All opinions expressed in this document are those of the author individually and are not reflective or indicative of the opinions and positions of any author’s present or past employer and/or its affiliated companies (hereinafter referred to as “employer”). The technology described in this document is or could be under development and is being presented solely for the purpose of soliciting feedback.

The content and any information in this document shall in no way be regarded as a warranty or guarantee of conditions of characteristics. This document reflects the current state of the subject matter and may unilaterally be changed by the employer at any time according to its entitlement to dispose.

Unless otherwise formally agreed with the entitled employer, the employer assumes no warranties or liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party with respect to the content and information given in this document.
Experimental Performance Evaluation of a Cooperative ARQ Scheme using OpenMAC

F. Vázquez-Gallego*, D. Gajic†, J. Alonso-Zarate*, C. Liss†, L. Di Gregorio†, and Ch. Verikoukis∗

∗Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), Castelldefels, Barcelona, Spain
†InnoRoute GmbH, Munich, Germany

{francisco.vazquez, jesus.alonso, cveri}@cttc.es
{gajic, liss, digregorio}@innoroute.de

Abstract—This paper experimentally validates the efficiency of Cooperative Automatic Retransmission reQuest (C-ARQ) versus traditional ARQ protocols in terms of delay, throughput, and energy consumption. We report on our development of a testbed for obtaining these results, which is based on OpenMAC, a platform we have designed for rapid prototyping of MAC algorithms. With this work we demonstrate that OpenMAC can provide high levels of abstraction and flexibility while effectively operating novel protocol drivers under field testing by sustaining real-time execution of device-independent software prototypes.

Index Terms—OpenMAC, C-ARQ, performance evaluation, experimental.

I. INTRODUCTION

The progressive shift of functionality from hardware to software has been a steady trend in the electronic industry, and when nowadays demand for a new functionality materializes, a software implementation on an embedded processor core becomes often the preferred choice within the available spectrum of possibilities. This situation is clearly visible in the development of Medium Access Control (MAC) protocol drivers. While a plethora of proposals has been studied, only in sporadic cases these proposals have been tested in the field, a step which is widely acknowledged as necessary to obtain reliable performance figures.

In this paper, we provide field tested performance figures for comparing conventional Automatic Retransmission reQuest (ARQ) versus Cooperative ARQ (C-ARQ) protocols, and we report on the implementation of the testbed based on OpenMAC [1], which is a platform supporting the migration from hardware to software with a high degree of abstraction and flexibility.

Next to experimentally validate the superiority of C-ARQ versus ARQ, our work demonstrates that the abstraction capabilities of OpenMAC support PHY-MAC cross-layer design, for experimenting cooperative MAC schemes, without hampering the hard real-time execution of software prototypes.

The remainder of this paper is organized as follows. In Section II, we provide indications on prior work about rapid prototyping of MAC protocols. Section III briefly describes the C-ARQ scheme and the network model. Section IV describes the implementation of the OpenMAC-based network testbed, including a description of the OpenMAC architecture to make this paper self-contained. Section V provides the experimental performance evaluation. Finally, Section VI concludes the paper.

II. RELATED WORK

A number of research works have addressed the implementation of hardware platforms to test the MAC layer [2]. Two categories of platforms have been identified: those based on commercial Network Interface Cards (NIC), and those based on custom hardware. NIC-based platforms [3] provide little flexibility to customize the MAC functionalities due to unpredictable latencies at the interface between the NIC and the computer hosting the card. Contrarily, custom hardware platforms (e.g., WARP [4]) are equipped with analog Radio-Frequency (RF) front-end and a computing subsystem, e.g., based on embedded processors or Field-Programmable-Gate-Arrays (FPGAs). They offer full flexibility and control over the MAC layer functionalities at the price of exposing the whole hardware details to the programming model. In software engineering, this is a widely acknowledged weakness leading often to race conditions, which must be prevented either by painstaking handcrafted low level design or by software layers at the cost of largely eroding available cycle budgets.

In our previous work [1], the OpenMAC architecture is presented as an alternative to custom hardware platforms to smooth the learning curve to start prototyping by developing device-independent C/C++ code on top of a hardware abstraction. OpenMAC provides a flexible framework that enables a tight interaction between MAC and physical layers, which is of fundamental importance for prototyping complex MAC protocols such as cooperative and cognitive schemes [5].

The concept and theoretical bounds of C-ARQ schemes have been widely studied in the literature [6]–[9]. Most of these works are based on theoretical analysis and computer simulation. However, only a few works have been dedicated to the experimental evaluation of C-ARQ on real hardware. The work in [3] presents the first implementation of the CoopMAC protocol [9] on WARP and on a NIC-based platform. Results show that CoopMAC outperforms the MAC protocol of the 802.11 Standard. In a more recent work [10],
a proof-of-concept of the Persistent Relay Carrier Sensing Multiple Access (PRCSMA) protocol [11] is implemented using NICs. To the best of our knowledge, previous works on experimental evaluation of C-ARQ schemes mainly focus on delay and throughput results. However, no work evaluates the energy efficiency of C-ARQ schemes compared to that of conventional ARQ.

III. C-ARQ SCHEME OVERVIEW

The C-ARQ scheme that we have considered in this work is very similar to PRCSMA [11]. We consider a single-hop network formed by a source node, a destination node, and one or more relay nodes. The source node transmits data packets to the destination. The relays overhear all the transmissions between source and destination. Whenever the destination node receives a data packet with errors, it broadcasts a Call for Cooperation (CFC) packet which is overheard by all nodes in the network. Then, the relays retransmit the original data packet until the destination receives the original error-free packet and acknowledges it with an ACK packet. The source node transmits a new data packet once it receives an ACK to the previous one. An example of the C-ARQ scheme is shown in Figure 1.

The number of required retransmissions depends on the channel conditions between the source and the destination, the source and the relay, and the relay and the destination. We assume that since the control-packets (i.e., CFC, ACK) are short, they are always transmitted without errors due to channel impairments.

The nodes can be in three modes of operation: (i) transmitting a packet, (ii) receiving, or (iii) idle listening. The associated power consumptions are \( p_{tx} \), \( p_{rx} \), and \( p_{ea} \), respectively. We assume that \( p_e = p_{rx} \).

IV. IMPLEMENTATION

A. OpenMAC Architecture

The architecture of OpenMAC [1] is shown in Figure 2. The MAC protocol is partitioned into the Software (SW) MAC and the Hardware (HW) MAC modules.

The SW MAC implements non time-critical functions in C/C++ code on an embedded processor. The HW MAC executes time-critical and time-consuming MAC operations by means of hardware acceleration, and provides mechanisms that enable precise and deterministic scheduling. The layers above the MAC are executed on a host computer connected through the Host Interface, or can be integrated in the embedded processor.

The shared-memory is used to accelerate the transfer of packets among SW MAC, HW MAC, and Host-PC. The HW MAC writes the packets received from the PHY into the shared-memory using Direct Memory Access (DMA), and reads the ones to transmit. Similarly, the Host-PC writes the packets to be transmitted into the shared-memory, and reads the ones received from the PHY. Packets are referenced by packet-pointers that identify their position in the shared-memory. The SW MAC is responsible for allocating packet-pointers and uses the OpenMAC interface to exchange packet-pointers with the HW MAC and with the Host-PC.

Each packet is comprised of three elements: (i) the MAC header contains MAC control information, (ii) the payload contains the useful data of the packet, and (iii) the packet-descriptors contain information used to configure and monitor the HW MAC and the PHY. The SW MAC can access directly the shared-memory to write and read the packet-descriptors. In addition, since accesses from the embedded processor to large data-structures may introduce huge and non-deterministic software latencies, the SW MAC is prevented from reading and writing payloads. Instead, the SW MAC has only to access to the MAC headers and packet-descriptors to make simple MAC decisions (e.g., check MAC address, identify packet types, etc).

In the transmission process, the SW MAC sends its packet-pointer to the HW MAC. The packet is then read from shared-memory by the HW MAC and parsed to extract the packet-descriptors. In the reception process, the packet-descriptors of the received packet are first extracted and registered in the HW MAC, and then the packet is stored into the shared-memory. Finally, the HW MAC provides the packet-pointer to the SW MAC to notify it of the new packet that has been stored. The HW MAC buffers the packet-pointers that are used in reception. In the case that no packet-pointers are available, the HW MAC block notifies the SW MAC, which then provides new available packet-pointers.

The HW MAC includes two packet processing paths for time-consuming MAC operations, e.g., Cyclic Redundancy Code (CRC). Packets are read from the shared-memory, processed, and transferred to the MAC/PHY interface, and inversely with packets received from PHY. The HW MAC configures the PHY (e.g., modulation, power, carrier selection,
while(1){
    status = check_get_packet();
    if (((status & RxPP_request) && buffer_ready) {
        if (waiting_ACK)
            pointer_allocation(&buffer_PP_a);
        else pointer_allocation(&buffer_PP_b);
        buffer_ready = 0;
    }
    if (status & RxPackets_ready) {
        RxPP = get_packet();
        buffer_ready = 1;
        SrcAddr = RxPP->SrcAddr;
        DstAddr = RxPP->DstAddr;
        if (SrcAddr==SrcNode && DstAddr==DstNode)
            if (RxPP->Type== DATA) {
                LastDataRxPP = RxPP;
                waiting_ACK = 1;
            }
        else
            if (SrcAddr==DstNode && DstAddr==SrcNode) {
                if (RxPP->Type==ACK) waiting_ACK = 0;
                else if (RxPP->Type==CFC) {
                    send_packet(LastDataRxPP);
                    waiting_ACK = 1;
                }
            }
    }
}

Figure 3. Example of the SW MAC code of the relay.

etc.), at any time when the SW MAC sends new parameters through the OpenMAC interface, or on a packet by packet basis by writing the PHY packet-descriptors through the MAC/PHY interface at the beginning of each transmission.

B. MAC Layer Prototyping Process

The prototyping process of a MAC protocol using Open-MAC is carried out by implementing a non hardware-specific C/C++ program (SW MAC) on the embedded processor without any operating system. The use of packet-descriptors and a simple Hardware Abstraction Layer (HAL) hide hardware details from the programming model. The HAL is comprised of a basic set of C functions that ease the access of the SW MAC to the OpenMAC interface.

Figure 3 shows an example of the SW MAC source code used to implement the basic relay node functionality. The relay checks the status of packet reception continuously. If a new packet is received, it verifies the source and destination addresses, and the packet type. The last data packet received from the source node is buffered, and it is re-transmitted when a CFC packet is received from the destination node.

C. OpenMAC-based Testbed

The experimental performance evaluation has been carried out using a real network composed of three OpenMAC-based nodes: source, relay, and destination. Each node is implemented with a Commercial Off-The-Shelf (COTS) FPGA board (Xilinx Virtex-5 FX100/FX70) with a PowerPC processor embedded in the FPGA. The SW MAC runs on the PowerPC and the rest of the OpenMAC architecture is implemented in the FPGA. The SW MAC source code is compiled using the Xilinx Software Development Kit (SDK) to get an executable for the PowerPC. The FPGA is programmed from the Xilinx SDK using a configuration file which integrates the OpenMAC architecture.

Since measuring instrumentation for radio channels was not yet in place at the time of writing, the OpenMAC-based nodes are connected over Ethernet. We have artificially introduced channel noise in order to be able to validate the testbed against known references. For this reason, and in order to provide readers with reproducible results, we report the achieved figures against packet error probabilities rather than against signal to noise ratios in each node-to-node channel.

The source node is connected to a computer over Ethernet. The computer enables the configuration of the experiment and the monitoring of several performance results as described in the next section.

D. Experiment Monitor

The Experiment Monitor software runs on a computer connected to the OpenMAC-based network over Ethernet. It includes a graphical user interface (GUI) that enables to:

1) Configure the experiment by programming the following parameters: (i) the length of the data packets, (ii) selection of ARQ or C-ARQ, (iii) the packet generation rate at the source node, (iv) the total time of the experiment or total number of data packets generated by the source node, and (v) the power consumption level associated to transmission, reception, and idle modes.

2) Monitor the experiment by displaying the following performance metrics in real-time: (i) delay (i.e., average time to deliver a data packet to the destination), (ii) throughput (i.e., ratio between the number of data bits delivered successfully to the destination and the total time elapsed in the experiment), and (iii) energy efficiency (i.e., ratio between the number of data bits delivered successfully to the destination and the energy consumed by the network nodes).

The time measurements in the different operation modes (i.e., transmission, reception, and idle listening) are performed directly by each OpenMAC node, whereas the energy efficiency is computed by the Experiment Monitor software using the time measurements and the power consumption parameters provided by the user.

V. PERFORMANCE EVALUATION

In this section, we compare the performance results of non-cooperative ARQ and the C-ARQ scheme described in Section III.

<table>
<thead>
<tr>
<th>System Parameters</th>
<th>Value</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data-rate</td>
<td>10 Mbps</td>
<td>Data-packet</td>
</tr>
<tr>
<td>ACK packet</td>
<td>60 bytes</td>
<td>CFC packet</td>
</tr>
<tr>
<td>ρtx</td>
<td>900 mW</td>
<td>ρrx = ρσ</td>
</tr>
</tbody>
</table>

Table 1
A. Scenario and System Parameters

The parameters used to execute the performance evaluation are summarized in Table I. Data and control packets are transmitted at a constant data-rate. The power consumption levels have been selected from the specification of the RN-131 low power Wi-Fi device. We have used 10,000 transmitted data packets and averaged the results for each test case.

In the next sections, we compare the performance of C-ARQ with that of non-cooperative ARQ in terms of delay, throughput, and energy efficiency.

B. Results

The average delay is represented in Figure 4 as a function of the packet error probability (PE) between source and destination nodes (PESD). The PE between relay and destination (PERD) has been fixed to a low value (PERD=0.1). As it can be expected, results show that the delay increases exponentially using ARQ. However, using C-ARQ it increases linearly at a very low rate. The delay achieved employing C-ARQ is significantly lower than the one achieved employing ARQ because the number of required retransmissions of data packets is reduced. C-ARQ yields a reduction up to a 75% with respect to ARQ when PESD=0.9.

Figure 5 shows the average delay as a function of PERD. PESD has been fixed to high value (PESD=0.7). As it can be expected, the delay achieved employing ARQ keeps constant for any PERD because the relay to destination channel is not used in ARQ. Results show that the delay increases exponentially when C-ARQ is employed, and it is higher than the one achieved with ARQ when PERD>PESD. Indeed, this happens because of the overhead introduced by a large number of required retransmissions of CFC packets. However, it can be observed that the delay achieved using C-ARQ is significantly lower than the one achieved employing ARQ when PERD<PESD, and it yields a reduction up to a 45% with respect to ARQ when PESD is very high (PESD = 0.7) and PERD<0.2.

The throughput is represented in Figure 6 as a function of PESD. PERD has been fixed to low value (PERD=0.1). Results show that the throughput decreases exponentially when employing ARQ. However, it decreases linearly at a very low rate when using C-ARQ. The throughput achieved using C-ARQ is significantly higher than the one achieved using ARQ, and it yields an increase of 150% of the throughput achieved using ARQ when PESD=0.9.
and PERD with respect to ARQ when PESD is very high (PESD=0.7) is lower than in ARQ. C-ARQ yields an increase up to a 30% when PERD is greater than PESD, the throughput of C-ARQ using ARQ when PERD is lower than PESD=0.7. However, when PERD is greater than PESD, the energy efficiency of C-ARQ is lower than in ARQ. C-ARQ yields an increase up to a 35% with respect to ARQ when PESD is very high (PESD=0.7) and PERD<0.2.

The energy efficiency is represented in Figure 8 as a function of PERD. PERD has been fixed to a low value (PERD=0.1). As it can be expected, the throughput achieved using ARQ keeps constant for any PERD. It can be observed that the throughput decreases exponentially when using C-ARQ. The throughput achieved using C-ARQ is significantly higher than the one achieved using ARQ when PERD is lower than PESD=0.7. However, when PERD is greater than PESD, the throughput of C-ARQ is lower than in ARQ. C-ARQ yields an increase up to a 30% with respect to ARQ when PESD is very high (PESD=0.7) and PERD<0.2.

The energy efficiency is represented in Figure 8 as a function of PERD. PERD has been fixed to a low value (PERD=0.1). As it can be expected, the energy efficiency using ARQ keeps constant for any PERD. It can be observed that energy efficiency decreases exponentially using C-ARQ. The energy efficiency using C-ARQ is significantly higher than in ARQ when PERD is lower than PESD=0.7. However, when PERD is greater than PESD, the energy efficiency of C-ARQ is lower than in ARQ. C-ARQ yields an increase up to a 35% with respect to ARQ when PESD is very high (PESD=0.7) and PERD<0.2.

VI. CONCLUSION

This paper presents an experimental comparison of C-ARQ versus ARQ protocols implemented in a testbed based on OpenMAC. Results show that C-ARQ outperforms non-cooperative ARQ, in terms of delay, throughput, and energy efficiency, when the conditions of the channel between relay and destination are better than source to destination channel conditions. This work demonstrates also that OpenMAC can provide abstraction and flexibility for PHY-MAC cross-layer design while effectively operating under field testing and sustaining real-time execution of device-independent software prototypes for novel MAC protocols. Future work is aimed at interfacing a custom wireless PHY layer to OpenMAC.

REFERENCES