anyMAC: adapting the Sun SPOT architecture for MAC development

anyMAC: adapting the Sun SPOT architecture for MAC development

Daniel van den Akker[1], Bart Braem and Chris Blondia
{Daniel.vandenAkker, Bart.Braem, Chris.Blondia}@ua.ac.be
Contact: Daniel van den Akker

PATS Research group, Dept. of Mathematics and Computer Science,
University of Antwerp - IBBT,
Middelheimlaan 1, B-2020, Antwerp, Belgium

The Sun SPOT platform allows sensor network nodes to be programmed in Java and is therefore well suited for prototyping new sensor network protocols. The design of the Sun SPOT network stack however, makes it hard to replace the default IEEE 802.15.4 based MAC implementation. We therefore introduce the 'anyMAC project': a number of alterations to the default Sun SPOT network stack. The resulting 'anyMAC network stack' allows MAC implementations to be easily interchanged and provides the flexibility needed to allow new MAC implementations to be developed, as demonstrated by a proof of concept 'DemoMAC' MAC implementation. Performance tests show that this flexibility is achieved with a very acceptable performance loss. At MAC level, the round trip time of the anyMAC network stack differs by only 12% from that of the Sun SPOT network stack, while the throughput is only marginally affected.

1 Introduction

Sensor networks currently form an active area of research that attracts interest from both the academic community and the industry.
Because for most applications sensor nodes are expected to have a very long battery lifetime, many frameworks for developing sensor networks such as TinyOS [1] focus on 'efficiency' and 'energy consumption' rather than usability. Since writing applications for these kinds of platforms is no simple task, they are best suited for developing sensor networks of which the requirements are known in advance. When prototyping new sensor network applications or new sensor network protocols, it can be a more viable solution to use a Java based framework.
The Sun SPOT [2] sensor network platform is completely open-source and is fully Java ME compliant. It uses the open-source Squawk [2] VM: a Java virtual machine that runs directly above the hardware. Moreover, both the hardware drivers and software libraries for this platform are also written in Java and can therefore easily be customized. Consequently, the Sun SPOT platform is an interesting tool for prototyping both sensor network applications and new sensor network protocols. However, the Sun SPOT platform is currently less suited for prototyping new MAC protocols because the Sun SPOT network stack is heavily optimized for usage with Sun's IEEE 802.15.4 [3] MAC implementation. As a result, the design of the network stack does not allow for the default IEEE 802.15.4 MAC implementation to be easily

[1]Funded by an Aspirant grant of the Fund for Scientific Research Flanders (FWO).

replaced, making the development of new MAC protocols a very daunting task. Moreover, this limitation is also an obstacle for providing interoperability with other sensor network platforms, since doing so usually requires a MAC implementations to be ported to the Sun SPOT platform.
In order to address these shortcomings, we introduce the 'anyMAC project'. This project alters both the design and the implementation of the original Sun SPOT network stack in order to provide a network stack which is flexible enough to allow MAC implementations to be easily interchanged and to support the development of new MAC protocols for the Sun SPOT platform.
The specifications and the design of this new 'anyMAC network stack' are further discussed in section 2. Its performance is discussed in section 3 and we conclude this paper in section 4.

2 The anyMAC network stack

The 'anyMAC network stack' has been designed to meet the following specifications:
The MAC implementation should be interchangeable. When a MAC implementation is replaced by another implementation, this should have no semantic consequences layers above the MAC layer.
The design of the anyMAC network stack may not impose any semantic restrictions on the MAC layer. MAC developers may be required to implement certain interfaces, but these interfaces may not, for instance, enforce any restriction on the packet format used by the MAC implementation.
Alterations made by the anyMAC project must be transparent to the application. Applications that were originally developed to operate on top of the Sun SPOT network stack, should work (almost) out of the box in combination with the anyMAC network stack.
Applications should be able to use one specific MAC implementation. They should not only be able to decide which MAC implementation is used, but also be able to access functionality that is only provided by that MAC implementation.
In order to preserve compatibility with the original Sun SPOT network stack, the anyMAC project should modify the upper layers of the network stack as little as possible.
Although 'performance' and 'energy consumption' are important metrics when designing sensor networks or new MAC protocols, performance constraints were only considered to be a 'secondary requirement'. Wherever possible the design and implementation of the anyMAC network stack have been optimized for power efficiency. Enforcing concrete performance or energy constraints however, would have impeded upon the most important requirement of the anyMAC network stack, namely its flexibility towards MAC developers.
Moreover, Sun SPOTs only have a battery lifetime of a few hours to a few days at most. They will therefore most likely either be used for rapid prototyping of sensor network applications or be deployed in an environment where an external power supply is available.
In order to meet these criteria, the IEEE 802.15.4 specific MAC and physical layer of the Sun SPOT network stack have been replaced by a Hardware Abstraction Layer

(HAL) and a new MAC layer that allows MAC implementations to be interchanged. The higher layers of the Sun SPOT network stack, such as the LowPAN protocol [4] and Radiogram and Radiostream transport layer protocols, were then adapted to the new design of the MAC layer and integrated into the anyMAC network stack. The key elements of these two new layers are discussed in sections 2.1 and 2.2. For a more extensive explanation about the design of the anyMAC network stack we refer to [5].

2.1 Hardware Abstraction Layer

Because the design of the Sun SPOT network stack only considers the IEEE 802.15.4 MAC protocol, it's physical layer is tailored to the needs of Sun's IEEE 802.15.4 MAC implementation. In contrast, the HAL of the anyMAC network stack exposes the majority of the possibilities of the TI CC2420 radio chip to the higher layers to increase flexibility.
As shown in figure 1, the HAL uses a component based design where each component of the HAL is responsible for a specific functionality of the radio chip such as sending packets, changing the channel and configuring the power used to send packets. Not all of these 'HALComponents' have to be loaded at startup. This allows MAC developers to decide which of the available 'HALComponents' are used, depending on the requirements of their MAC implementation. Moreover, MAC developers are also able to add new HALComponents or to replace existing components. For example: the 'ChannelController' component shown in figure 1 only allows to change the channel to one of the 5MHz wide channels defined by the IEEE 802.15.4 standard, while the CC2420 radio chip is also capable of operating on other frequencies. By adding a new component to the HAL, MAC developers are thus able to define their own channels.

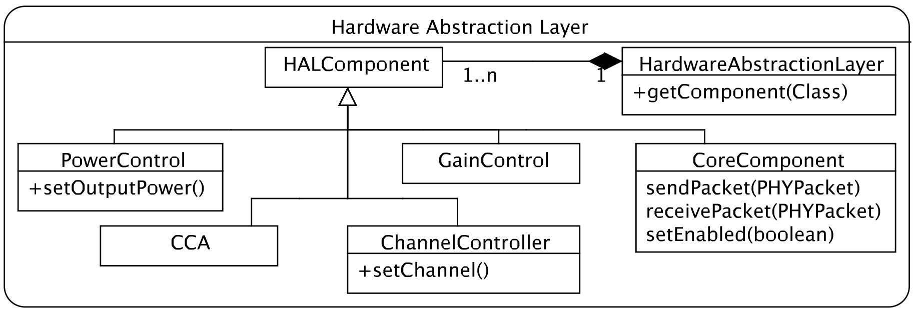

Figure 1: UML diagram of the Hardware Abstraction Layer of the anyMAC network stack.

2.2 MAC layer

The MAC layer of the anyMAC network stack has been designed so that it can only be accessed by the upper layers of the network stack through a collection of abstract Java classes and Java interfaces, called 'the Application Programming Interface of the MAC layer'. This allows multiple MAC implementations to be easily interchanged. Moreover, each MAC implementation that implements the API of the

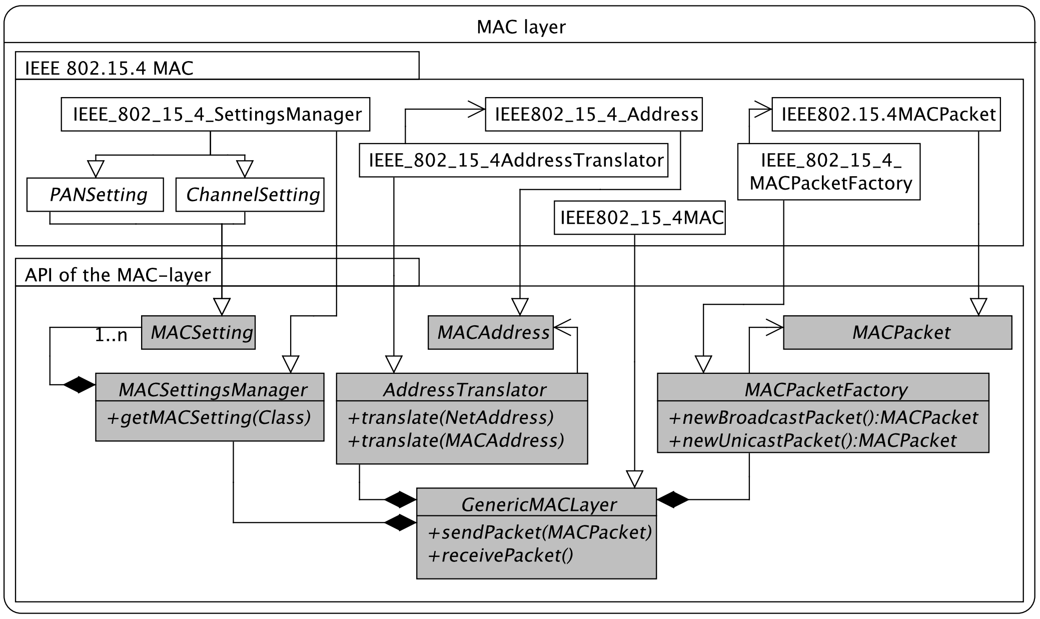

Figure 2: UML diagram showing the API of the MAC layer with a possible implementation.

MAC layer can be used in the anyMAC network stack. The design of this ‘API of the MAC layer’, together with a possible implementation, is illustrated by figure 2. The most important components of this API are now discussed.

The *MACPacket* class is an abstract base class that defines which packet operations must be provided by the MAC implementation. MAC developers can specify their own packet format by sub classing this class.

The *MACPacketFactory* is used to allow the higher layers of the network stack to create MACPackets without being aware of the actual packet format used.

The *AddressTranslator* is added to the API of the MAC layer in order to allow MAC developers to define their own address format. In the original Sun SPOT network stack, 64 bit IEEE 802.15.4 MAC addresses are used by the MAC as well as the LowPAN, transport and application layer to identify the nodes in the sensor network. The API of the MAC layer however, allows MAC developers to define their own address format. Since the LowPAN protocol operating above the MAC layer expects the MAC layer to use the IEEE 802.15.4 address format, a translation mechanism is used to translate between the IEEE 802.15.4 MAC addresses expected by the LowPAN layer and the MAC implementation specific MAC addresses used by the MAC layer. This mechanism is accessed through the *AddressTranslator* interface. MAC developers are thus able to define their own address format, on the sole condition that they provide the needed translation mechanism. This mechanism may be implemented, for instance, by using hashing, lookup tables or even a variation of the ARP protocol.

The *MACSettingsManager* is used to allow applications to access MAC implementation specific settings and functionality without requiring the upper layers to be aware of which MAC implementation is used. Each MAC implementation may im-

plement zero or more *MACSetting* interfaces. MAC developers may also define new MACSetting interfaces to describe functionality that is not described by the existing MACSetting interfaces. Applications can then ask the MACSettingsManager whether the used MAC implementation implements a certain MACSetting interface or not. For example, as shown in figure 2, the IEEE 802.15.4 MAC implementation implements both the ChannelSetting and PANSetting MACSettings. When a MAC implementation is used that does not support the use of PAN ids the PANSetting interface is not available.
The API's *GenericMACLayer* component is responsible for sending and receiving packets. It also acts as a facade to the other components of the API of the MAC layer so that most functionality provided by these components can be directly addressed through the *GenericMACLayer* component.
In order to demonstrate the flexibility of the MAC layer's API a proof of concept MAC implementation called 'DemoMAC' was implemented. This MAC implementation intentionally uses a packet format which is incompatible with that of IEEE 802.15.4 and employs a simple CSMA based mechanism to control access to the (wireless) medium. DemoMAC identifies nodes by using 16 bit addressing and applies a variation of the ARP protocol in order to translate between these 16 bit addresses and the 64 bit addresses used by the LowPAN layer.

3 Performance analysis

The difference in performance between the Sun SPOT network stack and the anyMAC network stack was investigated by measuring the throughput and round trip time (RTT) between two Sun SPOTs equipped with either the default Sun SPOT network stack or with the anyMAC network stack. The Sun SPOT network stack can only use its default IEEE 802.15.4 MAC implementation. Because this MAC implementation does not implement the API of the MAC layer, (as defined in section 2.2) we created an 'anyMAC-compatible' version of this MAC implementation. This implementation is identical to the MAC implementation in the Sun SPOT network stack except for a few cosmetic alterations that were needed for compatibility with the anyMAC network stack. This new version was then used in the anyMAC network stack during the performance tests.
The results of these tests are shown in figure 3. At MAC level, the round trip time of the anyMAC network stack is 12% higher than that of the Sun SPOT network stack. At application level the difference between these round trip times is 26%. These differences are explained by the fact that the Sun SPOT network stack is heavily optimized for usage with Sun's implementation of the IEEE 802.15.4 MAC protocol. By contrast, such optimizations cannot take place in the anyMAC network stack since it must be able to handle more than one MAC implementation.
The throughput of the anyMAC network stack is, at MAC level, almost indistinguishable from the throughput of the Sun SPOT network stack. When the throughput is measured at application level, the throughput of the anyMAC network stack does differ from the throughput of the Sun SPOT network stack, but the difference is smaller than 8%. This difference is, again, explained by the fact that the anyMAC network stack is optimized for flexibility rather than performance in combination

with one particular MAC implementation.

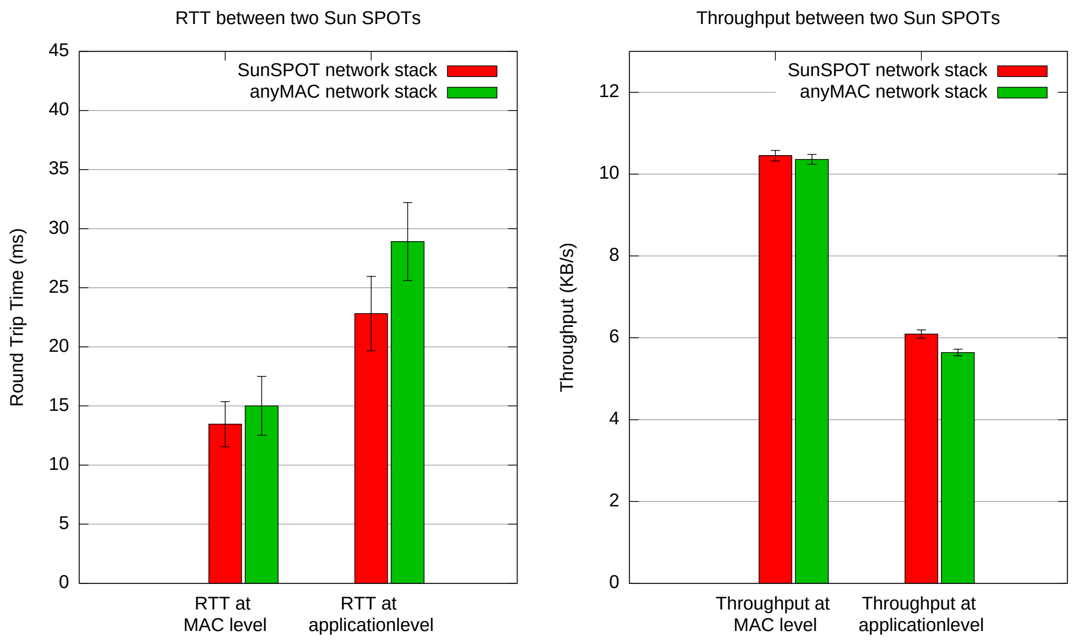

Figure 3: RTT and Throughput of the anyMAC and Sun SPOT network stack.

4 Conclusions and future work

The anyMAC project provides an alternative network stack for the Sun SPOT platform that allows the MAC implementation to be easily replaced. It is flexible enough to allow for the development of new MAC protocols that can be implemented in Java and can run on top of the TI CC2420 radio chip. At MAC level this flexibility is attained with a very acceptable loss in performance, compared to the default Sun SPOT network stack. The anyMAC network stack may be improved by further optimizing the LowPAN and transport layer to the new design of the MAC layer.

External links

anyMAC project website: `http://www.pats.ua.ac.be/software/anymac`

References

[1] Levis Phillip. *TinyOS*, October 2006. `http://www.tinyos.net`.

[2] Doug Simon, Cristina Cifuentes, Dave Cleal, John Daniels, and Derek White. Java(tm) on the bare metal of wireless sensor devices: the squawk java virtual machine. In *VEE '06: Proceedings of the 2nd international conference on Virtual execution environments*, pages 78–88, New York, NY, USA, 2006. ACM.

[3] IEEE std 802.15.4(tm)-2006. Technical report, The Institute of Electrical and Electronics Engineers, 3 Park Avenue, New York, NY 10016-5997, USA, June 2006.

[4] Montenegro G., Kushalnagar N., Hui J., and Culler D. RFC 4944: Transmission of IPv6 packets over IEEE 802.15.4 networks. Technical Report 4944, Internet Engineering Task Force, September 2007. `http://www.ietf.org/rfc/rfc4944.txt`.

[5] van den Akker Daniel. anyMAC: Aanpassingen aan de netwerkstack van het sunspot platform voor het ontwikkelen van nieuwe mac-protocollen. Master's thesis, University of Antwerp, May 2009. `https://euterpe.cmi.ua.ac.be/~dvdakker/files/anyMAC_masterthesis.pdf`.