on the PC that is equipped with the commercial wireless card. The second trial focuses on demonstrating the correct behavior of the implemented MAC protocol during the process of sending a data packet. In particular, this section compares the sequence of messages exchanged between two commercial wireless cards, and the messages handshake between two USRP2s configured with the implemented SDR MAC. The results illustrated in Figure 3.a and 3.b confirm that the implemented MAC generates the same packets as the commercial cards, although with the previously explained processing delay.

[2] To measure the *MAC processing delay*, the back off and carrier sensing processes are disabled to avoid the exponential random delay.

No.	Source	Destination	Protocol	TX Rate (Mbps)	Freq.	Type	Retry	Fragment
1	EttusResea_03:0c	EttusResea_03:10	802.11a	6.0	5300 [A 60]	Data	Frame is not being retransmitted	More fragments follow
2	EttusResea_03:0c	EttusResea_03:10	802.11a	6.0	5300 [A 60]	Data	Frame is not being retransmitted	This is the last fragment
3		EttusResea_03:10 (RA)	802.11a	6.0	5300 [A 60]	Acknowledgement	Frame is not being retransmitted	This is the last fragment
4	EttusResea_03:0c (TA)	EttusResea_03:10 (RA)	802.11a	6.0	5300 [A 60]	Request-to-send	Frame is not being retransmitted	This is the last fragment
5		EttusResea_03:10 (RA)	802.11a	6.0	5300 [A 60]	Clear-to-send	Frame is not being retransmitted	This is the last fragment
6	EttusResea_03:0c	Broadcast	802.11a	6.0	5300 [A 60]	Beacon frame	Frame is not being retransmitted	This is the last fragment
7	EttusResea_03:0c	EttusResea_03:10	802.11a	6.0	5300 [A 60]	Data	Frame is not being retransmitted	This is the last fragment
8	EttusResea_03:0c	EttusResea_03:10	802.11a	6.0	5300 [A 60]	Data	Frame is being retransmitted	This is the last fragment

Figure 2. Frames generated at the MAC layer

No.	Time	Source	Destination	Protocol	Type
1	0.000000	LiteonTe_3a:1f:51 (TA)	Ubiquiti_84:3b:87 (RA)	IEEE 802.11	Request-to-send
2	0.000348		LiteonTe_3a:1f:51 (RA)	IEEE 802.11	Clear-to-send
3	0.000354	192.168.1.46	192.168.1.49	ICMP	Data
4	0.000357		LiteonTe_3a:1f:51 (RA)	IEEE 802.11	Acknowledgement

a) Commercial wireless card

No.	Time	Source	Destination	Protocol	Type
1	0.000000	EttusResea_03:0c (TA)	EttusResea_03:10 (RA)	IEEE 802.11	Request-to-send
2	0.086678		EttusResea_03:0c (RA)	IEEE 802.11	Clear-to-send
3	0.182696	EttusResea_03:0c	EttusResea_03:10	LLC	Data
4	0.270464		EttusResea_03:0c (RA)	IEEE 802.11	Acknowledgement

b) Implemented SDR 802.11 MAC

Figure 3. Message exchange sequence

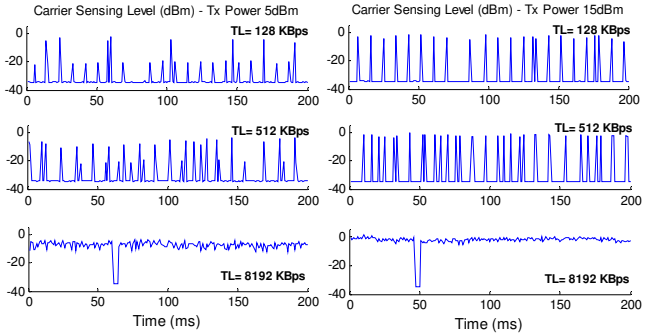

Figure 4. Carrier sensing level detected at 5.2GHz

Finally, Figure 4 aims at validating the performance of the carrier sensing mechanism since it is a critical function that detects whether a channel is idle or busy, and therefore whether an ad-hoc node is allowed to initiate the transmission of a packet. To this end, two PCs communicate with each other in ad-hoc mode in an area free of 802.11 interference, while an USRP2 is placed in the middle of both PCs and periodically senses the channel. Figure 4 shows the sensed signal level for two different PC's output transmission power and traffic loads (TL). The depicted results show that the implemented USRP2 carrier sensing function is capable to correctly detect the variations in both the transmission power and traffic load.

V. CONCLUSIONS

This work has proposed and evaluated an 802.11 MAC SDR implementation for experimental communications and networking research, with the code made available to the research community. The implemented MAC can be fully configured and modified, enabling the development of advanced cross-layer and cooperative communications and networking techniques. The trials carried out to validate the DCF MAC implementation have confirmed the timing restrictions of the SDR platform, but it has also been shown that the proposed MAC protocol is compatible with commercial wireless cards, and is IEEE 802.11 standard-compliant. The 802.11 SDR-based platform represents then a very valuable research testbed over which to investigate advanced cooperative and cross-layer techniques.

ACKNOWLEDGMENTS

This work has been supported by the Ministry of Science and Innovation (Spain) and FEDER funds under the project TEC2008-06728, by the Generalitat Valenciana under the project ACOMP/2010/111 and ACIF/2010/161, and by the Ministry of Industry, Tourism and Trade (Spain) under the project TSI-020400-2008-113 (CELTIC proposal CP5-013).

REFERENCES

[1] C.S Ram-Murthy and B.S. Manoj, *Ad Hoc Wireless Networks*: *Architecture and Protocols*, *Prentice Hall*, 2004.

[2] IEEE 802.11-2007. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.

[3] B. Coll and J. Gozalvez, "Energy Efficient Routing Protocols for Multi-Hop Cellular Networks", *in Proceedings of the IEEE Personal, Indoor and Mobile Radio Communications Symposium (PIMRC'09), pp.* 13-16, 2009.

[4] B. Coll and J. Gozalvez, "Neighbor Selection Techniques for Multi-Hop Wireless Mesh Networks", *in Proceedings of the IEEE International Workshop on Wireless Local Networks (WLN'09), pp. 20-23*, 2009.

[5] K. Amiri, et al., "A Unified Wireless Network Testbed for Education and Research", *in Proceedings of the IEEE Microelectronic Systems Education (MSE07)*, pp. 53-54, 2007.

[6] D. A. Scaperoth, "Configurable SDR Operation for Cognitive Radio Applications using GNU Radio and the Universal Software Radio Peripheral", M.*Sc. thesis, Virginia Tech,* 2007.

[7] T. Korakis, et al., "Cooperative Network Implementation Using Open-Source Platforms", *IEEE Communication Magazine, vol.47, no.2*, pp.134-141, 2009.

[8] K. Mandke, et al., "Early Results on Hydra: A Flexible MAC/PHY Multihop Testbed", *in Proceedings of the IEEE Vehicular Technology Conference (VTC07), pp. 1896-1900*, 2007.

[9] P. Fuxjäger, et al., *"IEEE 802.11p Transmission Using GNURadio"*, *in Proceedings of the IEEE Karlsruhe Workshop on Software Radios (WSR10)*, pp. 1-4, 2010.

[10] P. Roshan and J. Leary, *802.11 Wireless LAN Fundamentals, Cisco Press*, 2003.