A Flexible Framework for Wireless Medium Access Protocols

Invited Paper

Chris Hunter, Joseph Camp, Patrick Murphy and Ashutosh Sabharwal
Rice University
Department of Electrical and Computer Engineering
6100 Main St., Houston, TX 77005, USA
Email: {chunter, camp, murphpo, ashu}@rice.edu

Chris Dick
Xilinx Inc.
San Jose, CA 95124
Email: chris.dick@xilinx.com

***Abstract*— In this paper, we present a framework for Medium Access Control (MAC) protocol development and performance evaluation. The framework, developed for the Rice University Wireless Open-Access Research Platform (WARP), allows us to interface a large class of medium access protocols with custom physical layer (PHY) implementations, thereby providing a flexible and high-performance research tool. MAC protocols for our framework are written in C and targeted to embedded PowerPC cores within the Xilinx Virtex II-Pro class of FPGAs. A key innovation is a flexible interface between the PHY and the MAC capable of exposing user-defined parameters to either layer, thus enabling cross-layer research.**

I. Introduction

Commercial IEEE 802.11 wireless hardware has dominated the wireless networking market in recent years. Despite steady increases in the raw physical layer (PHY) throughput of these devices, the Medium Access Control (MAC) standard has seen fewer radical changes. As of 2006, the draft standard of 802.11n specifies a 540Mbps physical layer and an effective throughput of only 200Mbps at the MAC layer [1].

Novel algorithms to combat this disparity have been proposed in the literature. These algorithms include the Opportunistic Auto Rate (OAR) algorithm to adaptively throttle transmission rates based on channel quality, and algorithms to address throughput starvation in multihop networks [2], [3]. Demonstrations of the gains of these new MAC algorithms have existed primarily in simulation environments such as NS-2 and GloMoSim [4], [5]. Commercial wireless hardware is generally proprietary and lacks low-level programmability. To enable at-scale testing, a development platform is essential to demonstrate the viability of novel MAC protocols and to enable exploitation of PHY layer characteristics.

A number of research projects have partially addressed this need by overwriting behavior of commercial 802.11 chipsets [6], [7], [8]. However, these designs were intended to provide enhancements to existing 802.11 MAC implementations rather than to provide a general MAC development environment. Enhancing these existing projects, the SoftMAC project allows users to modify the format of transmitted packets via custom drivers that exploit reverse-engineered details about the Atheros wireless chipset [9]. While this provides a development environment that generally spans the space of MAC algorithms possible on an 802.11 PHY, it offers limited functionality for research in the larger space of clean-slate MAC-PHY pairs.

In this paper, we present a framework designed for the Rice University Wireless Open-Access Research Platform (WARP) [10]. The framework, known as WARPMAC, provides high-level, low-breadth tools for derivatives of standard random access protocols, and low-level, high-breadth tools for ground-up designs (see Figure 1). This two-tiered structure allows for user-defined abstraction of hardware details and maximum flexibility for a large class of algorithms.

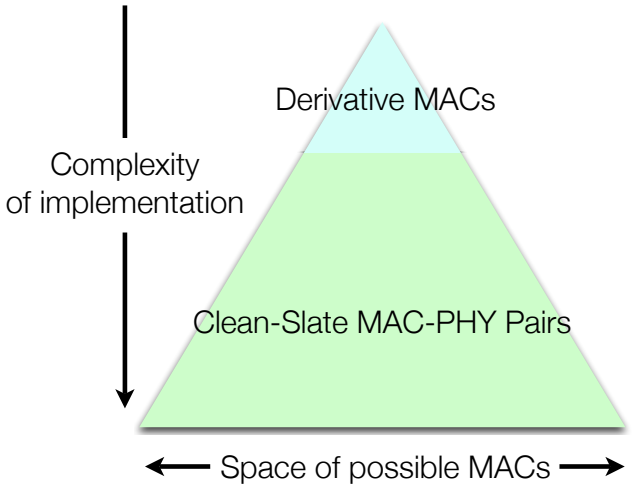

Fig. 1. *Two-tiered structure of WARPMAC*

However, the flexibility offered by the platform is not achieved at the cost of lower performance. The motivation for the creation of the platform is to demonstrate real-world gains based on novel MAC protocol concepts. This can only be achieved on a platform that is capable of operating in the time-scales of commercial wireless devices. To show how

our framework meets the needs of high performance and maximum flexibility, we will present results from our initial MAC implementation efforts on the platform.

II. WARP Hardware

The features provided by hardware serve as the foundation of what can be offered at higher networking layers. In this section, we discuss the WARP hardware capabilities. For more details see [11].

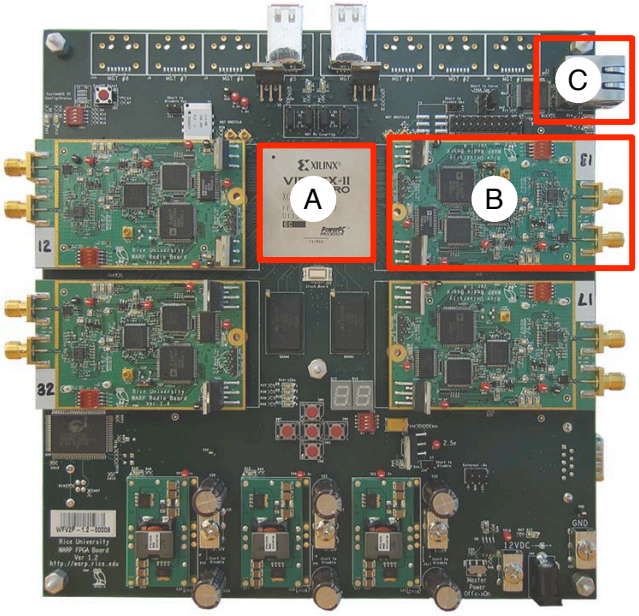

Fig. 2. *WARP FPGA board and radios*

The main hardware features of WARP boards shown in Figure 2 are:

A. *Xilinx Virtex-II Pro FPGA:*

The FPGA includes a large number of embedded multipliers and programmable logic blocks for real-time DSP applications as well as two PowerPC 405 cores (PPCs). When compiled for these cores, C-code can interact with peripherals implemented directly in the FPGA fabric. This system-level interaction is described in more detail in Section III.

B. *MIMO-Capable Radios:*

Each WARP node can be configured with up to four radios for MIMO applications. These radio boards, also designed at Rice University, are capable of targeting both the 2.4GHz and 5GHz ISM bands. Additionally, their Maxim MAX2829 radio chipset is intended for wideband applications such as OFDM.

C. *10/100 Ethernet:*

Ethernet is included on the WARP boards to serve multiple purposes. For example, it can be used to provide a traffic source and sink to test the implementation algorithms. Additionally, it can be used to offload statistics from an experiment, allowing for large-scale tests.

III. WARP System Architecture

In this section we will describe the system-level architecture of WARP from the point of view of network protocols. As stated earlier, programs compiled for the PPCs within the FPGA can communicate with hardware peripherals in the fabric. This interaction occurs via standard bus transactions and interrupts. MAC algorithms exist as compiled C programs that run on the PPC cores (see Figure 3).

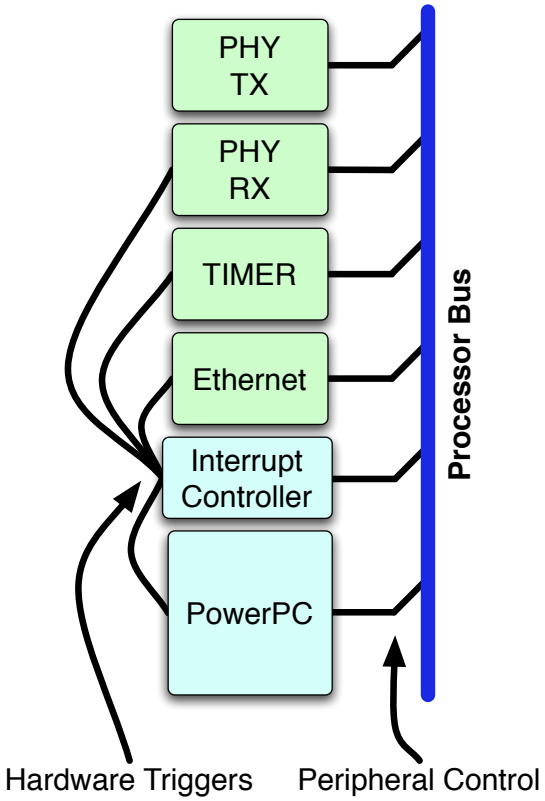

Fig. 3. *Example system within the Xilinx Virtex-II Pro FPGA*

Realizing a medium access protocol on a hardware platform intrinsically requires that the protocol be able to control, and be controlled by, hardware peripherals. Examples of peripherals include a wireless PHY transmitter, a wireless PHY receiver, an on-fabric timer, and an ethernet MAC to source and sink packets from the custom wireless MAC. Hardware interfaces exist in two forms: peripheral control via bus transactions and hardware triggers via interrupt handling.

1) Peripheral Control: Hardware peripherals are attached to either a simpler On-Chip Peripheral Bus (OPB) or a faster Processor Local Bus (PLB). Through memory reads and writes, software running on the PPC can pass values to custom cores and vice versa. Through this interface, a MAC can control a PHY and other pieces of hardware. Additionally, a MAC can read state from hardware peripherals, allowing for cross-layer optimizations that exploit tight coupling between the MAC and PHY.

2) Hardware Triggers: Often, the state of a MAC algorithm needs to be modified by some hardware event. For example, if a received packet needs to be processed, the PHY needs to notify the MAC that there is a packet in its buffer. This need is satisfied by the use of an interrupt controller attached to the necessary hardware peripherals. Users can register interrupt handlers to deal with each hardware trigger.

A. SISO-OFDM PHY

In a separate project at Rice University, we have undertaken the design of a custom OFDM physical layer similar in functionality to 802.11a. Currently, we have developed an operational PHY transceiver with flexible data rates starting from 12Mbps. Additionally, the transceiver has built-in cyclic redundancy checks (CRCs) for determining the presence of bit-errors, as well as carrier sense multiple-access (CSMA) abilities for enabling collision-avoidance MACs. A 2x2 full-multiplexing MIMO-OFDM extension is currently under development to achieve higher throughputs.

B. Open-Access

A fundamental attribute of WARP is the free distribution of hardware and software projects on the WARP website in the form of a central repository [10]. As community involvement with the platform grows, research groups will be able to leverage the engineering efforts of one another to build novel MACs and PHYs. The source-code for the MAC implementations presented in Section V is currently available in the repository.

IV. WARPMAC Framework

WARPMAC is a suite of software routines that sits above the physical layer and allows for flexible abstraction of hardware interactions. Starting with the physical layer, the WARPMAC structure is as follows (see Figure 4):

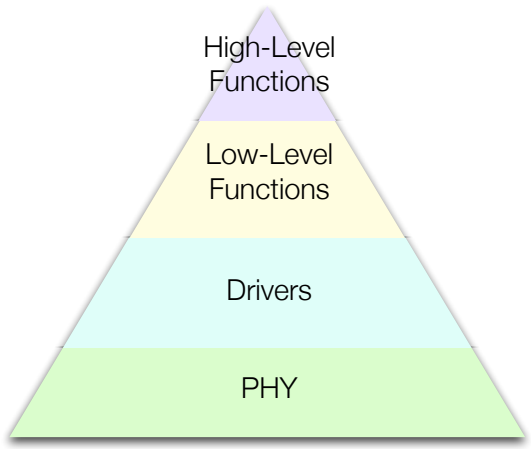

Fig. 4. *WARPMAC organization*

- **PHY:** Real-time dependent operations are implemented in the FPGA fabric. This includes the physical layer and all supporting hardware peripherals such as automatic gain control and packet detection. The custom physical layer can be Rice University's SISO-OFDM link, or any custom layer designed by the community and added to the central repository.
- **Drivers:** At this layer, software routines abstract register reads and writes to low-level hardware control functions. For example, the custom PHY driver has a function responsible for setting constellation order. Additionally, the radio controller has a function to set a center frequency and also has functions to switch between receive and transmit modes.
- **Low-Level Functions:** These functions provide an additional layer of abstraction by incorporating many driver calls into functions that read more like psuedocode. For example, a MAC designer can have access to a function that takes a packet as an input and returns when that packet has been sent over the air. Internally, the function places the radio into a transmit mode, loads the payload into the PHY, and starts the PHY processor. Another example of low-level functionality at this layer is interrupt handler registration. Users can attach functions to the framework such that custom routines will be called upon the reception of a packet that passed CRC, the reception of a packet that failed CRC, and the expiration of a timer. Additionally, timer control is handled on the clock-cycle level. Users can set a timer for a maximum number of clock cycles, and the corresponding handler will be called upon the passage of that amount of time.
- **High-Level Functions:** At the highest level, the WARPMAC framework provides functions to implement a large class of random access protocols. For example, timer control is handled on the behavioral level. Users can set timers for a timeout or backoff window. For the latter, the function internally manages the binary exponential backoff seen in all random-access MACs.

The choice of which layer of the WARPMAC organization to use is completely dependent on the requirements of a user's MAC algorithm. For example, because most random-access protocols require a binary exponential backoff to deal with medium contention, it is likely that the high-level function capable of such a behavior is portable to novel random-access MACs. However, for MAC algorithms substantially different from basic ALOHA, it is likely that other parts of the stack will be required [12]. All of the software for WARPMAC and the documented API are freely available in the WARP repository. Coinciding with the open-access philosophy of WARP, the functions provided in these layers will grow with increasing community involvement to encompass control of new physical layers.

V. Current MAC Implementations

We have designed and implemented two classical MAC protocols on WARP. The first, based on the ALOHA protocol, provides the foundation for most random-access MACs. The second algorithm is an enhancement of the first that provides CSMA capabilities.

A. ALOHA

The basic algorithm can be described simply in three steps:

1) If a node has a packet to send, it should send the packet.
2) If a node receives a data packet, it should acknowledge that packet by sending an ACK.
3) If no ACK is received by the transmitter of a data packet, it should try to send the packet again after a random backoff time.

In this way, the algorithm resolves packet collisions due to medium contention. This behavior translates naturally to a more code-ready state machine as shown in Figure 5.

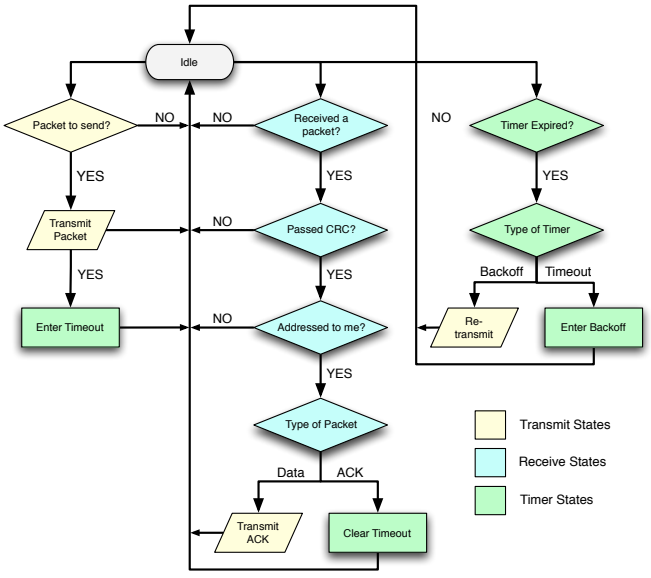

Fig. 5. *Subset of MAC behavior*

In this diagram, there are three primary code branches:

- **Transmit States:** These states are responsible for taking a packet from a higher layer of the networking stack and pushing it down to the PHY for transmission. The left-most branch in the flowchart describes this process. After successfully transmitting the packet, the algorithm must enter a timeout period where it will wait for an acknowledgment.
- **Receive States:** These states perform the inverse task. They extract a packet from the wireless medium via the PHY and pass it to higher layers. The middle branch of the flowchart describes the different checks that the MAC must perform on each received packet. If the received packet was a good data packet meant for the receiver, the node creates an ACK and sends it back to the original transmitter.
- **Timer States:** These states are responsible for managing inferred packet collisions. For ALOHA, there are two types of timers. The first, the timeout window, is a deterministic amount of time that the original transmitter is willing to wait for an ACK. If this time is exceeded without the successful reception of an ACK, the node enters into the second type of timer interval, known as a binary exponential backoff window. This backoff period is stochastic in order to ensure that all mutually contending nodes do not attempt to transmit packets simultaneously after sensing an idle period. When this timer expires, a retransmission of the original packet begins.

Important metrics include the round-trip time (RTT) and the turn-around time (TAT), especially given how these values expose inefficiencies in the interaction between the MAC and PHY (see Figure 6).

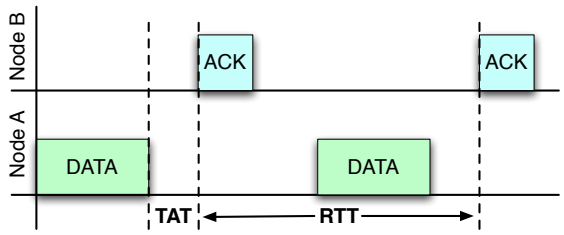

Fig. 6. *MAC performance metrics*

- TAT corresponds to the amount of time between the end of a data packet transmission and the start of an ACK transmission from the receiving node. This value is crucial to the performance of the MAC protocol because it corresponds to the minimum timeout time the transmitter is willing to wait for an ACK.
- RTT is a measure of the duration of time between transmissions in a fully backlogged MAC, which directly relates to the maximum theoretical throughput of the MAC. This rate assumes no bit-errors, and hence no packet drops. In other words, it captures the amount of MAC overhead seen by higher layers in the best-case scenario.

For our implementation of ALOHA on a 12Mbps PHY, measurements can be seen in Table I.

TABLE I
MAC-Layer Performance Measurements of the ALOHA Implementation driving a 12Mbps Physical Layer

Length(bytes)	TAT(μs)	RTT(ms)	Throughput(Mbps)
500	52	.696	5.74
750	52	.896	6.70
1000	52	1.108	7.22
1250	52	1.320	7.58
1500	52	1.532	7.84

These results demonstrate the intuitive fact that shorter packet lengths result in lower rates due to the increased overhead of a static-sized MAC header relative to a shorter payload. Additionally, the TAT remains constant with varying packet lengths since ACK generation is only dependent upon the static-sized MAC header. These throughput measurements are on par with commercial 802.11 systems [13]. While longer than commercial systems, these TAT measurements are on par with competing platforms [9]. We are currently working to reduce TAT to the level of commercial systems.

We performed an additional experiment to describe the effective rates seen at higher networking layers. For this test scenario, nodes were configured to source and sink packets from Ethernet. In other words, nodes acted as wireless Ethernet hubs, forwarding all Ethernet traffic over the custom wireless link. To test the application-layer throughput ("goodput") of the custom MAC, iperf was used to generate UDP streams of arbitrary packet size [14]. The results are shown in Table II. For comparison to the previous experiment, theoretical throughput results are also included.

TABLE II
APPLICATION-LAYER PERFORMANCE MEASUREMENTS OF THE ALOHA IMPLEMENTATION DRIVING A 12MBPS PHYSICAL LAYER

Length(bytes)	Throughput(Mbps)	UDP Goodput(Mbps)
500	5.74	4.57
750	6.70	5.71
1000	7.22	6.19
1250	7.58	6.63
1500	7.84	6.73

As expected, effective goodputs of the link are considerably slower than the theoretical maximum throughput due to packet drops from bit errors. This performance gap can be decreased with improved reliability in the physical layer by using techniques such as channel coding and MIMO, and by exploiting novel MACs to take advantage of channel information.

B. CSMA-Enabled ALOHA

As an extension to the previously discussed ALOHA implementation, support of carrier sensing was added to improve the performance of the link under heavy contention scenarios. This protocol adds additional software states to the MAC in order to check the medium before beginning a transmission in order to avoid collisions. To test the implementation of the algorithm, iperf was used to saturate traffic from one node to another and vice versa. Results from this heavy contention experiment are shown in Figure 7.

As expected, carrier sensing dramatically outperforms standard ALOHA in heavy contention scenarios. CSMA is the first step in PHY-level awareness in medium-access. In the future we plan to build new MACs with even closer MAC-PHY coupling into this existing system.

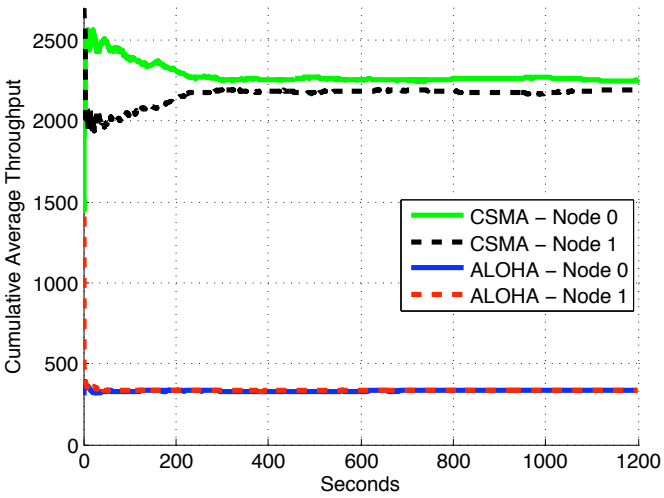

Fig. 7. *Performance of ALOHA versus CSMA*

VI. CONCLUSIONS

We have created a highly flexible and performance-driven medium access development framework. Its two-tiered structure provides high-level functionality for rapid development of derivative MACs, and low-level support for clean-slate designs. The framework has been used at Rice University to implement classical medium access protocols, and is currently being used to investigate novel algorithms.

VII. ACKNOWLEDGMENTS

Special thanks to Xilinx and Maxim. This work was partially supported by NSF grants CNS-0551692 and CNS-0619797.

REFERENCES

[1] *http://www.ieee802.org/11/*.
[2] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, "Opportunistic Media Access for Multirate Ad Hoc Networks," in *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, 2002, pp. 24–35.
[3] V. Gambiroza, B. Sadeghi, and E. Knightly, "End-to-End Performance and Fairness in Multihop Wireless Backhaul Networks," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, 2004, pp. 297–301.
[4] *http://www.isi.edu/nsnam/ns/*.
[5] *http://pcl.cs.ucla.edu/projects/glomosim/*.
[6] B. Raman and K. Chebrolu, "Revisiting MAC Design for an 802.11-based Mesh Network," Third Workshop on Hot Topics in Networks (HotNets-III), Nov 2004.
[7] *http://madwifi.org/*.
[8] *http://pdos.csail.mit.edu/roofnet/doku.php*.
[9] M. Neufeld, J. Fifield, C. Doerr, A. Sheth, and D. Grunwald, "SoftMAC - Flexible Wireless Research Platform," Fourth Workshop on Hot Topics in Networks (HotNets-IV), Nov. 2005.
[10] *http://warp.rice.edu*.
[11] P. Murphy, A. Sabharwal, and B. Aazhang, "Design of WARP: A Wireless Open-Access Research Platform," in *European Signal Processing Conference*, Florence, Italy, Sept. 2006, Accepted.
[12] N. Abramson, "The ALOHA System - Another Alternative for Computer Communications," in *Proceedings of the Fall Joint Computer Conference*, 1970, pp. 281–285.
[13] *http://standards.ieee.org/*.
[14] *http://dast.nlanr.net/Projects/Iperf/*.