WiMAC: Rapid Implementation Platform for User Definable MAC Protocols Through Separation

Simon Yau[†], Liang Ge[†], Ping-Chun Hsieh[†], I-Hong Hou[†], Shuguang Cui[†], P. R. Kumar[†], Amal Ekbal[*], and Nikhil Kundargi[*]
[†]Texas A&M University
[†]{symoyau, geliang87, lleyfede, ihou, cui, prk}@tamu.edu
[*]National Instruments
[*]{amal.ekbal, nikhil.kundargi}@ni.com

ABSTRACT

This demo presents WiMAC, a general-purpose wireless testbed for researchers to quickly prototype a wide variety of real-time MAC protocols for wireless networks. As the interface between the link layer and the physical layer, MAC protocols are often tightly coupled with the underlying physical layer, and need to have extremely small latencies. Implementing a new MAC requires a long time. In fact, very few MACs have ever been implemented, even though dozens of new MAC protocols have been proposed. To enable quick prototyping, we employ the mechanism vs. policy separation to decompose the functionality in the MAC layer and the PHY layer. Built on the separation framework, WiMAC achieves the independence of the software from the hardware, offering a high degree of function reuse and design flexibility. Hence, our platform not only supports easy cross-layer design but also allows protocol changes on the fly. Following the 802.11-like reference design, we demonstrate that deploying a new MAC protocol is quick and simple on the proposed platform through the implementation of the CSMA/CA and CHAIN protocols.

CCS Concepts

•**Networks** → **Network experimentation;**

Keywords

Wireless testbed; MAC; software-defined radio

1. INTRODUCTION

Driven by the emerging applications such as multimedia streaming and VoIP, a wide variety of wireless protocols have been proposed to meet the increasingly stringent requirements for wireless networks, such as quality of service (QoS) and power saving. For example, the IEEE 802.11 standard, which is one of the most widely used wireless protocols, has undergone several major enhancements in each layer over the past decade. Moreover, due to the substantial rate improvement in the physical layer, much effort has been dedicated to design new protocols in the Medium Access Control (MAC) layer to enhance the original CSMA/CA mechanism. Despite the abundance of the theoretical studies on wireless MAC protocols, very few of them make it beyond simulation towards real implementation with transmissions over the air. The main reason is that the MAC layer is often tightly coupled with the PHY layer, with the result that MAC developers are often required to implement not only the MAC functionality but also the features in the physical layer, which lie outside their main expertise. Therefore, new protocols often need to be reconfigured from the very ground up, leading to long development time.

To reduce the prototyping time of new protocols, one promising alternative is to make the MAC layer programmable by adopting the software-defined radio (SDR). Conventionally, SDR implements the time-critical functionality in the programmable hardware domain, either using FPGAs or MCUs, while it handles time-insensitive signal processing on a computer host. Based on this split architecture between software and hardware, various techniques have been presented in [3] to allow flexible MAC customization directly from the host while maintaining the real-time performance. However, the exact criterion for splitting the MAC functions is still not clear. To enable efficient modular design in wireless testbeds, Ansari *et al.* [1] studied the commonality of popular MAC protocols to define a class of elementary MAC blocks. Despite such a flexible framework, the platform is mainly hardware-based and thus the programmability outside the proposed class of elementary blocks might not be guaranteed. Tinnirello *et al.* [4] also present a MAC processor based on the runtime composition of several IEEE 802.11 standards to achieve flexibility in the MAC layer. Nevertheless, the processor is implemented on the target commodity hardware, which offers less freedom than required by a general-purpose MAC implementation.

Different from the prior efforts, WiMAC is the first general-purpose wireless testbed that supports multi-user medium access control with tested real-world implementation of both CSMA and non-CSMA protocols. To provide easy configuration for the wireless MAC protocols, WiMAC is built on the philosophy of *mechanism vs. policy separation* to achieve higher programmability. The "mechanism" refers to the category of low-level operations that handle the real packet transmissions over the network, while the "policy" indicates the high-level schemes for channel contention and packet scheduling. By describing the policies in a high-level language, the mechanisms can be easily adapted to different policies. For example, to avoid possible collisions during transmissions, a random backoff algorithm has been implemented in the CSMA/CA protocol. No matter how the contention window is chosen at each time, a backoff timer is required for any possible backoff algorithms. Therefore, a backoff timer is considered as a mechanism, while determining the size of the contention window is categorized as a policy. Hence, the effort needed to develop new

SIGCOMM '15 August 17-21, 2015, London, United Kingdom

ACM ISBN 978-1-4503-3542-3/15/08.

DOI: http://dx.doi.org/10.1145/2785956.2790031

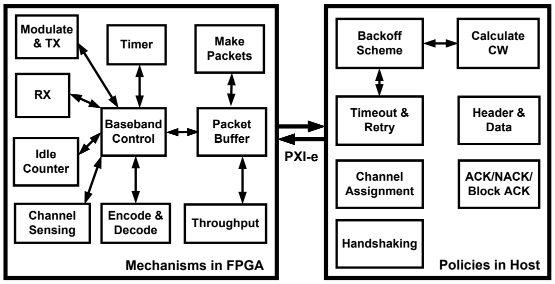

Figure 1: Mechanism vs. policy separation in WiMAC.

MAC protocols is greatly reduced. Figure 1 shows a more detailed illustration of the separation framework.

We now summarize the important features of WiMAC.

- **Independence of software from hardware.** In WiMAC, the software-defined policies are decoupled from the hardware functionality based on the separation framework. Therefore, developers are not required to redesign the hardware from scratch since the functional blocks can be reused. The independence of software from hardware minimizes the deployment overhead for emerging protocols.
- **Enabling protocol changes on-the-fly.** Given the programmability from the computer host, WiMAC provides a user-friendly GUI and allows the users to switch between different MAC protocols while the testbed is in operation.
- **Supporting cross-layer design.** The MAC layer, which serves as an interface between the link layer and the physical layer, is often tightly coupled with the underlying physical layer. Built on a dedicated FPGA, WiMAC provides enough freedom for developers to reconfigure the physical layer, enabling protocol design across layers.
- **Quick prototyping.** Based on the above three features, WiMAC lowers the barrier that currently blocks novel ideas from being implemented, helping researchers implement new MAC protocols with short development time.

To the best of our knowledge, there is no existing wireless testbed that exhibits all the features described above. We believe that WiMAC is a promising platform for the networking research community.

As a design example to illustrate the proposed wireless testbed, we use an 802.11-like reference design to implement the CSMA/CA and the CHAIN [6] protocols. A demo video of WiMAC is available on YouTube[1]. The illustratory implementation is only meant to serve as a starting point to explore the common mechanisms used in most MAC protocols. Based on the current deployment, our short-term goal is to bring a broader class of the real-time protocols, such as the debt-based policies for real-time wireless [2], and the energy-saving protocol SOFA [5], to the implementation stage.

2. WIRELESS MAC IMPLEMENTATION

First, we briefly describe the 802.11-like reference design in the proposed testbed. We mainly employ the platform supported by National Instruments, which includes a PXI Chassis and several USRPs as wireless nodes. Instead of using commercial wireless PHY cards, we build the mechanisms in the FPGAs to enable more flexible protocol deployment since we can thereby make the PHY management also available to users if necessary. The design primarily uses the 2.4GHz ISM band with a channel bandwidth of either 20MHz or 40MHz. In terms of the coding rate, WiMAC supports the options of BPSK, QPSK, 16-QAM, and 64-QAM. In the present design, we use fixed-size packets with the standard ACK for all packets.

2.1 CSMA/CA

As a starting point, we illustrate the separation framework by implementing the popular CSMA/CA protocol. On the software side, the computer host handles the programmable policies, such as ACK control, packet header, and contention window calculation, etc. In the FPGA domain, we have built the core mechanisms that are fundamental to most MAC protocols. We have studied a list of several proposed MAC protocols to determine the common functional blocks. We have thereby identified and implemented a group of features in the FPGA, such as configurable slot times, configurable inter-frame time, configurable congestion window algorithm, backoff timer, and channel sensing, etc. To handle time-critical mechanisms, instead of using a state-diagram structure, we use flags or simply preload values to the FPGA to update information packet by packet. These features serve as the foundation for developing any new MAC protocol in the testbed.

2.2 CHAIN

CHAIN [6] is a MAC protocol for enhancing the uplink efficiency. Consider a wireless network with one AP and several clients. Conventional CSMA uses random exponential backoff to handle contention in medium access. However, as the network becomes more congested, collisions happen so frequently that the network efficiency becomes extremely low. While being compatible with the existing CSMA, CHAIN employs a sequential *piggyback transmission* scheme where a client that has obtained initial access triggers a *chain* of transmissions in a predetermined order when the traffic is heavy. The precedence is assigned and maintained in the AP, and followed by the clients. Based on the CSMA configuration, CHAIN can be implemented by reusing the core mechanisms in the FPGA and adding a few new functional blocks in the FPGA, such as ACK overhearing and a debt system for fairness. The complexity of implementing CHAIN is greatly reduced due to the decomposition strategy.

3. ACKNOWLEDGEMENT

This material is based upon work partially supported by AFOSR Contract FA9550-13-1-0008, NSF under Contract Nos. CNS-1302182, Science Technology Center Grant CCF-0939370, CNS-1343155, ECCS-1305979, CNS-1265227, DoD with grant HDTRA1-13-1-0029, and by grant NSFC-61328102.

4. REFERENCES

[1] J. Ansari, X. Zhang, A. Achtzehn, M. Petrova, and P. Mahonen. Decomposable MAC framework for highly flexible and adaptable MAC realizations. In *Proc. IEEE Symposium on New Frontiers in Dynamic Spectrum*, pages 1–2, 2010.

[2] I.-H. Hou and P. R. Kumar. Utility maximization for delay constrained QoS in wireless. In *Proc. IEEE INFOCOM*, pages 1–9, 2010.

[3] G. Nychis, T. Hottelier, Z. Yang, S. Seshan, and P. Steenkiste. Enabling MAC protocol implementations on software-defined radios. In *Proc. 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 91–105, 2009.

[4] I. Tinnirello, G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, and F. Gringoli. Wireless MAC processors: Programming MAC protocols on commodity hardware. In *Proc. IEEE INFOCOM*, pages 1269–1277, 2012.

[5] Z. Zeng, Y. Gao, and P. R. Kumar. SOFA: A sleep-optimal fair-attention scheduler for the power-saving mode of WLANs. In *Proc. International Conference on Distributed Computing Systems (ICDCS)*, pages 87–98, 2011.

[6] Z. Zeng, Y. Gao, K. Tan, and P. R. Kumar. CHAIN: Introducing minimum controlled coordination into random access MAC. In *Proc. IEEE INFOCOM*, pages 2669–2677, 2011.

[1] A demo video is available: http://youtu.be/yj7YEWjx1HM