Ad Hoc Networks 17 (2014) 37–59

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

snapMac: A generic MAC/PHY architecture enabling flexible MAC design ☆

Pieter De Mil *, Bart Jooris, Lieven Tytgat, Jeroen Hoebeke, Ingrid Moerman [1], Piet Demeester

Department of Information Technology, IBCN Research Group, Ghent University – iMinds, G. Crommenlaan 8 Box 201, 9050 Gent, Belgium

ARTICLE INFO

Article history:
Received 20 April 2013
Received in revised form 15 November 2013
Accepted 12 January 2014
Available online 22 January 2014

Keywords:
Radio hardware abstraction
MAC/PHY architecture
Reconfigurable
MAC
Flexible
Time accurate radio driver

ABSTRACT

Timing is a key issue in many wireless, lower-layer (e.g., physical and data link layer) communication protocols. Maintaining time-critical behavior while increasing MAC protocol complexity is the challenge for many MAC implementations. To comply with stringent time constraints, current MAC implementations typically require such a tight integration to the radio driver that they become one monolithic block of code with MAC-specific logic hard coded at the lowest firmware level. Execution of time-critical functions in the firmware is a good strategy, but results in limited flexibility for MAC designers because the radio driver is dedicated for specific MAC protocol logic. We propose "snapMac": a generic MAC/PHY architecture with a clean separation between the MAC protocol logic at the user level and the execution at the radio firmware level (Patent Pending). Our generic programming interface enables more flexibility, an easy way to compose new MAC designs, and getting feedback from the radio capabilities We demonstrate the feasibility and performance of this architecture by implementing it on a resource-constrained wireless sensor node. The experimental evaluation shows, for example, that we can simultaneously keep the flexibility of a software ACK and meet the ACK timing constraints as specified in the 802.15.4 standard. We also achieve 97% (i.e., 218 kbit/s) of the theoretical 802.15.4 throughput. This new implementation approach for MAC/PHY interactions has potential to be applied in other domains (e.g., WiFi, software defined radio, cognitive radio, etc.). Demonstrating the portability of snapMac is future work. "snapMac" enables the design and execution of new MAC protocols in a *snap*.

1. Introduction

1.1. Situation and problem statement

More and more devices are being connected to the Internet (Internet of Things (IoT) vision [1]), enabling a wide range of applications having varying requirements. Libelium [2] – a Wireless Sensor Network (WSN) platform provider – lists more than 50 IoT applications, grouped in different domains such as Smart Cities, Smart Metering, Security & Emergencies, Retail, Logistics, Home Automation, etc. Communication is mostly wireless, although the underlying PHY technology can vary (e.g., IEEE 802.15.4, Bluetooth, WiFi). The PHY hardware sets the upper bounds on the data rate (e.g., 250 kbit/s for the 2.4 GHz 802.15.4 PHY) but it are the radio driver and the Medium Access Control (MAC) layer who play an important role to meet the dispersed set of application requirements [3].

Because of these diverse applications requirements, lots of MAC protocols [4] were proposed over the last decades. Even in the IEEE 802.15.4-2006 [5] standard, multiple options are available: beacon-enabled or not, (un) slotted

☆ Patent Pending

* Corresponding author. Tel.: +32 479445858.
E-mail address: pieter.demil@snapTonic.net (P. De Mil).
URL: http://ibcn.intec.ugent.be (P. De Mil).
[1] Principal corresponding author.

1570-8705/$ - see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.adhoc.2014.01.004

CSMA-CA or Guaranteed Time Slots. Also in the 802.15.4e [6] standard multiple MAC techniques (time slotted channel hopping, asynchronous multi-channel adaptation, low latency deterministic networks, etc.) and frame formats (multi-superframe, slotframe, etc.) are specified to support a wide range of industrial and commercial applications. As a result, a MAC designer needs to implement a complex algorithm (with stringent time constraints) that can execute radio actions in a time accurate manner.

A MAC designer basically wants the freedom to compose any MAC design, with time-accurate execution and without the need for changing the radio driver firmware. Conversely, a firmware engineer basically wants to implement a performant driver, with a generic interface and without the need for making it MAC protocol specific. With *snapMac* we target both MAC designers and firmware engineers. In [7], an important requirement of the future data link layer is given in the definition: *"In order for IoT systems to achieve full interoperability, as well as the support of heterogeneous technologies (...), this data link layer must allow for diversity"*. Diversity means being able to design completely new MAC protocols, but also being able to change the MAC protocol at run-time in order to achieve interoperability with other devices. Ideally, the driver firmware is MAC agnostic and a generic interface is presented to a MAC designer. This will enable both reusability and portability.

1.2. Contributions

We have been working on WSN solutions since 2005 and experienced that we had to design various MAC protocols. In 2012, we published our pluralisMAC [8] framework. This allows for flexible switching between MAC protocols (so-called maclets). These maclets use shared functions for controlling the radio. We have designed the first version of pluralisMAC on top of a traditional radio driver (TinyOS CC2420 driver). This meant that most of the MAC functions were tied to the driver, and time-accurate execution was not possible. In a national project, iMinds and RMoni have designed a new sensor platform (RM-090,[2] using the CC2520 radio chip). There was no CC2520 driver available, so we took this opportunity to design it ourselves, keeping in mind the lessons we had learned. We addressed this challenging problem by designing snapMac: a generic MAC/PHY framework for flexible MAC design. Such a framework must allow to achieve the required time accuracy but with a clean separation between MAC and radio driver, hereby increasing flexibility, adaptability and portability at both levels. To proof this, we have implemented the proposed architecture on a resource-constrained sensor node and evaluated key metrics like ACK timing, throughput, round trip time and energy consumption. As such the key contributions of this article are the following:

- We present a novel generic MAC/PHY architecture, implemented on a resource-constrained device.
- We describe in detail how this architecture works.
- We show that MAC design is more flexible, easier and more time-accurate by using our *chain of commands* technique.
- We experimentally validate the implementation of the snapMac architecture both in terms of functionality and performance regarding all essential individual mechanisms and a Low Power Listening MAC protocol.
- We describe the portability and reusability options of snapMac, which are not demonstrated in this paper.

1.3. Paper structure

The remainder of the paper is organized as follows. Section 2 presents the related work. In Section 3, we describe the goals that enable MAC protocols to use a generic programming interface, portable across multiple heterogeneous platforms without sacrificing time accurate execution of the protocol logic. In Section 4 we propose our snapMAC concept. In Section 5 we show how to use snapMAC from a MAC layer point of view. We demonstrate the operation in an asynchronous multi-channel receiver-based communication scenario. Section 6 presents the performance evaluation of our snapMAC implementation. Section 7 discusses the portability of snapMac (because porting to other platforms is work in progress). Finally, Section 8 states our conclusions.

2. Related work

In this section, we highlight the bottlenecks on resource-constrained platforms, the related work in the WSN domain, and the related work in non-WSN domains.

2.1. Bottlenecks on resource-constrained platforms

We have identified the bottlenecks on a resource-constrained platform. A WSN platform is relatively constrained in term of resources (e.g., a class 1 node has 10 kB data size and 100 kB code size [9]). The processing power should be kept to the minimum to support the application and to guarantee the maximum node lifetime. The design of a generic MAC/PHY architecture should therefore cope with the resource scarcity, while still allowing sufficient flexibility.

The radio chip of a typical sensor node (or Network Interface Card), depicted in Fig. 1, is responsible for transmitting or receiving data that is generated or consumed on the processing unit (processor on host PC, microcontroller on a stand-alone sensor node). This data needs to be transferred to the radio unit via a communication bus (e.g., SPI). When multiple peripherals are connected to a *shared* communication bus, unpredictable access timings and/or packet transfer rate are often unavoidable. On the TelosB (TMote Sky) platform [10], the CC2420 radio chip and the flash storage unit share the SPI bus with the processing unit. An SPI bus arbiter is needed to grant access to a specific slave unit. Hence, it might take some time to get access to the bus when a transfer to/from the flash storage device is active. The same is valid for the Zolertia Z1 platform [11], while on the waspmote platform [12] the

[2] RM-090 info: http://www.rmoni.com/en/products/hardware/rm090.