Practical issues in the development of a minimalistic power management solution for WSNs

Jorge M. Soares and Bruno J. Gonçalves

Instituto Superior Técnico – Technical University of Lisbon,
Av. Prof. Dr. Cavaco Silva,
2744-016 Porto Salvo,
Lisbon, Portugal
Email: jorgesoares@ist.utl.pt
Email: brunojfgoncalves@ist.utl.pt

Rui M. Rocha*

Instituto de Telecomunicações,
Instituto Superior Técnico – Technical University of Lisbon,
Av. Prof. Dr. Cavaco Silva,
2744-016 Porto Salvo,
Lisbon, Portugal
Email: rui.rocha@lx.it.pt
*Corresponding author

Abstract: A flexible Wireless Sensor Network platform for implementation of diverse applications has been developed and deployed at Instituto Superior Técnico - Technical University of Lisbon (IST-TUL). Since its initial deployment in 2007, this testbed has grown steadily, supporting new nodes, applications and experiments. However, some initial problems, which were solved on an ad hoc basis, were becoming more serious as the network spanned throughout the campus. Major issues, like global power management, have to be tackled not only with traditional protocol level approaches but also from a system's viewpoint, providing solutions capable of guaranteeing a consistent testbed. We discuss the main issues related with the development of power management solutions, presenting our architecture, design choices and implementation, and address the lessons learnt from its integration. Experimental evaluation of our solution has shown considerable energy savings, extending network lifetime by up to nine times.

Keywords: power management; WSN testbeds; radio cycling; time synchronisation; synchronous rounds; energy efficiency; energy saving; development issues; infrastructure services; network protocols; ad-hoc sensor networks.

Reference to this paper should be made as follows: Soares, J.M., Gonçalves, B.J. and Rocha, R.M. (2010) 'Practical issues in the development of a minimalistic power management solution for WSNs', *Int. J. Sensor Networks*, Vol. 8, Nos. 3/4, pp.136–146.

Biographical notes: Jorge M. Soares (IEEE S'04, GSM' 07) is currently a PhD candidate in Distributed and Cognitive Robotics at IST-TUL and Ecole Polytechnique Fédérale de Lausanne, Switzerland. He received his BSc and MSc degrees in Communication Networks Engineering at Instituto Superior Técnico – Technical University of Lisbon (IST-TUL) in 2007 and 2009, respectively. His research activities focused on wireless sensor networks and, specifically, on the use of opportunistic communications in that context.

Bruno J. Gonçalves (IEEE S'07, GSM' 08) is currently doing research for his MSc thesis, dealing with home networking and applications for smart homes. He received his BSc degree in Communication Networks Engineering at Instituto Superior Técnico – Technical University of Lisbon (IST-TUL) in 2008.

Rui M. Rocha (IEEE S'89, M'94, SM'07) has been working at the Instituto Superior Técnico – Technical University of Lisbon (IST-TUL) since 1986, where he is now an Associate Professor with the Electrical and Computer Engineering Department. He is also affiliated with the Instituto das Telecomunicações (IT) in Lisbon, leading the GEMS research group. Graduated in Electrical Engineering at the IST-TUL in 1981 and received MSc and PhD degrees in Electrical and Computer Engineering in 1987 and 1995, respectively. His current research interests are focused on the heterogeneous networks area, namely on self-organised networks and wireless sensor systems.

1 Introduction

Over the last few years, Wireless Sensor Networks (WSNs) have been gradually moving from a mostly theoretical, academic approach to real-world roles in industrial and commercial applications. The availability of better and increasingly cheaper sensor platforms has decisively contributed to foster WSN deployment worldwide.

Recognising this trend, and taking advantage of the experience gained through the participation in several internal and external projects in this area, IST-TUL decided to develop and deploy a WSN testbed at one of its campus (Pedrosa et al., 2009). Since then, Tagus-SensorNet, as the testbed was later named, has grown to support additional applications and experiments. New nodes were added and the whole network has been migrated to TinyOS 2.1, a new major version of the operating system (see http:// www.tinyos.net/scoop/section/Releases). Several students worked on the project, developing new hardware and software components.

However, as with any other deployment of its kind, there are several operational problems affecting Tagus-SensorNet, of which the rapid decay of energy levels, and consequent short network lifetime, was one of the most serious. Even taking advantage of Lower Power Listening (LPL) modes (Moss et al., 2007a), low routing traffic activity, and in-network processing techniques for compressing collected data, without a power management solution batteries had to be replaced approximately once a week. Considering the number of nodes and the difficulty in accessing some of their locations, such frequent replacement was simply not practical. This caused the network to be unavailable most of the time, requiring activation every time a specific experiment or demonstration was to be performed.

Clearly this situation was far from optimal, as one could not tap the otherwise continuous stream of data made available by the deployed applications. Aiming to enable always-on operation of Tagus-SensorNet, the decision to invest in the development of an overlay solution for this energy problem was taken. The solution – Tagus-SensorNetPM or TagusPM for short – involving both node-level and network-wide power management schemes comprises two basic components:

- *A hardware component*, consisting of higher capacity batteries, energy harvesting schemes and quicker and easier recharge methods.
- *A software component*, meant to reduce energy consumption and allow easy monitoring of nodes' energy status.

The hardware component uses environmental energy harvesting, either based on solar or vibration sources, to charge a lithium rechargeable battery through a dedicated interface circuit that also drives a supercapacitor, the secondary energy buffer of the system. The supercapacitor is also used to rapidly replenish the energy supply of sensor nodes in situations where there is no easy way to install the harvesting technology (e.g. there are no light or vibration sources available) and there is easy access to the node's hardware. In such a case, the use of a small-sized lead battery can present an interesting solution to quickly recharge the motes. This intelligent power supply is managed by an ultra low-power microcontroller that interfaces with the sensor node main board through an I^2C interface, providing information about energy resource levels that can be used for monitoring purposes.

While the hardware component is still under development, the required software has already been implemented and deployed, and will be our focus for the rest of this paper.

Our goal is not to present a revolutionary solution for energy saving in WSNs, but rather to recount our experiences with development and real-life deployment. Of the numerous previously proposed approaches to solving this problem, most take the form of power-aware single-layer protocols. Popular examples come from the MAC (e.g. S-MAC, WiseMAC and B-MAC; Demirkol et al., 2006) and routing (e.g. PEGASIS, TEEN and MECN; Akkaya and Younis, 2003) camps. There are also some, albeit fewer, more encompassing approaches, such as the ones presented by ALPL (Jurdak et al., 2007) and UPMA (Xing et al., 2009). These approaches will later be described in more detail. A real application, featuring both software techniques and energy harvesting hardware, can be seen in the ZebraNet Project (Juang et al., 2002; Zhang et al., 2004).

The remainder of this paper is organised as follows: in Section 2, we briefly present our target network, Tagus-SensorNet; in Section 3, we discuss the requirements and initial choices related to the system design; in Section 4, we provide an overview of the system architecture and the reasons that led to it; in Section 5, we detail the implementation, the development process and the problems we faced; in Section 6, we show some experimental results; in Section 7, we summarise some related work; finally, in Section 8 we draw some conclusions and finish by suggesting some future work.

2 Tagus-SensorNet

Instituto Superior Técnico – Technical University of Lisbon (IST-TUL) has been involved in several WSN experiments, acquiring important experience in the subject. However, most of these initial experiments were conducted in isolated testbeds, requiring significant duplication of resources and effort in their development and deployment. This led to the idea of creating a common testbed that would integrate the multiple projects into a single network and also provide common functionality that eased application development.

This common testbed – Tagus-SensorNet – is mostly composed of Crossbow MicaZ nodes (Crossbow Technology, 2004b). There are still some Mica2 nodes deployed (Crossbow Technology, 2004a) in the process of being phased out, and Sun Microsystem's SPOT (Smith, 2007) nodes were recently acquired, with integration between both platforms planned for the near future. On the software side, it is based on TinyOS (Levis et al., 2005), an operating system for networked embedded systems that uses nesC (Gay et al., 2003), an extension of the traditional C language.

Development work on the testbed was followed by two different lines of work. One of them deals with the development of a software framework that allows the coexistence of multiple applications and provides a set of key services. The other involves the development and deployment of applications on the network, enabling the trialling of new concepts while also providing useful functionality. For a detailed description of both the framework and some of the applications, we refer the reader to see Pedrosa et al. (2009).

2.1 Testbed architecture

Considering the high level of flexibility that the goals set for our testbed required, a traditional single-sink network would be unsuitable. The chosen topology uses a multi-sink network, with Ethernet connections between the several sub-networks at separate physical locations.

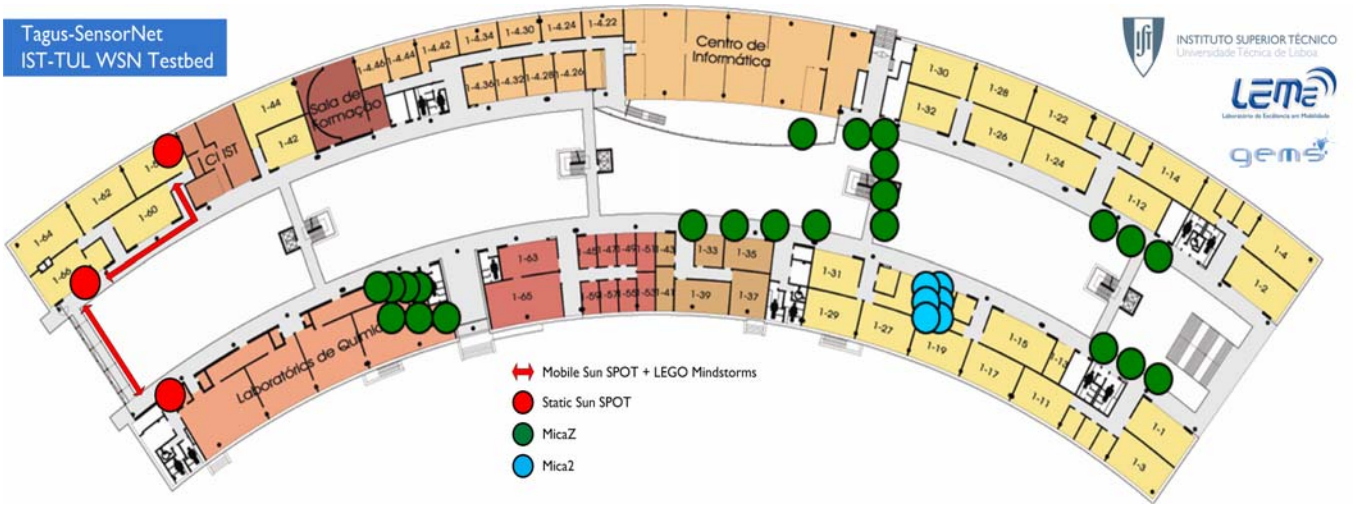
Tagus-SensorNet includes a total of 34 nodes, forming five connected sub-networks around the IST-Taguspark main building, as shown in Figure 1. The main connected region lies in the middle of the building and includes five hallway motes (used as general purpose monitoring nodes), four bridge motes (used to measure structural vibrations) and a sink node. The connected region immediately to the left is made up of seven air quality monitoring motes installed in the chemistry laboratory. The six motes that are deployed within a single room, just to the right of the main area, are part of an ultrasound user location project. Finally, the nodes to the far right and far left of the building are used for two different experiments in opportunistic communications.

2.2 Services

To make application development easier, a set of common services was integrated into the Tagus-SensorNet framework. They are implemented as shared libraries that can be used by any application within the network. The most relevant of these services are:

- *A centralised graphical management console* that allows users to interact with the different applications from a management station or over the internet.
- *A local user interface panel*, implemented as an LCD and keypad pair that provides a way for users to interact with part of the network within the monitored environment.
- *Convergecast data collection*, using the TinyOS native routing capabilities with custom extensions to enable data multiplexing and run-time configurable fixed routing.
- *Unicast data delivery*, allowing an application running in the management console to send data to individual nodes, typically orders, requests or configuration parameters.
- *Time synchronisation*, using the TinyOS-provided FTSP implementation, enabling the development of applications that depend on synchronous operations or network-wide time stamping.

Figure 1 Tagus-SensorNet testbed plan (see online versions for colours)



2.3 Applications

While development of applications for Tagus-SensorNet is a continuous and ongoing effort, several have already been deployed and run successfully on our testbed:

- An environmental interaction application that allows users to configure what data are collected within the network and how it should be processed.
- A vibration monitoring application that allows users to either collect raw or pre-processed vibration data to monitor structural health.
- A temperature mapping application that implements a distributed algorithm used to determine the temperature gradient map within a room.
- A remote monitoring and control console that enables remote access to sensors in one or more WSNs through a gateway that translates queries into native network messages sent to the desired node.

3 System requirements

Taking the previously discussed specificities of our network into account, we started by defining a set of basic requirements for the TagusPM solution:

- It has to provide a significant extension of network lifetime.
- It should not seriously affect existing applications, i.e. applications should not have to be rewritten to use its basic functionality.
- It should be easy to integrate with by applications that wish to take advantage of the full functionality.
- It should provide easy remote access to each node's energy level.

Considering these requirements and the general constraints posed by the used platform, we opted for a simple solution: controlling only the radio power state. While this choice may be seen as limiting, the radio is by far the largest energy consumer on a MICAz mote (Krämer and Geraldy, 2006), and, for a typical usage pattern, most of this energy is wasted on unnecessary idle listening.

There are two basic strategies for radio power management:

- *Asynchronous switching*, in which each node turns its radio on at a self-chosen time, independently of the nearby nodes, using fixed or variable intervals.
- *Synchronous cycling*, in which nodes switch their radios' power state at approximately the same time, with some (generally) fixed round period.

The asynchronous sleep mode can be used in the TinyOS 2.x platform provided LPL is turned on. In this mode, every node wakes periodically to check if there is another node with a message addressed to it. However, as the nodes are not synchronised, they wake up at different times forcing the sender to transmit long preambles to ensure that all receivers are ready to receive a message it wants to send (Polastre et al., 2004). Moreover, as soon as a receiver senses the preamble, it has to stay awake waiting for the upcoming message. Besides the overhearing inefficiency involved, the lack of synchronisation implies a larger radio power consumption on both receivers and transmitters.

Synchronous approaches tend to be more efficient, as a node meaning to transmit a message does not need to keep its radio on while waiting for the destination to wake up. In particular, a synchronous MAC layer, of which there are several implementations for TinyOS, could provide a more than satisfactory solution for this energy efficiency problem. However, such a single-layer mechanism, having its own duty-cycle rounds to cope with, hardly matches the application timings, naturally leading to some increase on buffering needs and latency.

On the other hand, the existing Tagus-SensorNet software framework already provided a time synchronisation service, used by several applications to perform round-based in-network data aggregation and processing. Thus, a cross-layer approach, integrating the application rounds with radio control procedures seemed to be the best option.

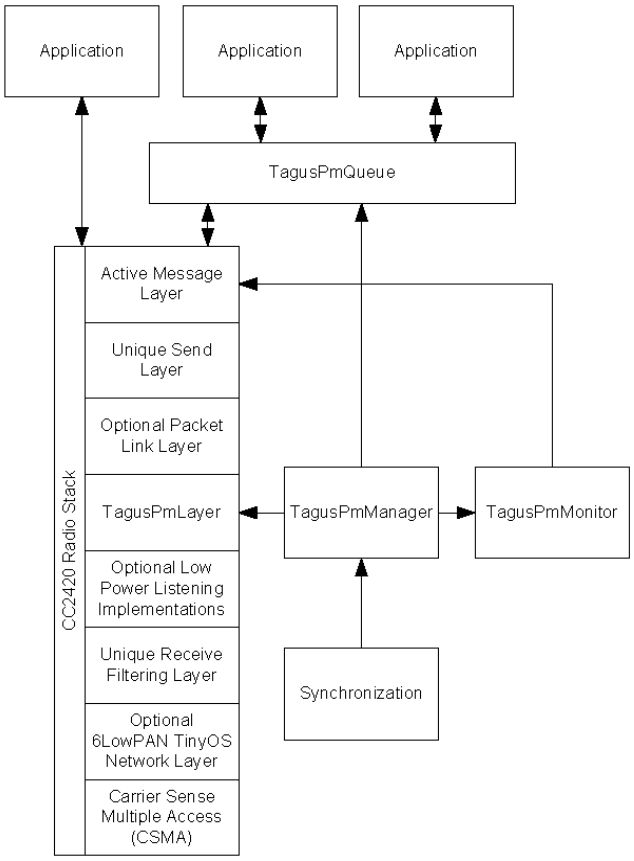
Our basic working model so far consisted of turning the radio on and off synchronously across the entire network. Then, in order to minimise the impact of the power management system on applications developed under the assumption of an always-on radio, the natural solution was to implement a message queuing system. Finally, the energy monitoring requirement was virtually separated from the rest of the functionality and could be freely implemented as an isolated component or application.

4 System architecture

With the initial decisions already made, we set out to draft a pluggable component model capable of achieving our energy-saving goal while still fulfilling the other requirements, namely easy integration and low impact on existing applications.

The first approach was centred on the insertion of a new layer on the CC2420 Radio (Moss et al., 2007b). This layer included not only the radio control logic, but also a common global queue for all applications. We soon found out that the interface contract for the message sending interfaces imposes a limit of a single pending message per sender (Levis, 2007), forcing us to extract the queuing functionality to an external component. The resulting architecture is shown in Figure 2.

Figure 2 First approach to the TagusPM system architecture



Under this architecture, the system was composed of the following components:

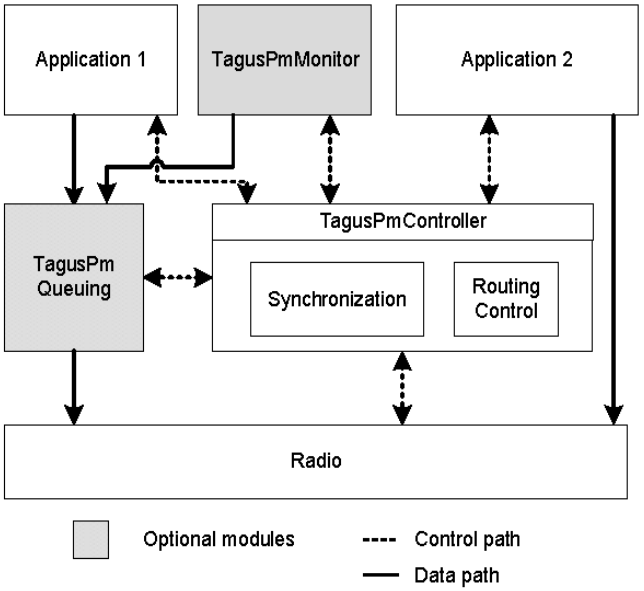
- *TagusPmManager*, which had the function of coordinating the other modules.
- *TagusPmLayer*, which was responsible for controlling the radio power state.
- *TagusPmMonitor*, which obtained and sent the energy readings.
- *TagusPmQueue*, which stood between the applications and the sending interface and implemented a common queue for the messages handled.

We quickly began to notice the disadvantages of this model: it was tied to the CC2420 stack, involved modifying TinyOS files and was extremely difficult to debug. Although inadequate, it did pave the way for a second, cleaner approach.

Using the lessons learnt in our first try, we designed a new architecture that did not require the use of a layer, or any other modification to the TinyOS core system. The result is illustrated in Figure 3, with the power management functionality being split into three separate modules:

- *TagusPM Controller*, responsible for controlling the entire system, keeping the system state and interacting with external services and applications.
- *TagusPM Monitor*, an entirely separate, application-level component that includes the battery-level monitoring functionality.
- *TagusPM Queuing*, a set of multi-instanced components that may be used by applications to buffer data during radio-off times.

Figure 3 Final TagusPM system architecture



The queuing and monitoring components are completely optional, with the core functionality being contained in the controller. This allows for some flexibility regarding resource usage, as not all applications and scenarios require the extra features, and memory is usually a scarce resource.

The queues were implemented as wrapper components for the sending interface, and are instantiated and used by each application independently. The adoption of individual queues brings a significant advantage over the previously proposed global solution, as it lets applications dimension their queues according to their real needs. Applications that choose not to use the queuing components, in order to spare memory or because they require *deterministic* message dispatching, are still able to subscribe to power management events, being notified every time the radio is turned on. This allows application to time their messages (or, in some cases, their full behaviour) as to not lose any data, even in the absence of queuing.

As for the synchronisation block, it is actually a simplified representation of a set composed by the FTSP synchronisation components and our own module that generates synchronous rounds from the time reference, and which will be described further ahead. The block referred to as Routing Control is the control interface for the routing protocol, which allows us to manually send route discovery messages at will.

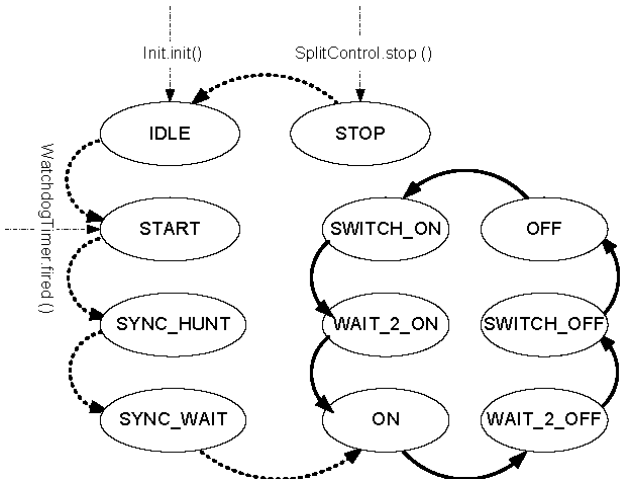
5 Design and implementation

Once the system architecture was defined, a prototype was designed and implemented. While parts of the implementation were straightforward conversions of the architecture, some aspects required careful thought and consideration. Over the next sub-sections, we will discuss some of the most relevant, including the system core, synchronous rounds component, the message queuing implementation and the monitoring functionality.

5.1 TagusPmController

The TagusPmController component assumes a central role in the system, encompassing all the logic and controlling the remaining modules and services, including the radio, the synchronisation mechanisms and the routing messages. Its implementation is small, spanning just over 300 lines of nesC code, but its development required some care, as it deals with a relatively complex state machine, shown in Figure 4.

Figure 4 State diagram for the TagusPmController module



The state diagram is divided into two parts:

- On the right, with transitions in full lines, are the states corresponding to the stable regime, i.e. in normal operation the system is synchronised and cycles between these states with a defined period.
- On the left, with transitions in dashed lines, are the special states, used when the system is activated or deactivated, as well as when it is still synchronising.

For clarity, the events leading to the state transitions were omitted from the diagram. The three labelled events correspond to external interventions: either by other modules, such as applications or framework components, or, in the case of *WatchdogTimer.fired()*, by the synchronisation watchdog, whose function is to reset the system if it loses a correct time reference. In the absence of this watchdog, it would be possible for a node to fall out of synchronisation with the rest of the network, never to be able to communicate again. The remaining transitions are triggered by timers, callbacks from split control interfaces or signals from the synchronous rounds generator. A more detailed description of each state is tabulated in Table 1.

Table 1 Explanation of TagusPmController states

	State	*Description*
Normal behaviour	ON	Radio on, messages flowing freely.
	OFF	Radio off, no radio traffic.
	WAIT_2_ON	Radio just turned on, waiting for sufficient time to guarantee all nodes activate their radios.
	WAIT_2_OFF	Radio on but about to be turned off, finishing transmission of pending messages.
	SWITCH_ON	Radio off, but request to turn on already sent to the stack.
	SWITCH_OFF	Radio on, but request to turn off already sent to the stack.
Other states	IDLE	System disabled, either not yet started or already stopped.
	START	System currently being enabled, as a consequence of an external function call.
	STOP	System currently being disabled, as a consequence of an external function call.
	SYNC_HUNT	System enabled, radio on, waiting to acquire a time reference.
	SYNC_WAIT	System enabled, radio on, acquired a time reference, waiting to validate it.

In a normal situation, the following sequence of events takes place:

- The node is powered on and the system is loaded in the IDLE state.
- The application or the OS call *Init.init()*, placing the system into state START. The system then initialises the routing algorithm, starts the watchdog timer and asks for the radio to be turned on. When the radio comes up (or if it already was), the system moves to state SYNC_HUNT and configures the synchronisation module to send periodic beacons.
- The acquisition of a time reference signals the system, which moves to state SYNC_WAIT and starts a timer, in order to allow global time to stabilise before beginning radio cycling.
- The firing of this timer means that the system is ready to enter normal operation. The state is set to ON, the synchronisation module is set to manual control and a timer is started. After T_{round}, the timer fires and the system enters the WAIT_2_OFF state, starting another

timer. After a brief delay, the timer fires, causing the system to go into state SWITCH_OFF and request the radio to be turned off. When the radio callback is received, the system moves to state OFF.

- A generally similar process occurs when a new round is signalled. First, the system is moved into state SWITCH_ON, while waiting for the radio to come up, and then to WAIT_2_ON, which has a small random delay added in order to reduce collisions. Finally, the system sets the ON state, starts a timer with T_{round}, triggers the sending of the necessary control messages, and, after a brief delay, alerts queues and applications that the radio is available. From then on, it continues the cycle already described.

5.2 Round synchronisation

The synchronisation reference for the TagusPM system is provided by the Flooding Time Synchronisation Protocol (FTSP) (Maróti et al., 2004) implementation supplied in TinyOS and used in our software framework. Oversimplifying, this protocol establishes a synchronisation tree, propagating the time reference from the root to the leaves. This time reference is a single integer counter, separate from the local clock, and valued in milliseconds.

The value, as it is, is useful as a way to timestamp events. For our purpose, however, we need to convert this reference into a synchronous alarm at all nodes. We created a component, RoundSyncC, which uses a set of one-shot timers to achieve this transformation.

FTSP had been in use in Tagus-SensorNet for two years, showing good performance – this was still while running TinyOS 1.x. The shift towards TinyOS 2.x and the new implementation was actually concurrent with the development of TagusPM, and it was assumed the system would perform just as well – an assumption that turned out to be wrong. While there were no major problems (just some infrequent synchronisation losses) during our lab evaluation, on deployment to the full testbed we found that the nodes never managed to get a stable reference, dropping out so often that the system seldom got past the SYNC_WAIT state. Worse, sometimes FTSP acquired an invalid reference but did not become aware of it, leading to long periods in which nodes had their rounds triggered at different times, and could not listen to other nodes' synchronisation broadcasts. The same problems were later detected in the network even in the absence of TagusPM, although the critical role FTSP plays in our system caused it to be much more obvious. Further investigation allowed the linking of this behaviour to a bug report on the TinyOS website, dealing with the implementation of low-level packet times stamps, as of yet unfixed (see http://docs.tinyos.net/index.php/PacketTimeStamp_CC2420_bug).

5.3 Queuing

What shows up in Figure 3 as TagusPmQueuing is, in reality, a set of two components, called QueuedSender and QueuedAMSender. As previously stated, these serve as wrapper components for the native Sender and AMSender, exposing the same interface but buffering the messages they receive.

These components do not, in fact, entirely comply with their interface specifications, as they do not enforce the existence of a single pending message. While this is not an elegant solution, the behaviour is fully intended and documented. Our goal while designing the queuing modules was to keep the necessary adaptations to existing applications to a bare minimum. This way, and considering applications should not be counting on a reliable channel to start with, we are able to reduce required changes to only two lines, thereby decreasing the adaptation cost.

5.4 Monitoring

The monitoring functionality was implemented as a separate module that, while conceptually a service, is in practice an application. It periodically collects battery-level information, using the built-in *sensor*, creates a message and sends it to the management console, through the multi-hop 'Collection' protocol (Fonseca et al., 2006).

On reception, these messages are parsed and the battery values are updated on a table featuring all of the nodes, allowing the operator to check the energy status of the whole network at a glance. In the future, we expect the system to collect additional information provided by the hardware currently being developed.

6 Evaluation

In order to ascertain the real impact of our system, some evaluation tests were conducted. First, we had to determine the real energy usage under normal operating conditions, with the radio on and off. The values presented in Table 2 were obtained from a MICAz node, running a simple application that constantly collects and sends light sensor readings (somewhat of a worst-case scenario for power management), and are an average of 10 minutes of continuous measurement.

Table 2 Average power depending on power state

Radio state	*Average Power (mW)*
On	76.42
Off	8.33

Looking at Table 2, we can see that there is a relevant savings potential, the energy consumption with the radio on being approximately nine times higher. It should be noted that these values are dependent on the behaviour of each application, especially the energy consumption with the radio powered off. Its value can exhibit extreme variations depending on the fraction of time the microcontroller spends in its low-power states, as well as on how the application uses the remaining hardware: a single LED, for instance, can require up to 2.5 mA (Krämer and Geraldy, 2006).

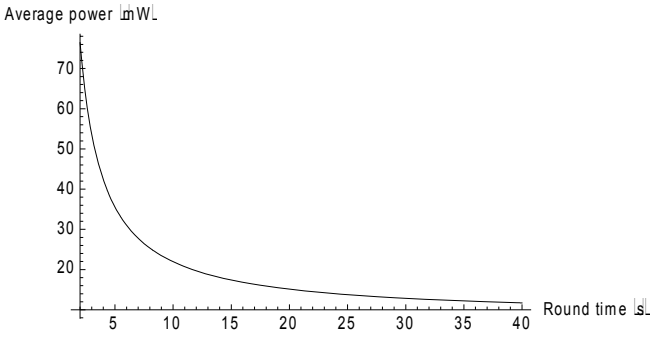
We then proceeded to quantify the expected average power (1) and power saving factor (2), which is also an estimate of the network lifetime extension factor. In both formulae, P_{off} and P_{on} refer to the average power with the radio respectively off and on, T_{round} is the round time and T_{on} measures the time the radio is on in each round (it is therefore included in the round time).

$$P_{avg} = \frac{P_{off}\left(T_{round} - T_{on}\right) + P_{on}\left(T_{on}\right)}{T_{round}} \quad (1)$$

$$r = \frac{P_{on}T_{round}}{P_{off}\left(T_{round} - T_{on}\right) + P_{on}T_{on}} \quad (2)$$

We then plotted (1), considering the power values from Table 2, as well as a T_{on} value of 2 seconds, which, we believe, provides a good balance between the needs of the test application, the wish for a short on time and the reduction of wasted energy on the on/off guard times. Figure 5 shows, as expected, that average power decreases exponentially as a function of the round time.

Figure 5 Impact of round time on energy consumption, for a fixed radio on time of 2 seconds



The choice of round time depends on the exact applications, as it must consider both the acceptable delay and the desired energy savings. For our application, we decided to use a round time of 10 seconds, and performed real-life tests to validate both the system and our expected power reductions. The results of one such test, consisting of a 60 minutes long measurement of a node's average power, can be seen in Table 3.

Table 3 Predicted and measured values for our test setting

Parameter	*Value*
P_{avg} (predicted)	21.95 mW
P_{avg} (measured)	22.17 mW
r (measured)	3.447
Duty cycle	20%

Even when using these admittedly conservative settings, we achieve a network lifetime of approximately 3.5x the one we had before. Further tests showed that, with these times and adequate queue sizes, there was no change in the packet loss rate when using TagusPM. It is, however, important to note that these measurements refer to an always-on application with a high sampling rate, in which there are high data rates, short CPU sleep times and low potential for energy saving. In a less demanding application, not only could the duty cycle be reduced (equivalent to the round time being increased), but, thanks to the TinyOS microcontroller power management subsystem, the average power for the radio off situation would also be lower, leading to a steep decrease in energy consumption.

7 Related work

With energy being a topic of critical importance in WSNs, there are many previously proposed solutions that try to reduce energy consumption in some way. In this section, we will briefly discuss some representative approaches.

7.1 Mac-based approaches

MAC protocols are the prime candidates for energy saving in WSNs, as they operate closest to the radio. Duty-cycled MAC protocols can generally be divided into synchronous and asynchronous, although hybrid variations exist too.

Asynchronous approaches rely on preamble sampling, otherwise known as LPL. Nodes wishing to send a message start transmitting a long preamble. All other nodes periodically sample the channel, and, when hearing the beacon, stay awake waiting for the message. B-MAC (Polastre et al., 2004), used by the CC2420 radio in the MICAz nodes, is an example of such a protocol. Because they are purely asynchronous, they avoid the communication and processing overhead of scheduling and synchronisation. WiseMAC (Enz et al., 2004) is another example. It uses non-persistent CSMA with preamble sampling to reduce idle listening. While in a basic LPL implementation this preamble should be the same length as the radio off time (common to all nodes), WiseMAC offers a method to dynamically resize the preamble according to the sleep schedules of the neighbouring nodes, reducing over-emitting energy waste. X-MAC (Buettner et al., 2006) introduces new ideas, namely the embedding of the target node's ID in the preamble, preventing other nodes from having to wake up, and the use of strobed preambles, allowing early interruption by the receiving node.

Synchronous approaches, on the other hand, work by negotiating a common schedule in which neighbouring nodes wake up to exchange messages. In S-MAC (Ye et al., 2004), neighbouring nodes form virtual clusters that use a common sleep schedule, i.e. all nodes in a cluster wake up and exchange messages at the same time. Nodes belonging to more than one cluster wake up at each cluster's listening times. T-MAC (van Dam and Langendoen, 2003) improves

S-MAC by making the duty cycle adaptive and increases efficiency by grouping messages and transmitting them in bursts.

Hybrid protocols take many forms, and a general description is not easy to accomplish. They do, however, share features of the two classes in which they are based. SCP (Ye et al., 2006) is one such protocol. It can be described as *synchronised LPL*, in the sense that it uses the preamble sampling technique (characteristic of LPL protocols) but with synchronised sampling times.

7.2 Routing-based approaches

Energy usage is also a concern in the design of most WSN routing protocols. The protocol proposed in Shah and Rabaey (2002) builds a routing table with the reception and transmission costs, as well as the residual energy of each node. Instead of using the minimum energy path, each route is assigned a selection probability – in this way, a set of sub-optimal paths is used, preventing rapid depletion of nodes along the best path.

PEGASIS (Lindsey and Raghavendra, 2002) forms nodes chains through which messages are routed, aggregating data along the way. The protocol manages to replace long-distance and high-power transmissions to the sink with short-distance transmissions between nodes, and a single transmission of aggregated data to the sink, made by a randomly chosen node.

TEEN (Manjeshwar and Agrawal, 2001) uses a hierarchical model with multi-level clusters. Cluster heads broadcast threshold values that limit the situations in which nodes transmit data. In this way, the number of messages is reduced, thereby also reducing energy consumption.

In Rodoplu and Meng (1999), a geographic routing approach is presented, in which a low-power GPS receiver is used to construct a sparse graph of globally optimal links in terms of energy consumption, which is then fed to a Belmann-Ford shortest path algorithm.

7.3 Cross-layer approaches

Most cross-layer approaches work on a combination of MAC and routing techniques, achieving what is expected to be better efficiency by taking advantage of extra information on the network topology and communication needs. One such approach is presented in Jurdak et al. (2007), whose authors built a framework that optimised power usage through greedy local decisions based on local and neighbourhood state information. The system adapts the behaviour of both the routing and MAC layers, the latter being based on B-MAC.

A different solution is discussed in Xing et al. (2009), where a framework is presented that integrates transmission power control and sleep scheduling into a so-called *Minimum Power Configuration*, further enhanced by a unified cross-layer architecture that allows coordination between different power management strategies.

8 Conclusions and future work

The developed system, TagusPM, was able to considerably reduce energy consumption and extend the network lifetime, as shown on the course of our evaluation. The ability to monitor the battery status can also aid network operators in their task, allowing them to know which nodes need replacement, instead of having to check each and every one or wait for them to run out of energy. It presents a significant contribution to the future of Tagus-SensorNet, increasing the network availability and reducing the maintenance effort.

We also came to the conclusion that, in order to develop an efficient power management solution, it is critically important to follow a cross-layer approach, in our case involving the applications, the power management system, the synchronisation system and, to a lesser extent, the routing protocol, with which we only interact to guarantee that route discovery messages are only sent when the radio is on. It is, however, possible to improve this solution by integrating new energy-aware MAC and routing protocols, causing the message flow to take into account the nodes' energy state.

Unfortunately, our efforts fell short of a real-world deployment, frustrated by the lack of a mature time synchronisation implementation in TinyOS 2.1. While we expect the current implementation to be fixed, and there are others in the works, we have already started the development of a new time synchronisation protocol at IST, which we hope to deploy in the near future.

Although our current solution fulfils our initial requirements, energy saving in sensor networks is a very broad and open research area and, using TagusPM as the basis, much can still be done. As part of our future work we intend to tackle the following issues:

- The definition of duty cycling parameters at run time, either automatically or by an operator's request. As duty cycling coherence is critical for the nodes to be able to communicate, this should involve the use of some transactional semantics (e.g. two-phase commit), in order to guarantee simultaneous switchover of all nodes.
- The usage of different round times in each node, namely multiples of a base round time T_r. Methods should be found to dynamically choose these parameters, according not only to internal needs but also to the network load.

- The expansion of the monitoring interface to accommodate calculation of derived measurements, prediction of battery replacement times and notification of the operators.
- The interaction between our system and the LPL layer distributed with the TinyOS CC2420 stack, as the concurrent usage of both could lead to further reductions on the energy consumption.
- The evaluation of the system's performance while deployed on the full network and running its typical applications instead of our worst-case scenario. This is dependent on the time stamping bug being fixed, as the current system is not stable enough to perform such tests.

Acknowledgements

This work was supported by Instituto de Telecomunicações and Instituto Superior Técnico, Technical University of Lisbon. We wish to thank our colleagues at the Group of Embedded Networked Systems and Heterogeneous Networks (GEMS) for their input, and especially Luís Pedrosa for the precious help and advice given over the entire course of this project.

References

Akkaya, K. and Younis, M. (2003) 'A survey on routing protocols for wireless sensor networks', *Ad Hoc Networks*, Vol. 3, No. 3, pp.325–349.

Buettner, M., Yee, G.V., Anderson, E. and Han, R. (2006) 'X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks', *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, SenSys'06, University of Colorado, Boulder, CO, USA.

Crossbow Technology (2004a) 'MICA2 wireless measurement system', *Datasheet*, Crossbow Technology.

Crossbow Technology (2004b) 'MICAz wireless measurement system', *Datasheet*, Crossbow Technology.

Demirkol, I., Ersoy, C. and Alagöz, F. (2006) 'MAC protocols for wireless sensor networks: a survey', *IEEE Communications Magazine*, Vol. 44, pp.115–121.

Enz, C.C., El-Hoiydi, A., Decotignie, J-D. and Peiris, V. (2004) 'WiseNET: an ultralow-power wireless sensor network solution', *IEEE Computer*, Vol. 37, No. 8, pp.62–70.

Fonseca, R., Gnawali, O., Jamieson, K. and Levis, P. (2006) 'Collection', *TEP 119*, TinyOS Net2 Working Group.

Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E. and Culler, D. (2003) 'The nesC language: a holistic approach to networked embedded systems', *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation*, San Diego, CA, USA, pp.1–11.

Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, S.L. and Rubenstein, D. (2002) 'Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet', *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, San Jose, CA, USA, pp.96–107.

Jurdak, R., Baldi, P. and Lopes, C.V. (2007) 'Adaptive low power listening for wireless sensor networks', *IEEE Transactions on Mobile Computing*, Vol. 6, No. 8, pp.988–1004.

Krämer, M. and Geraldy, A. (2006) *Energy Measurements for MicaZ Node*, Technical Report KrGe06, University of Kaiserslautern, Kaiserslautern, Germany.

Levis, P. (2007) 'Packet protocols', *TEP 116*, TinyOS Core Working Group.

Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E. and Culler, D. (2005) 'TinyOS: an operating system for sensor networks', in Weber, W., Rabaey, J.M. and Aarts, E. (Eds): *Ambient Intelligence*, Springer Berlin Heidelberg, New York, pp.115–148.

Lindsey, S. and Raghavendra, C.S. (2002) 'PEGASIS: power-efficient gathering in sensor information systems', *Proceedings of the IEEE Aerospace Conference*, Big Sky, Montana, USA, pp.1125–1130.

Manjeshwar, A. and Agrawal, D.P. (2001) 'TEEN: a routing protocol for enhanced efficiency in wireless sensor networks', *Proceedings of the 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, San Francisco, CA, USA.

Maróti, M., Kusy, B., Simon, G. and Lédeczi, Á. (2004) 'The flooding time synchronization protocol', *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys'04*, Baltimore, MD, USA, pp.39–49.

Moss, D., Hui, J. and Klues, K. (2007a) 'Low power listening', *TEP 105*, TinyOS Core Working Group.

Moss, D., Hui, J., Levis, P. and Choi, J.I. (2007b) 'CC2420 radio stack', *TEP 126*, TinyOS Core Working Group.

Pedrosa, L.D., Melo, P., Rocha, R.M. and Neves, R. (2009) 'A flexible approach to WSN deployment', *Proceedings of 17th International Conference on Computer Communications and Networks, ICCCN'08*, St. Thomas, VI, USA.

Polastre, J., Hill, J. and Culler, D. (2004) 'Versatile low power media access for wireless sensor networks', *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys'04*, Baltimore, MD, USA, pp.95–107.

Rodoplu, V. and Meng, T.H. (1999) 'Minimum energy mobile wireless networks', *IEEE Journal of Selected Areas in Communications*, Vol. 17, No. 8, pp.1333–1344.

Shah, R.C. and Rabaey, J.M. (2002) 'Energy aware routing for low energy ad hoc sensor networks', *Proceedings of the IEEE Wireless Communications and Networking Conference*, WCNC, Orlando, FL, USA.

Smith, R.B. (2007) 'SPOTWorld and the Sun SPOT', *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, Cambridge, MA, USA, pp.565–566.

van Dam, T. and Langendoen, K. (2003) 'An adaptive energy-efficient MAC protocol for wireless sensor networks', *Proceedings of the First ACM Conference on Embedded Network Systems, SenSys'03*, Los Angeles, CA, USA.

Xing, G., Sha, M., Hackmann, G., Klues, K., Chipara, O. and Lu, C. (2009) 'Towards unified radio power management for wireless sensor networks', *Wireless Communications and Mobile Computing*, Vol. 9, No. 3, pp.313–323.

Ye, W., Heidemann, J. and Estrin, D. (2004) 'Medium access control with coordinated adaptive sleeping for wireless sensor networks', *IEEE/ACM Transactions on Networking*, Vol. 12, No. 3, pp.493–506.

Ye, W., Silva, F. and Heidemann, J. (2006) 'Ultra-low duty cycle MAC with scheduled channel polling', *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, SenSys'06*, University of Colorado, Boulder, CO, USA.

Zhang, P., Sadler, C.M., Lyon, S.A. and Martonosi, M. (2004) 'Hardware design experiences in ZebraNet', *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, Baltimore, MD, USA, pp.227–238.