MultiMAC : a multiple MAC network stack architecture for TinyOS

MultiMAC: A Multiple MAC network stack architecture for TinyOS

Daniel van den Akker* , Chris Blondia
PATS research group, Department of Mathematics and Computer Science
University of Antwerp – IBBT, Antwerp, Belgium
{daniel.vandenakker,chris.blondia}@ua.ac.be

***Abstract*—Due to the extreme energy requirements often found in sensor networks and the fact that these sensor networks are being used for an increasingly wide variety of applications, there currently exist many different sensor network MAC protocols each optimised for a different set of application requirements. Since these MAC protocols are not compatible with each other, there is generally no interoperability between sensor networks. This lack of interoperability becomes a severe problem when connectivity between sensor networks is required. In this paper, we demonstrate the feasibility of using *Virtual Gateways*, sensor nodes running multiple MAC protocols simultaneously on top of a single radio interface, to enable connectivity between sensor networks. To this end we have developed MultiMAC, a network stack for TinyOS capable of running multiple MAC protocols at the same time. We show that the architecture of the MultiMAC stack is flexible and extensible enough to support a wide variety of MAC protocols and that the overhead of this network stack is minimal.**

I. Introduction

In sensor networks, energy efficiency has always been one of the most important design considerations. As a result, it has become rule rather than exception to optimise or even custom build the entire network stack to the needs of a specific sensor network application. Over the years, this has led to the creation of many different sensor network MAC protocols, each tailored to a different set of application requirements. For instance, Low Power Listening MAC (LPL-MAC) protocols, such as B-MAC [1] and its more recent successor X-MAC [2] optimise energy usage for variable-bitrate, non-infrastructure sensor networks with relaxed delay and low bandwidth requirements. TDMA-based MAC protocols on the other hand, work best in infrastructure environments where a constant bitrate is required. While these kinds of specialised sensor network MAC protocols allow the operation of the sensor network to be greatly optimised, they are also incompatible with each other and as a result there is generally no interoperability between different sensor networks. This poses a severe problem in the case that connectivity between different sensor networks is required. The traditional manner to achieve interoperability between wireless networks is to enforce the use of a standardised Physical and MAC layer for all networks based on the same wireless technology and use gateway nodes to enable communication between networks based on different technologies (such as for instance between Wifi and cellular networks). The standardisation approach has been only partially successful for sensor networks. E.g: the IEEE 802.15.4 [3] specification defines a standard Physical and MAC layer aimed at low power sensor networks, but only the Physical layer has been widely adopted. The MAC layer is often only partially implemented or replaced entirely by a more optimal protocol. Moreover the standardisation approach cannot be used for providing interoperability between existing networks.

The second approach of introducing gateways into the network seems to be a more viable solution since it would allow each network to continue using its own MAC protocol. Unfortunately this requires gateway nodes to be deployed in the wireless environment. These gateway nodes would need to be equipped with multiple radio interfaces and therefore require the development of specialised hardware. As a result, this approach is infeasible for most scenarios.

In this paper we propose an alternative approach for providing MAC-layer interoperability between heterogeneous sensor networks. We exploit the fact that all sensor network MAC protocols generally operate on top of a single, standardised Physical layer: the one specified in IEEE 802.15.4 [3]. Instead of using regular gateways to enable communication between sensor networks, we propose to use *Virtual Gateways*. On a normal gateway node each MAC protocol operates on top of a separate radio interface. On a *Virtual Gateway* all MAC protocols make use of a single, shared, radio interface. Virtual gateways have a number of advantages compared to normal gateways. Firstly, they only require one radio interface and therefore do not need specialised hardware to be developed. Secondly, since every node in a sensor network is already equipped with a radio interface, it can be configured as a virtual gateway by performing a software upgrade. There is no need to add new nodes. Moreover, these nodes can also dynamically enable or disable specific MAC protocols depending on the requirements of the nodes and the traffic conditions in the wireless environment. To investigate the feasibility of this approach, we have developed *MultiMAC*, a network stack that is capable of running multiple MAC protocols simultaneously. The architecture and implementation

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

Part of the research leading to these results has received funding from the Agency for Innovation by Science and Technology (IWT) as part of the Symbionets project and from the Interdisciplinary Institute for Broadband Technology (IBBT).

of this network stack are discussed in more detail in Section II. In Section III the network stack is evaluated and we conclude this paper in Section IV.

II. The MultiMAC network stack

The MultiMAC stack was developed from scratch for the Tmote Sky platform [4] in TinyOS [5] 2.1.0. In order for Virtual Gateways to be a feasible solution, the MultiMAC network stack should be able to meet the following requirements:

Minimal overhead: In sensor networks energy efficiency is significantly more important than interoperability. As a result the cost of introducing interoperability between networks should be as small as possible in order for the benefits of interoperability to outweigh the costs. This implies that the impact of the MultiMAC stack on the performance of the network should be as small as possible in order for Virtual Gateways to be feasible.

Flexibility: To ensure that Virtual Gateways are generally usable, the MultiMAC network stack should be flexible enough to accommodate a wide range of MAC protocols. Within the scope of this work, we required the MultiMAC stack to be able to support the two main categories of sensor network MAC protocols: contention-based and time-division based MAC protocols.

Extensibility: Virtual Gateways would not be a feasible solution if the entire network stack would need to be rewritten from scratch each time a MAC protocol is added or removed from the code base. Consequently, developers should be able to easily extend the MultiMAC network stack with new protocols. Moreover, the protocols in the MultiMAC stack must be isolated from each other as much as possible to ensure that MAC developers do not need to take the presence of other MAC protocols into account. Section II-A discusses the requirements that the MultiMAC stack imposes on the frame format, while in Section II-B the architecture of the MultiMAC network stack is explained. Section II-C discusses a number of interesting issues that were encountered during the implementation MultiMAC.

A. Frame Format

One of the prerequisites for using Virtual Gateways is that nodes must be able to distinguish between frames sent by different MAC protocols. This is currently not possible due to the wide variety of sensor network MAC protocols that are available and the low level of standardisation between the frame formats used. To circumvent this problem, the MultiMAC stack imposes a number of limitations on the frame formats that can be used. Firstly, MAC protocols are required to use a frame-format that is compatible with IEEE 802.15.4. This requirement should not pose too great a problem since most sensor network MAC protocols use a frame format that is, mostly due to technical limitations, already largely compatible with that of IEEE 802.15.4. Moreover this frame format is very flexible and does not even require any addressing information to be present in the frame. Secondly each MAC protocol is assigned a unique *MAC-id*. As shown in Figure 1, this MAC-id is stored in one of the reserved fields of the MAC header.

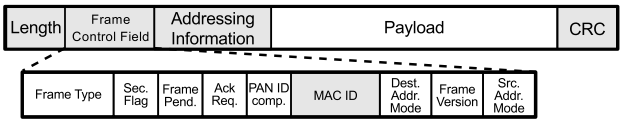

Fig. 1. The frame format used by the MultiMAC network stack

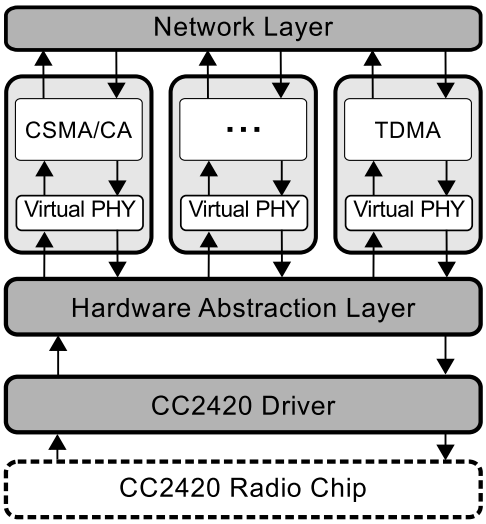

Fig. 2. The Architecture of the MultiMAC network stack

B. Architecture

The architecture of the MultiMAC network stack is outlined in Figure 2.

The *CC2420 Driver* is responsible for all interactions with the CC2420 radio chip. It manages the transmission and reception of data frames and is also responsible for turning the radio on and off. One of the most important features of this component is that it allows MAC protocols to control certain aspects of how frames are transmitted and received. Not only are MAC protocols able to enable or disable Clear Channel Assessments (CCA) when sending a frame, but they are also able to decide whether or not the transmission of a frame may be delayed in order to allow an incoming frame to be received. This flexibility is needed to support both contention-based and time-division based MAC protocols. Contention-based MAC protocols will usually require CCA-checks to be performed and can generally allow the transmission of a frame to be delayed. Time division-based MAC protocols on the other hand have very strict timing requirements and will therefore usually require the frame to be transmitted without delay.

The *Hardware Abstraction Layer* (HAL) operates on top of the CC2420 Driver and is responsible for 'multiplexing' the different MAC protocols in the network stack on top of the radio interface. When a frame is received by the HAL for instance, it will dispatch the frame to the correct MAC protocol based on the MAC-id stored in the frame. When a MAC protocol wishes to transmit a frame, the HAL will only pass that frame on to the CC2420 Driver if no other transmission is currently in progress. Otherwise, the HAL reports to the MAC protocol that the channel was busy. Although in this case one

of the MAC protocols is prevented from transmitting its frame, this is still more preferable to what would have happened if both MAC protocols had accessed the same channel using separate radio interfaces. In that case either the CCA-check would have failed in one of the two radio interfaces or both frames would have been transmitted and would have consequently collided with each other. The only advantage of using two radio interfaces is that these radio interfaces could be configured to operate on different channels. As discussed in Section III-C however, using two radios consumes much more energy than multiplexing multiple MAC protocols on top of a single interface. A similar mechanism is used for enabling or disabling the radio interface. Each MAC protocol can enable or disable the radio through the Virtual PHY interface, but the radio will only be actually disabled if all MAC protocols have disabled the radio.

The HAL is also capable of performing address recognition and sending automatic acknowledgements for each MAC protocol individually. Normally, the CC2420 radio chip is able to perform these functions in hardware and as a result the time between the reception of a correct MAC frame and the transmission of the acknowledgement is only 128μs. Unfortunately, the MAC-id of these hardware generated acknowledgements is always equal to zero, which means that the HAL is not able to discern between acknowledgements of different MAC protocols. As a result, hardware acknowledgements cannot be used in the MultiMAC stack. By transmitting acknowledgements from the HAL rather than from the MAC implementation itself, we were able to reduce the ACK-turnaround time to less than 800μs. This increase in ACK-turnaround time is mostly due to the time that is needed to transfer the frames between the micro controller and the radio chip. On other platforms that are equipped with a micro controller with an embedded radio chip (such as the STM32W [6]), it should be possible to reduce this additional delay even further.

The MAC protocols in the network stack are each presented with a *Virtual PHY* interface that can be used to perform a number of Physical Layer operations such as sending and receiving packets and turning the radio on and off. To ensure that the different MAC protocols remain isolated from one another, they can only access the HAL through this interface. Although the Virtual PHY interface provides most operations required for implementing a MAC protocol, some operations have been excluded from this interface on purpose. MAC protocols are not able to alter the transmission power or change the channel of the radio interface, since these operations not only affect the MAC protocol itself but also affect all the other MAC protocols in the network stack.

C. Implementation Concerns

While implementing the MultiMAC stack, a number of interesting and often frustrating issues were encountered that required significant changes to be made. Firstly the CC2420 Driver proved to be one of the most difficult components to implement. This was partially due to a number of undocumented features of the CC2420 radio chip, but mostly due to the fact that the efficiency of this component was critical for minimising the overhead of the entire stack. Secondly, the latency induced by the TinyOS timer implementation proved to be too large for these timers to be used by time-critical MAC protocols. Therefore a smaller, simpler and more efficient interface had to be developed. A third issue was the lack of support for real-time processing in TinyOS. TinyOS uses a scheduling mechanism whereby tasks are executed on a first-come-first-served basis and can only be interrupted by interrupt handlers. For time-critical MAC protocols this mechanism does not suffice since there is no upper bound on the execution time of tasks and as a result time-critical tasks may have a large queueing delay. To address this issue, an 'Interrupting task scheduler' was added to the MultiMAC stack. This interrupting task scheduler allows so-called 'Interrupting tasks' to be scheduled with a number of different priorities. Like TinyOS tasks, interrupting tasks can be interrupted by interrupt handlers. Unlike TinyOS tasks however, interrupting tasks are able to interrupt both TinyOS tasks and other interrupting tasks that have a lower priority. By using the Interrupting Task scheduler, MAC protocols are thus able to ensure that time-critical tasks are executed on time.

III. Evaluation

A. Flexibility & Extensibility

Three different MAC protocols were implemented for the MultiMAC stack. While the MultiMAC stack was being developed, the CSMA/CA MAC protocol was implemented as a simple test protocol. After the MultiMAC stack had been completed, two other MAC protocols were implemented: an LPL-MAC protocol, which uses the same channel access mechanism as the LPL-MAC protocol provided by TinyOS and a TDMA MAC protocol with very strict timing requirements. The fact that implementing these protocols proved to be very straightforward, even after the completion of the MultiMAC stack itself, shows that this stack is sufficiently *Flexible* to support a wide range of MAC protocols and that it is *Extensible* enough to allow MAC protocols to be easily implemented.

B. Overhead When using a Single MAC protocol

To measure the overhead of the MultiMAC stack when only a single MAC protocol is used, the performance of this stack is compared to that of the TinyOS and the 'Passthrough' network stack. The 'Passthrough' stack is derived from the MultiMAC stack by replacing the HAL component with a component that passes the calls from the MAC protocol directly to the CC2420 Driver and vice versa. Unlike the TinyOS network stack, the Passthrough network stack has, apart from the different HAL, exactly the same code base as the MultiMAC stack. Any difference in performance between these stacks should be the result of the Multiplexing that is performed by the MultiMAC HAL.

To measure the overhead of the MultiMAC stack three different metrics are considered: throughput, delay (round trip time) and duty cycle. The duty cycle of the node is used as a

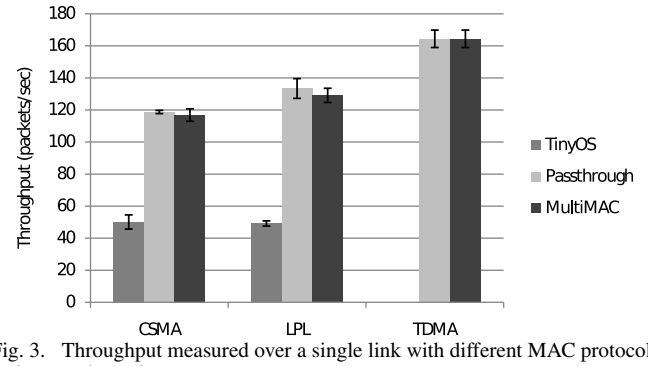

Fig. 3. Throughput measured over a single link with different MAC protocols and network stacks.

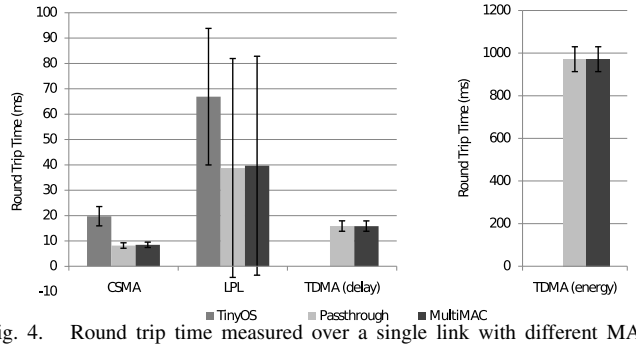

Fig. 4. Round trip time measured over a single link with different MAC protocols and network stacks.

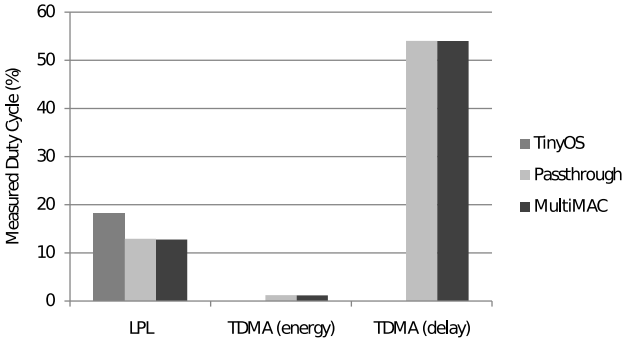

Fig. 5. Average Duty Cycle measured on two nodes with different MAC protocols and network stacks.

measure for the energy efficiency of the network stack. Some MAC implementations allow the duty cycle to be configured by the user, in which case the timings of the MAC protocol needed to achieve the specified duty-cycle are calculated based on the known overhead of the MAC implementation and network stack. For these tests however, the MAC protocols use fixed timings so that the measured duty cycle can be used to compare the energy efficiency of the network stacks.

Throughput is measured by continuously sending maximum-sized data frames from one node to another over a single link, for sixty seconds. The number of successfully transmitted packets is recorded and this test is repeated 20 times. The delay is measured by sending 'ping' and 'ping-reply' messages between two nodes once every four seconds and recording the round trip time. This test is repeated 300 times. While measuring the round trip time, the duty cycles of both nodes are also measured. To this end both nodes are connected to a logic analyser which is able to capture the precise moments that the radio is turned on and off. After the round trip time had been measured for a specific test scenario, the duty cycle over the entire test run is calculated.

In Figure 3 the single-hop throughput measured between two nodes running either the TinyOS, Passthrough or MultiMAC stack is shown for the CSMA/CA, LPL and TDMA MAC protocols. As with all following figures, the bars represent the average over all measured values while the whiskers represent the standard deviation. As this figure shows, the MultiMAC and Passthrough network stack have a much higher throughput than the TinyOS network stack when the CSMA/CA and LPL MAC protocols are considered. When the throughput of the MultiMAC stack is compared to that of the Passthrough network stack, it is shown that the throughput of the Passthrough network stack is only marginally larger than that of the MultiMAC stack for the CSMA/CA and LPL-MAC protocols (1.65% and 3.32% respectively).

Figure 4 shows the round trip time measured between two nodes using either the CSMA/CA, LPL or TDMA-MAC protocol. For the TDMA MAC protocol, the round trip time is highly dependent on which slot allocation is used. Therefore the round trip time was measured both with a slot allocation optimised for minimal delay and a slot allocation optimised for minimal energy consumption. As this figure shows, the Passthrough and the MultiMAC stack have almost the same round trip time both for all three MAC protocols. Moreover the two network stacks outperform the TinyOS stack both when the CSMA/CA protocol and when the LPL-MAC protocol is used. The relatively large standard deviation of the round trip times measured for the LPL-MAC protocol are explained by the fact that when using this MAC protocol nodes disable their radios for most of the time and it takes a variable amount of time to wake these nodes up.

The duty cycle of the nodes is shown in Figure 5. Since the duty cycle could only be measured over the entire test run of all round trip time measurements, only a single data point was collected for each test scenario and as a result there are no error bars shown in Figure 5. Despite this, the true duty cycle should be accurately reflected since these values were measured over a relatively long time period. The duty cycle for the CSMA/CA MAC protocol was always almost 100% and is therefore not shown. For the LPL-MAC and TDMA MAC protocols, the duty cycle of the Passthrough network stack is almost exactly the same as that of the MultiMAC network stack. Moreover, both network stacks outperform the TinyOS network stack when only the LPL-MAC protocol is considered.

C. Multiple MAC protocol Overhead

To measure the impact of running multiple MAC protocols on a single node, a test setup with three nodes is used whereby a single 'router node' routes packets between two 'end nodes'. When both end nodes use the same MAC protocol, the router node is equipped with the same MAC protocol and acts as a regular router. When the end nodes use different MAC protocols, the router node is equipped with both MAC

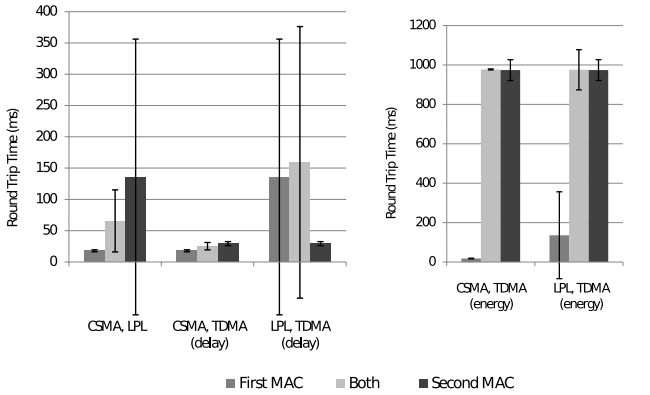

Fig. 6. Round trip time measured over two links with the same and different MAC protocols.

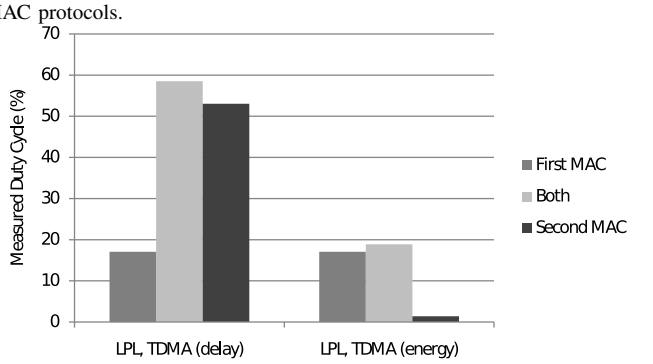

Fig. 7. Duty Cycle of a Virtual gateway running both a single and two different MAC protocols.

protocols and acts as a Virtual Gateway. For these tests the round trip time is measured between the two end nodes. At the same time the duty cycle of the *router node itself* is also measured. By comparing the measurements for the case where only a single MAC protocol is used to the case where two MAC protocols are used, the overhead of running multiple MAC protocols can be determined.

The measured round trip times are shown in Figure 6. For each category, the first and third value show the round trip time when all three nodes are using either the first or the second MAC protocol. The middle value shows the round trip time when both MAC protocols are being used. This figure shows that when a Virtual Gateway is used, in most cases the round trip time lies between the round trip times when only a single MAC protocol is used. The only exception is in the case where the LPL-MAC protocol and TDMA MAC protocol (optimised for delay) are used. In that case the round trip time is 17% higher than when only LPL-MAC is used which is not negligible but still relatively small.

The measured duty cycle is shown in Figure 7. As with the Single MAC tests, the duty cycle was almost 100% when CSMA/CA was used and these measurements are therefore not shown. As can be seen in this figure, using more than one MAC protocol on a single node has a noticeable effect on the duty cycle. This increase in duty cycle is to be expected since there is more than one MAC protocol that may want to keep the radio enabled. Both when using a slot assignment optimised for delay and when using a slot assignment optimised for energy, the duty cycle is almost 10.5% higher than in the case where only the least energy efficient MAC protocol is used. Although this increased duty cycle needs to be taken into account when deploying Virtual Gateways, this increase is not sufficiently large to make the use of Virtual Gateways infeasible. Moreover, the increase in power consumption resulting from this increased duty cycle is actually fairly small considering that a usual gateway would need to power two radio interfaces and would therefore consume at the very least the sum of the energy used by each individual MAC protocol.

IV. Conclusion & Future Work

In this paper we have investigated the feasibility of using *Virtual Gateways* to enable connectivity between sensor networks using heterogeneous MAC protocols. To this end a network stack capable of running multiple MAC protocols (MultiMAC) was developed in TinyOS for the Tmote Sky platform. We have shown that this MultiMAC stack is *Flexible* and *Extensible* enough to support a wide variety of MAC protocols and that its *Overhead* compared to other network stacks is minimal. Given the fact that the MultiMAC stack was developed for an extremely resource constrained sensor node it should not only be feasible to use Tmote Skys but also a wide range of more recent and less resource constrained nodes as Virtual Gateways. When Virtual Gateways are used to enable connectivity between sensor networks, the effect that these Virtual Gateways have on the overall operation of these networks is likely to be largely dependent on the location of the Virtual Gateways in these networks. Future work will focus on determining where Virtual Gateways should be deployed to enable connectivity between sensor networks in the most optimal manner.

Acknowledgements

The authors would very much like to thank Wim Torfs for his invaluable contribution to the development of the MultiMAC stack. Without his advice and expertise on low-level programming, implementing the MultiMAC stack would have been an impossible task.

References

[1] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM New York, NY, USA, 2004, pp. 95–107.

[2] M. Buettner, G. Yee, E. Anderson, and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proceedings of the 4th international conference on Embedded networked sensor systems*. ACM, 2006, p. 320.

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

[4] *Tmote Sky Low Power Wireless Sensor Module*, moteIv, http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf.

[5] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *SIGPLAN Not.*, vol. 35, no. 11, pp. 93–104, Nov. 2000.

[6] *High-performance, IEEE 802.15.4 wireless system-on-chip with up to 256 Kbytes of embedded Flash memory*, STMicroelectronics, http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00248316.pdf.