Reference:


“A SURVEY ON PROTOTYPING PLATFORMS FOR THE DEVELOPMENT AND EXPERIMENTAL EVALUATION OF MEDIUM ACCESS CONTROL PROTOCOLS”

F. Vázquez Gallego1, J. Alonso-Zarate1, Ch. Verikoukis1, and L. Alonso2

1 Centre Tecnològic de Telecomunicacions de Catalunya (CTTC)
Parc Mediterrani de la Tecnologia (PMT), Av. Carl Friedrich Gauss 7
08860 Castelldefels (Barcelona), Spain
francisco.vazquez@cttc.es, jesus.alonso@cttc.es, cveri@cttc.es

2 Department of Signal Theory and Communications
Universitat Politècnica de Catalunya (UPC) – EETAC – BarcelonaTECH
08860 Castelldefels (Barcelona), Spain
luisg@tsc.upc.edu

ABSTRACT

The key role that the Medium Access Control (MAC) layer plays towards the performance of wireless communication networks justifies the vast amount of MAC protocols which can be found in the literature. Indeed, most of these works evaluate the performance of the proposals by means of either theoretical analysis or computer-based simulation. Unfortunately, theoretical models typically adopt some idealistic assumptions for the sake of mathematical tractability, and usually lack physical (PHY) layer accuracy, thus leading to inaccurate results and misleading conclusions regarding the actual performance of communication protocols. These limitations can be overcome through experimental prototyping and testing, which enable the actual assessment of the protocols under realistic PHY layer conditions. This paper identifies the key functions to be supported by a MAC prototyping platform, and presents a qualitative description and a comprehensive comparative study of existing experimental platforms for the development and assessment of MAC protocols for wireless communication networks.
1 INTRODUCTION

Medium Access Control (MAC) protocols define the rules which nodes of a communications network should follow to get access to a shared medium in order to transmit or receive data. The fact that the performance of a network is highly influenced by the MAC protocol has motivated many researchers to design innovative solutions in order to satisfy the increasingly demanding requirements of a wide range of applications. Most of the research effort carried out in the design of MAC protocols for wireless communications over the last years has been based on theoretical analyses and computer-based simulation. Unfortunately, these approaches have certain limitations. While theoretical models typically adopt some simplified assumptions for the sake of mathematical tractability, computer-based simulations usually lack physical (PHY) layer accuracy, thus leading to inaccurate results and conclusions regarding the actual performance of communication protocols [1]. These limitations can be overcome through experimental prototyping and testing, which enable the actual assessment of the protocols under realistic PHY layer conditions. However, the prototyping capabilities are beyond the reach of the majority of researchers due to the need of a multidisciplinary skilled team capable of creating an experimental platform to translate the MAC models into real hardware and software implementations. Despite those facts, we can find in the literature some experimental platforms. In particular, we can mention that in [4], [5] and [6] the software drivers of commercial IEEE 802.11 Network-Interface Cards (NIC) have been modified to conduct incremental research on protocols based on the IEEE 802.11 standard. These solutions do not provide flexibility to change the PHY layer implementation and only minimum MAC functionalities can be customized due to, e.g., host-computer latencies.

Motivated by the limitations of NIC-based platforms, several research groups have worked in the development of flexible custom platforms, i.e., embedded Software-Defined-Radio (SDR) platforms. An overview of existing experimental platforms for cognitive radio applications can be found in [2] and [3]. However, to the best of our knowledge, there is yet no qualitative comparison analysis of the different alternatives for prototyping at the MAC layer. This is the motivation for the work presented in this paper, where we present a qualitative description of the most relevant existing platforms for the development and experimental assessment of MAC protocols for wireless communication networks.

The remainder of this paper is organized as follows. In Section 2, we identify the key functions that have to be efficiently supported by a MAC prototyping platform. In Section 3, we compare the features of commercial NIC-based platforms versus embedded-SDR platforms, listing their pros and cons, and review the most relevant existing embedded-SDR solutions for MAC prototyping, focusing on how each platform supports the implementation of the key functions. Finally, Section 4 summarizes the paper and discusses open research issues.

2 FUNCTIONAL REQUIREMENTS FOR A MAC PROTOTYPING PLATFORM

The functional requirements that a MAC prototyping platform must meet are tightly related to the functional operation of MAC protocols. These protocols can be classified into three groups: contention-based, reservation-based, and hybrid protocols, which are a combination of the other two.

In contention-based protocols, nodes compete for the use of the medium, leading to collisions when multiple nodes make a simultaneous transmission attempt. ALOHA and Carrier Sense Multiple Access (CSMA) are the classical schemes of contention-based protocols. In ALOHA, a node transmits whenever it has data to send, and in case of collision, it retries after a random period. In CSMA, a node having data to send first senses the channel. In the case that the channel is clear, the node starts transmitting. Otherwise, the node postpones its transmission depending on the type of CSMA. For example, for 1-persistent CSMA, the node keeps sensing the channel until idle, but for the variation used in the IEEE 802.11 Standard, the node also initiates a backoff after the channel becomes idle. In addition, some Collision Avoidance (CA) techniques may be implemented. As an example, CSMA/CA uses a handshaking of Request-to-Send (RTS) and Clear-to-Send (CTS) control packets before transmitting the data, which are acknowledged (ACK) by the destination.
In reservation-based protocols, a schedule is established to allow nodes to communicate without collisions. Time-Division Multiple Access (TDMA) and Frequency-Division Multiple Access (FDMA) are usually combined. In TDMA, time is divided into time-slots and each node is assigned a time-slot to transmit. In FDMA, different frequency-channels can be used by different nodes.

Based on the above classification, the key functions of a MAC protocol (i.e., time-critical or delay-sensitive functions) that a prototyping platform should support are:

- **Access to PHY layer information**: The platform must have quick access to information related to the PHY layer. For example, access to the received signal strength indicator (RSSI) in order to execute functions, such as the Clear-Channel-Assessment (CCA), which allows deciding whether the channel is idle or busy.

- **PHY layer re-configuration**: The MAC layer may need to rapidly re-configure various PHY parameters such as data-rate, modulation, transmit-power, and frequency-channel. In addition, it must be possible to switch the radio-frequency (RF) front-end operation among transmit, receive, and idle modes.

- **Precise scheduling and timing**: In TDMA-like protocols, precise scheduling must be guaranteed to ensure that transmissions occur during time-slots. In the case of contention-based protocols, precise timing is also required to execute backoff algorithms, define contention windows, and leave inter-frame spacing (IFS) to compensate propagation delays. As an example, in the context of the IEEE 802.11 Standard, an ACK packet must be transmitted in a Short Inter Frame Space (SIFS) interval, i.e., tens of microseconds, after the reception of a data packet.

- **Fast identification of control packets**: The incoming control packets (e.g., RTS, CTS, ACK) have to be quickly detected and identified to fulfill protocol time-constraints.

- **Fast packet transmission**: In order to reduce collisions in contention-based protocols, packets must be transmitted with minimum delay when the channel is detected to be clear. In addition, the control packets which are responses to received packets must be quickly transmitted to fulfill the SIFS requirements.

### 3 MAC PROTOTYPING PLATFORMS

State-of-the-Art experimental platforms used for MAC protocols prototyping can be classified into two groups, namely:

1. Commodity NIC-based platforms.
2. Embedded SDR-based platforms.

Besides these two types of MAC prototyping platforms, emulator-based platforms are testbeds devised to conduct large wireless network experiments, e.g., ORBIT and CMU. The analysis of this kind of platforms is out of the scope of this paper.

#### 3.1 COMMERCIAL NIC-BASED PLATFORMS

Commodity NIC-based platforms (e.g., [4]-[6]) typically divide the implementation of the MAC layer functions between the hardware and the firmware on the networking card, which implement the time-critical functions, and the software-driver running on the computer hosting the card, which implements the time-tolerant functions.

**Advantages** - NIC-based solutions are low-cost and facilitate reprogramming some MAC protocol functions by modifying the software-driver installed in the host-computer (as MadWifi driver based on Atheros chipsets, and HostAP driver based on Intersil Prism chipsets). In addition, since these NIC-based solutions are commercial networking cards, they are already
equipped with an IEEE 802.11 Standard-compliant implementation that makes it possible to compare new protocols with legacy ones.

**Disadvantages** - The connection between the host-computer and the NIC introduces unpredictable delays and latencies, which may compromise precise scheduling and compliance with the strict timing requirements of MAC protocols. For this reason, time-critical functions are programmed in the NIC firmware. Due to Intellectual Property Rights issues, NIC manufacturers do not provide the tools required to modify this firmware, thus limiting the flexibility to efficiently customize certain MAC functionalities. This limitation to modify the firmware is extended to the access to the PHY layer for information and configuration, thus not allowing MAC-PHY cross-layer designs.

Therefore, experimental platforms based on commercial NICs are only convenient to test incremental corrections on 802.11 standard protocols. This limitation has led researchers to develop solutions based on embedded-SDR, as explained in the next section.

### 3.2 EMBEDDED-SDR PLATFORMS

Embedded-SDR are custom-hardware platforms equipped with an analog RF front-end connected to a built-in computing subsystem by means of analog-to-digital and digital-to-analog converters. The computing subsystem performs both the PHY digital base-band processing and the MAC functions. It is based on Field-Programmable-Gate-Arrays (FPGAs), embedded processors and/or Digital-Signal-Processors (DSP).

**Advantages** – They offer full flexibility and control over MAC layer functionalities. Protocols are fully programmable in an embedded processor, and the implementations based on FPGA also allow partitioning the MAC protocol into custom hardware-accelerators, offering even more processing power. This control over the MAC may be extended to the PHY, allowing to implement MAC-PHY cross-layer techniques and to experiment with different PHY layers.

**Disadvantages** - The implementation costs of custom-hardware are very high and its use shows a steep learning curve. The protocol designer needs to understand the hardware details to develop optimized C-code (or even assembler) to fulfill the tightest protocol time-constraints. In general, MAC designers are familiar with high-level programming languages like C++, which may endanger the implementation of time-critical functions. However, e.g., the use of FPGAs allows hardware-acceleration and may relax the software optimization requirements, but forces the protocol designer to learn a Hardware Description Language (HDL). Besides complexity, any protocol development has to be done from scratch, thus implying a great effort to get implementations fully in compliance with standards.

Table 1 summarizes and compares the advantages and disadvantages of embedded-SDR and NIC-based platforms. Table 2 describes how both types of platforms support the implementation of the key MAC functionalities described in Section 2.

Several research groups have developed embedded-SDR platforms for MAC protocols prototyping and for cognitive radio applications [2] [3], e.g., Universal Software Radio Peripheral (USRP) and GNU Radio, Wireless Open-Access Research Platform (WARP), CalRadio, Berkeley Emulation Engine (BEE), Iris, Kansas University Agile Radio (KUAR), OpenAirInterface, etc. Due to space limitations, this section reviews the most relevant and representative platforms ([7]-[13]), and focuses on how each platform addresses the requirements previously described in Section 2.

#### A. TUTWLAN

TUTWLAN [7] is an experimental platform for developing hybrid MAC protocols over Wireless Local Area Networks (WLAN).

**Implementation** - TUTWLAN consists of two elements: a commercial Altera FPGA-based board, and a MAC-less Prism HW1151 radio module. The radio module is compatible with the
IEEE 802.11b standard. It contains the PHY layer, integrating an RF transceiver and a baseband modem, but no MAC layer functionalities.

The FPGA contains an Advanced RISC Machine (ARM) processor and a custom digital circuitry block, i.e., Programmable Logic Device (PLD). The MAC protocol is C-coded and runs on the ARM processor. The PLD block is coded in VHDL and connects the processor with the radio module (radio interface) and with a host computer (application interface). The communication between the processor and the PLD is performed through a dual-port Static Random Access Memory (SRAM) and an Advanced High Performance Bus (AHB) bridge. The architecture of TUTWLAN is shown in Figure 1.

The TUTWLAN design flow consists of protocol formal modeling, using Specification and Description Language (SDL), and automatic translation to C-code by using the commercial Telelogic TAU software.

- **Access to PHY layer information:** The radio interface, implemented inside the FPGA, provides RSSI measurements from the radio module to the ARM via the dual-port SRAM.

- **PHY layer re-configuration:** The radio module can be configured (e.g., operation mode, frequency-channel, etc.) by sending control messages from the ARM processor to the radio interface via the AHB bridge. The speed of these interfaces is crucial to configure PHY on a per-packet basis or only during the initialization process.

- **Precise scheduling and timing:** The radio interface contains hardware-timers to control the TDMA frame cycle, and to precisely determine the transmission times. The ARM reads the hardware-timer to know the current frame cycle. As an example, in the TUTMAC protocol [7] implementation, the hardware-timer is reset when a beacon packet is received, i.e., frame synchronization.

- **Fast identification of control packets:** The radio interface is designed to quickly identify beacon packets for TDMA frame synchronization. This hardware identification avoids software delays and latencies introduced in the packet transfer to the processor through dual-port SRAM. In addition, the radio interface integrates other hardware-accelerators to speed-up intense computation operations, e.g., Cyclic-Redundancy-Code (CRC), Forward-Error-Correction (FEC), data-encryption, etc.

- **Fast packet transmission:** Besides hardware-accelerators, a Direct-Memory-Access (DMA) controller in the radio interface enables fast reading of data and control packets from dual-port SRAM during transmission. However, the control packets (e.g., CTS, ACK) which are responses to received packets could be more rapidly transmitted by being automatically generated by the radio interface. The latency in the dual-port SRAM transfers may introduce high delays in the initialization of transmission from the processor, thus increasing the probability of collisions. Therefore, the radio interface hardware could improve this operation and initialize the transmission faster.

**Discussion** - The use of an FPGA to implement the radio interface provides full re-configurability to design custom hardware-accelerators to address precise scheduling and time-critical functions, but sound knowledge in HDL is needed. In addition, the radio interface offers flexibility to experiment with physical layers different from the 802.11b standard. However, it requires manufacturing a custom-made radio connector card to plug the radio module into the FPGA board. In this sense, the design of the interface to the radio module is usually a complex procedure due to the great number of interface registers, control signals, bus timings and bus-widths adaptation.

Regarding the TUTWLAN design flow, the conversion from SDL to C-code may considerably decrease development times. However, the performance of the C-code may not be optimal in terms of processor execution times and memory requirements. Thus, the most complex and delay-sensitive algorithms may need to be implemented directly using optimized C-functions. As an example, the performance of the TUTMAC protocol is verified in [8]. The IFS duration was actually longer than the estimated value from theoretical calculations (60 μs).
B. USRP/USRP2 and GNU Radio

The authors in [9] integrate the USRP platform with GNU Radio [14] for the development of PHY and MAC layers on the software of a host-computer.

Implementation - The USRP implements the RF front-end functionality, together with analog-to-digital and digital-to-analog conversions. It includes an FPGA for digital up/down conversion from base-band to IF (Intermediate Frequency) and vice versa. However, it does not implement any PHY layer base-band processing, which is made at the host-computer software using GNU Radio. The base-band samples are exchanged between the host-computer and the USRP FPGA through a Universal Serial Bus (USB) port. The USRP2 is connected to the host-computer using Gigabit-Ethernet. This interface introduces important non-deterministic delays between the host-computer and the RF front-end, which limit the implementation of key MAC functionalities. The architecture of USRP/USRP2 is shown in Figure 2.

GNU Radio was originally designed to support the development of PHY layer base-band processing algorithms in Python and C++. It is an open-source toolkit for building software radios on computers connected with the USRP. The GNU Radio library provides signal processing blocks for modulation, demodulation, filtering, etc.

- **Access to PHY layer information:** The performance of carrier sensing is limited by the delay between the RF front-end and the computer software, the processing delay of the energy-estimation algorithm, and the transmission delay from the software once it detects the medium to be idle. Therefore, these delays limit how quickly the MAC protocol can respond to the changes in channel conditions.

- **PHY layer re-configuration:** It is possible to configure any of the PHY layer parameters.

- **Precise scheduling and timing:** Besides the delays associated to data transfers and processing, the host-computer software and the operating-system introduce significant delay jitter. This jitter prevents precise control over the scheduling of time-slots and over the timing of packet transmissions.

- **Fast identification of control packets:** It is limited by the delay between the RF front-end and the host-computer, and the software delay of the processing algorithms involved in demodulation and identification.

- **Fast packet transmission:** It is limited by the delay of the signal processing involved in modulation, and the delay between the host-computer and the RF front-end.

Discussion – GNU Radio offers a high degree of flexibility to implement PHY layer processing algorithms. In addition, several low-cost off-the-shelf RF transceiver boards are available operating at different frequency bands. However, the key MAC functions are difficult, or even impossible, to entirely implement it in the host-computer’s software, due to the high-latency and low-bandwidth of the interface bus. Not only the latency is high, but it is non-deterministic, leading to an unavoidable uncertainty in performance. In order to avoid the delay problems, the MAC protocol key functions could be moved from the host-software to the FPGA integrated in the USRP [10].

C. WARP

WARP [11] is a reconfigurable platform to implement PHY, MAC, and network layer protocols.

Implementation - WARP is composed of an FPGA-board constituted by a Virtex-4 FPGA with an embedded PowerPC processor. The PowerPC provides a complete programming environment for MAC and network layer design. WARP includes an Ethernet connection to provide a traffic source and sink to the platform, and to offload experiment measures to a host-computer. The FPGA-board includes an interface to four radio daughter-cards to enable Multiple-Input Multiple-Output (MIMO) PHY layer configurations. The radio daughter-cards
integrate dual-band 802.11 a/b/g RF transceivers capable of targeting 2.4 GHz and 5 GHz bands with bandwidths up to 40MHz. The architecture of WARP is presented in Figure 3.

The radio daughter-cards do not include any PHY base-band processing, which is integrated in the FPGA as a peripheral of the PowerPC. The PHY base-band processing is modeled using Matlab/Simulink and then automatically converted to VHDL using Xilinx System-Generator. MAC protocols are C-coded on the PowerPC.

A custom PHY reference model, similar to the 802.11a standard but not fully compliant, has been implemented on WARP by Rice University. Firstly, it can be configured as a Single-Input Single-Output (SISO) Orthogonal-Frequency Division Multiplexing (OFDM) PHY layer using one radio daughter-card. The second PHY configuration is a 2 x 2 MIMO OFDM using two radio daughter-cards. The custom PHY reference models, MAC source-code, platform support packages, and design documents are available in the open-access repository at WARP website.

- **Access to PHY layer information**: The MAC protocol running in the PowerPC can quickly access to several parameters (e.g., RSSI) provided by the PHY peripheral via bus transactions and interrupts.

- **PHY layer re-configuration**: The PowerPC can rapidly configure PHY parameters, e.g., data-rate, modulation. In addition, software can select the center-frequency of the radio daughter-cards and switch among transmit and receive modes.

- **Precise scheduling and timing**: A hardware-timer is used to precisely control time-slots, backoff-periods, inter-frame spacing, etc. The timer is configured for a maximum time, and the corresponding software-handler is called upon the completion of that time.

- **Fast identification of control packets**: Received packets are stored in a packet-buffer. The identification of control packets can either be done by the PowerPC, with the associated transfer delays and software-latencies, or accelerated with an auto-responder in the FPGA. The auto-responder can be configured to identify specific packet types (e.g., RTS, CTS, ACK and data).

- **Fast packet transmission**: The CCA can be implemented in software by monitoring the RSSI provided by the PHY peripheral. In this way, software-latencies can delay the transmission of packets after software detects the channel to be clear. Therefore, it would be better to run the CCA process with hardware-accelerators. Regarding the response to received packets, it can be performed by software, or by the auto-responder, which can be configured with an automatic response-packet to specific incoming packets (e.g., ACK response to data).

**Discussion** - WARP is a powerful platform designed to leverage custom designs of OFDM PHY base-band processors and high-performance hybrid MAC protocols. In addition, the daughter-cards slots in the FPGA-board offer the possibility to change the radio daughter-cards. However, this flexibility comes at the cost of requiring a great design effort in order to get a solution fully-compliant with standards.

The use of an FPGA provides full flexibility to design custom hardware-accelerators attached to the PowerPC (e.g., auto-responder) to speed-up MAC time-critical functions, however, that requires expertise in VHDL or Xilinx System-Generator. The latter provides greater code productivity, but the implementation is less optimal than VHDL-coding in terms of FPGA density.

The first performance evaluation of WARP has been presented in [11] by measuring turn-around times (TAT), i.e., time between the end of a packet reception and the start of a packet transmission. The mean TAT is 52 µs, which is still longer than the 10 µs required by the IEEE 802.11 Standard, but much lower than in NIC-based platforms, which typically are above 150 µs. As an example of hardware-acceleration, TAT may be reduced below 20 µs by using the auto-responder.

**D. CalRadio**
CalRadio [12] [13] is a flexible 802.11b wireless platform for research and evaluation of 802.11 MAC protocols.

**Implementation** - CalRadio is composed of two hardware elements: a main board and a radio daughter-board. The main board includes the processing core and uses an energy-efficient DSP, i.e., TMS320VC5471 from Texas Instruments, where the MAC functionalities are C-coded. Even intense computation functions are performed by the DSP (e.g., CRC). Furthermore, the DSP integrates an ARM processor for the implementation of network and transport layer functionalities. The ARM and the DSP communicate through shared-memory buffers.

The radio daughter-board is constituted by a radio-chipset which fully implements the 802.11b physical layer, including RF front-end and base-band processing. The architecture of CalRadio is shown in Figure 4.

- **Access to PHY layer information**: CalRadio provides an interface between MAC software and PHY through the radio-chipset registers. Each received packet has certain associated parameters, such as the RSSI, which may be read from the DSP.

- **PHY layer re-configuration**: The radio-chipset allows the MAC software to control important PHY parameters by writing to configuration registers for each packet: data-rate, preamble length, CCA thresholds, frequency-channel, and transmit-power.

- **Precise scheduling and timing**: The DSP contains a hardware-timer, configured from the MAC software, which can be used to control arbitrary time intervals.

- **Fast identification of control packets**: Packets are fully checked and identified by the DSP software with no hardware-acceleration. This approach may introduce delays due to packet-transfers between the radio-chipset and the DSP, besides inherent software-latencies of non-optimized C-code.

- **Fast packet transmission**: CCA is performed by the radio-chipset, which provides a status signal to the DSP indicating if the channel is clear. In CSMA protocols, the DSP checks the CCA status before transmitting. Therefore, minimum software-delay must be guaranteed since the channel is clear until packet is transmitted.

**Discussion** - The split between main board and radio daughter-board offers the possibility to experiment with physical layers different from the 802.11b standard. However, it requires manufacturing an interface to adapt new radio boards to the main board connector. In addition, the design of software-drivers for new radio boards is usually complex due to the great number of interface registers, control signals, etc.

Due to the limited amount of hardware resources, software implementation of MAC protocols in CalRadio may require strong code optimization in order to implement strict time requirements of key MAC functions. For example, turn-around times may not be in compliance with the ones required by the 802.11 standard. Therefore, the protocol designer needs to understand the hardware details to develop optimized C-code which fulfills the tightest time-constraints.

### 4 CONCLUSIONS AND OPEN RESEARCH ISSUES

In this paper, several existing alternatives of prototyping platforms for the experimental assessment of MAC protocols for wireless communications have been reviewed. Two main categories of MAC prototyping platforms have been identified, i.e., commercial NIC-based and embedded SDR-based platforms, and their advantages and disadvantages have been analyzed.

Mixed hardware/software architectures of embedded-SDR platforms provide full flexibility and re-configurability, and make them the best choice to support key MAC functions. However, their cost of design and implementation, the expertise and the steep learning curve required for protocol prototyping, have forced researchers to employ off-the-shelf hardware as an
inexpensive prototyping environment based on commercial Network-Interface Cards (NIC), which only provide a limited flexibility and partial control over the MAC layer.

The challenge is to create low-cost experimental platforms, based on the embedded-SDR approach, which can simplify the prototyping process to accelerate the protocol development, guarantee the fulfillment of precise MAC timing-constraints, and provide a flexible interface to PHY layer devices. The following open research issues are identified.

**MAC protocol design flows** – Innovative platforms could allow developing protocols entirely in a high-level language (e.g., C++) with hardware transparent to the protocol designer. The platform could include a set of optimized software-functions which abstract accesses to hardware. In addition, protocol simulation could be integrated in the design flow with a fast conversion from models into implementation.

**Fulfillment of MAC time-constraints and scheduling** - Innovative hardware/software architectures could release the software MAC from processing high-volume data-payloads, but instead, delegate the packet processing to hardware-accelerators. Software could access only to short packet-descriptors and packet-headers. Regarding timing and scheduling, it could be simplified by doing the management of hardware-timers transparent to the designer, e.g., the time-stamps approach used in TUTWLAN [7].

**Interface to PHY** – In order to reduce software-latencies, the platforms could use an interface based on hardware-accelerators for PHY configuration and CCA. In this sense, the PHY baseband processors (BBP) could be tightly coupled to the MAC/PHY interface inside the FPGA. However, it is difficult to find off-the-shelf PHY solutions providing low-cost RF front-end and open-source BBP (e.g., WARP).

**Implementation cost** – In order to reduce effort and cost, inexpensive off-the-shelf FPGA-based development boards with an embedded processor (e.g., PowerPC, Nios-II) could be considered instead of custom-hardware design.

**Deployment and mobility** – Low-power platforms could facilitate the deployment of battery-powered autonomous network-nodes. In this sense, small form-factor specifications could be considered for nodes mobility.

**Experimental measurements** – The platforms could enable experimental measurements to quantitatively compare the protocols energy-efficiency (e.g., energy consumption per transferred bit) with regard to their specific implementation.

To conclude, the review of the State of the Art shows that there is still a long road ahead towards the development of low-cost and low-complexity prototyping platforms that bridge the design of MAC and higher layer protocols with the actual implementation of prototypes. The development of such platforms will help leverage innovation in the design of novel communication protocols under realistic conditions.

5 REFERENCES


Figure 1: TUTWLAN platform architecture

Figure 2: USRP/USRP2 and GNU Radio architecture

Figure 3: WARP platform architecture
Table 1. Comparison table of NIC-based platforms vs. embedded SDR-platforms.

<table>
<thead>
<tr>
<th>Cost and ease of platform development</th>
<th>NIC-based platforms</th>
<th>Embedded-SDR platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ease of protocol prototyping</td>
<td>Easy and cheap.</td>
<td>Difficult and expensive.</td>
</tr>
<tr>
<td>Control over the MAC layer</td>
<td>Not fully programmable host part, only partial control over the MAC layer part on the NIC.</td>
<td>Full flexibility, re-configurability, and control over the MAC layer.</td>
</tr>
<tr>
<td>Control over the PHY layer</td>
<td>Impossible to change or customize the 802.11 PHY layer.</td>
<td>Flexible interfacing to experiment with different PHY layer devices.</td>
</tr>
<tr>
<td>Capability for cross-layer design</td>
<td>MAC/PHY cross-layer design is impossible.</td>
<td>MAC/PHY cross-layer design is possible.</td>
</tr>
<tr>
<td>Type of MAC protocols and compatibility with standards</td>
<td>Only reasonable to conduct research on protocols similar to the original protocol of the NIC.</td>
<td>Development of MAC and PHY protocols has to be done from scratch.</td>
</tr>
</tbody>
</table>

Table 2. Support of key MAC functions in NIC-based and in embedded SDR-platforms.

<table>
<thead>
<tr>
<th>Access to PHY layer information</th>
<th>NIC-based platforms</th>
<th>Embedded-SDR platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHY layer re-configuration</td>
<td>Not configurable from the software-driver level.</td>
<td>Full PHY layer configurability.</td>
</tr>
<tr>
<td>Precise scheduling and timing</td>
<td>Embedded on the NIC firmware. Difficult to get a precise scheduling at the software-driver level.</td>
<td>Custom hardware elements can be designed to get an easy and precise scheduling. Another option is to use generic processor hardware timers and interrupts management.</td>
</tr>
<tr>
<td>Fast identification of control packets</td>
<td>Embedded on the NIC firmware. Unpredictable delays at the host-interface make it difficult to meet MAC time-constraints at the software-driver level.</td>
<td>Hardware/software partitioning and code optimization are critical to meet MAC time-constraints.</td>
</tr>
<tr>
<td>Fast packet transmission</td>
<td>Embedded on the NIC firmware.</td>
<td></td>
</tr>
</tbody>
</table>