Cross-layer routing and time synchronisation in Wireless Sensor Networks

Szymon Fedor

Ericsson Ireland,
Ericsson Software Campus,
Athlone, Ireland
szymon.fedor@ericsson.com

Martin Collier

School of Electronic Engineering,
Dublin City University,
Dublin 9, Ireland
E-mail: collierm@eeng.dcu.ie

Cormac J. Sreenan

Department of Computer Science,
University College Cork,
Cork, Ireland
E-mail: cjs@cs.ucc.ie

Abstract: A common time reference across nodes is required in most Wireless Sensor Networks (WSNs) applications. It is needed, for example, to time-stamp sensor samples and for long-term duty cycling of nodes. Also many routing protocols require that nodes communicate according to some predefined schedule for reasons of energy efficiency. However, independent distribution of the time information, without considering the routing algorithm schedule or network topology may lead to a failure of the synchronisation protocol. This was confirmed empirically, and was shown to result in loss of connectivity. This can be avoided by integrating the synchronisation service into the network layer with a so-called cross-layer approach. This approach introduces interactions between the layers of a conventional layered network stack, so that the routing layer may share information with other layers. We explore whether energy efficiency can be enhanced through the use of cross-layer optimisations and present two novel cross-layer routing algorithms. The first protocol, designed for hierarchical, cluster based networks and called CLEAR (Cross Layer Efficient Architecture for Routing), uses the routing algorithm to distribute time information which can be used for efficient duty cycling of nodes. The second method - called RISS (Routing Integrated Synchronization Service) - integrates time synchronization into the network layer and is designed to work well in flat, non-hierarchical network topologies.

We implemented and tested the performance of these solutions in simulations and also deployed these routing techniques on sensor nodes using TinyOS. We compared the average power consumption of the nodes and the precision of time synchronization with the corresponding parameters of a number of existing algorithms. All proposed schemes extend the network lifetime and due to their lightweight architecture they are very efficient on WSN nodes with constrained resources. Hence it is recommended that a cross-layer approach should be a feature of any routing algorithm for WSNs.

Keywords: Sensor networks, time synchronization, routing, cross-layer

Reference to this paper should be made as follows: Fedor, S., Collier, M. and Sreenan, J.C. (2010) 'Cross-layer routing and time synchronisation in Wireless Sensor Networks', *Int. J. of Sensor Networks*

Biographical notes: Szymon Fedor holds a M.Sc. degree from INSA Lyon and Dublin City University. He obtained a PhD from DCU in 2008 for research on the energy optimization of Wireless Sensor Networks. After PhD he has been doing research in the domain of WSN in Ericsson Ireland. He submitted 5 patent proposals and published a book, a book chapter and 6 papers in peer-reviewed conferences.

Dr. Martin Collier is the director of the Network Innovations Centre in Dublin City University, Dublin, Ireland. His research interests include telecommunications switching systems, sensor networks, network protocols and network security.

Cormac J. Sreenan, is a Full Professor of Computer Science at UCC and Head of Department from 2000-2004 inclusive. Prior to joining UCC in August 1999, he was a researcher at AT&T Labs in Florham Park, New Jersey and Lucent Bell Labs in Murray Hill, New Jersey. He holds a PhD in Computer Science from the University of Cambridge. He has over 80 peer-reviewed publications and 7 patents granted. He is active on the technical program committees of many well-established conferences and workshops.

1 Need for synchronisation in Wireless Sensor Networks

The time synchronisation problem is a standard problem in distributed systems, but especially in Wireless Sensor Networks (WSNs). We can distinguish three main domains where synchronised clocks are required: at the interface between the sensor network and an external observer, among the nodes of the sensor network, and at the interface between the sensor network and the observed physical world.

In many applications, a sensor network interfaces to an external observer (a human operator or an autonomous computing system) for tasking, reporting results, and management. Tasking a sensor network often involves the specification of time windows of interest such as "only during the day". As a sensor network reports observation results to an external observer, the temporal properties of observed physical phenomena may be of interest. For example, the time-stamping of the samples gathered by sensors or times of occurrence of physical events are often crucial for the observer. In these scenarios nodes should be synchronised to an absolute time reference.

Time synchronisation is also required for intra-network coordination among different sensor nodes Karl and Willig (2007); Wang and Wang (2007); Faizulkhakov (2007). This includes security (*e.g.* authentication), data consistency (*e.g.* cache consistency, consistency of replicated data), concurrency control (*e.g.* atomicity, mutual exclusion such as TDMA) Macedo et al. (2009), and communication protocols (*e.g.* at-most-once message delivery). In these applications nodes can be synchronised to an absolute time value or they can estimate a relative time to abstract time reference (*e.g.* neighbour's local clock). The energy efficiency of WSN can be improved by frequently switching sensor nodes or components thereof into power-saving sleep modes if the energy saved during the sleep mode compensates the additional energy dissipation due to putting into sleep and to awaking the node. In order to ensure seamless operation of the sensor network, temporal coordination of the sleep periods among sensor nodes may be required. This is called a *transmitter-receiver rendezvous problem* Anastasi et al. (2009). Also, many data-fusion[1] algorithms have to process sensor samples in order of their time of occurrence. However, sensor networks may suffer from highly variable message delays and thus messages from distributed sensor nodes may often not arrive at a receiver in the order in which they were sent. These can be reordered according to the time of sensor readout only when nodes are synchronised. Methods for localisation of sensor nodes based on the measurement of time of flight or difference of arrival time of certain signals also require synchronised time.

In the third group of applications where node synchronisation is required we can distinguish data fusion which may extract higher-level information about an observed object (*e.g.* its size, speed, or shape) by correlating measurements from multiple locations. Also, sometimes one of the tasks of a WSN is to partition sensor samples into groups (that each represent a single physical phenomenon) when many instances of a physical phenomenon occur within a short time. Then the temporal relationships among sensor measurements are a key factor in performing this separation. In these scenarios only a relative time synchronisation is needed.

All these different requirements for time synchronisation have led to the development of many synchronisation protocols. The next section describes the main difficulties in designing such protocols and describes a representative synchronisation algorithm that we used as reference to estimate performance of our solutions described in sections 4 and 5. Section 6 compares the proposed solutions, their scope and application scenarios and concludes the paper.

2 WSN synchronisation difficulties

Clock synchronisation algorithms face two problems: time-stamping jitter caused by delays in transmitting a packet and time errors due to the operating differences of hardware (a so-called clock drift[2]). Time-stamping jitter is a result of a variance of time interval between inserting a time-stamp into the packet by the sender and reading of it at the receiver. The source of message time-stamping inaccuracy was analysed by Maroti *et al.* Maróti et al. (2004). It includes sending, channel access, transmission, propagation, reception and receive time uncertainty. These errors are random and difficult to predict and eliminate especially in a network with many communicating nodes. Clock drifts are difficult to overcome and they can vary with time and temperature. One of the methods to compensate clock drifts requires a periodic updating of the time information between nodes.

A popular synchronisation technique for WSNs is the Flooding Time Synchronisation Protocol (FTSP) described by Maroti *et al.* Maróti et al. (2004). It utilises periodic flooding messages containing current local time information and originated at the elected coordinate, which is the node with the lowest ID. Upon receiving this packet a node records the contained time-stamp and the time of arrival, and broadcasts the message to its neighbours after updating the time-stamp.

There are many other protocols in the literature that address the synchornisation problem Karl and Willig (2007); Lasassmeh and Conrad (2010) in WSNs. Most of them were designed with an assumption that the network stack architecture compiles with the traditional layered hierarchy. As a result they operate as individual entities located at the application layer of the network stack which use services of lower layers (e.g. physical or routing layer) and provide time information to other applications. These solutions are efficient in terms of synchronisation precision but in WSNs we have also to consider the power cost of a protocol. Energy efficiency of WSN nodes can be improved by a using a so-called cross-layer design approach instead of a traditional layered architecture. In next section I will compare these two approaches and give more details about problems that may occur while using traditional layered hierarchy of the network stack in WSNs.

3 Problems with traditional approach to synchronisation in WSN

3.1 Layered vs cross-layer protocol architecture

Most of the proposed communication protocols for WSN follow the traditional layered protocol architectures. More specifically, the majority of these communication protocols are individually developed for different networking layers, i.e., physical, medium access control (MAC), network, and transport layers. While these protocols may achieve very high performance in terms of the metrics related to each of these individual layers, they are not jointly optimized in order to maximize the overall network performance while minimizing the energy expenditure. Considering the scarce energy and processing resources of WSN, joint optimization and design of networking layers, i.e., cross-layer design, stands as the most promising alternative to inefficient traditional layered protocol architectures. Cross-layer design refers to protocol design done by actively exploiting the dependence between protocol layers to obtain performance gains.

Cross-layer design can be realised in different ways. It may consist of combining information and functionalities of several traditional communication layers into a single protocol or defining dependence of one protocol's actions on the actions of protocols from other layers. It generally decreases the level of modularity, but provides highly reliable communication with minimal energy consumption which is very challenging in the context of severe energy and processing constraints of wireless sensor nodes.

The layered architecture may be unduly restrictive in the WSN context, where packets are often routed by a cluster-based protocol. The autonomous operation of the network stack layers can lead to a significant drop of communication performance. For example we observed that the LEACH routing scheme Heinzelman et al. (2000) may cause a failure of the synchronisation protocol when it puts the node into low power mode. As a consequence, the WSN stops delivering measurements to the base station. Another issue that we identified was the high packet loss observed with the same routing scheme. This problem has two origins: the harshness of the communications environment and hardware limits of the nodes (which may be exacerbated by different configurations of LEACH). These problems can be encountered in a WSN when the routing is performed by a cluster-based protocol using duty-cycling of the nodes. We demonstrated these shortcommings using the LEACH protocol which is one of the most widely used hierarchical routing schemes in WSNs. The analysis of these problems are presented in next section.

3.2 Intra-cluster communication problem

Energy resources are the main constraint of a WSN. The power being drained from the nodes can be reduced by arranging sensor communication with a schedule, turning off the transmitter when data sending does not occur. This solution was implemented in many protocols, for example in LEACH where subordinate nodes communicate with cluster heads according to a TDMA scheme. However, in order to identify the allocated communication time slots, the sensor node clocks must be synchronised. The authors of LEACH propose to implement a separate synchronisation protocol and then share time information with the routing scheme. In our experiments this task was performed by the Flooding Time Synchronisation Protocol (FTSP) Maróti et al. (2004). The use of a synchronisation scheme independent of the routing protocol caused some of the problems we encountered with a deployed WSN.

We carried out all the described experiments with a 10 node network of sensor nodes uniformly distributed in a grid topology with a minimal inter-node distance of 5 m. We employed a commonly-used sensor network platform; namely the Tmote Sky sensor node Polastre et al. (2005), and the networking stack as implemented in TinyOS Levis and Gay (2009). Its networking stack includes a default physical layer that supports single-bit error correction and double bit error detection capabilities. On top of this, its default MAC layer (LPL) implements a simple CSMA/CA scheme. LEACH uses a CSMA/CA protocol in the setup phase when nodes randomly access the channel to elect cluster-heads which then create TDMA schedule for the subordinate nodes. Then the nodes communicate according to the TDMA scheme with their cluster-heads.

For test purposes we implemented the LEACH protocol in the nesC programming language. There was no available implementation of LEACH on real WSN nodes. LEACH requires network synchronisation and the authors of LEACH do not specify how it should be performed. Therefore we chose FTSP for this task because it is one of the most popular synchronisation protocols and also an implementation on TinyOS is provided by the authors. We also installed a simple application on the nodes which periodically collects humidity and temperature measurements from sensors integrated on the sensor nodes. This data is then routed with LEACH to the base station which is a Tmote Sky sensor node connected to a laptop with an application for processing measurements from WSN. To test the routing scheme each node records the number of generated packets, whereas the base station stores the number of received messages. At the end of the experiment sensor nodes send data reports which are compared with the base station summary. These measurements are then examined for a quantitative study of the protocol performance.

First a series of tests demonstrated the data delivery efficiency of LEACH depending on two key parameters: the probability of becoming a cluster-head and the frequency of data packets. There may be many reasons for communication failure. First of all, interference can be caused by the signals sent by nodes from other clusters. Also the MAC layer does not prevent the hidden terminal problem. Finally, the computational resources of the nodes are too small to accept high rate and long bursts of data. In consequence, if the receiver's processor is performing a task and the reception buffer or task queue is full, the message is discarded. This situation is unacceptable in applications where packet delivery must be guaranteed (*e.g.* intruder detection or medical applications).

Figure 1 shows how the cluster-head election process influences the packets delivery rate to the base station. In these experiments we collected packet failure delivery statistics as described above. We changed the probability of becoming a cluster-head (one of LEACH parameters) for every test round of 6 hours. Sensors were taking measurements every 10 seconds and sending 3 packets per minute. The performance of LEACH drops with increasing probability of becoming a cluster-head. A higher density of cluster-heads increases the number of nodes trying to communicate with the base station. In consequence, the hidden terminal problem occurs more often and the base station will also have problems handling many signals arriving at the one time.

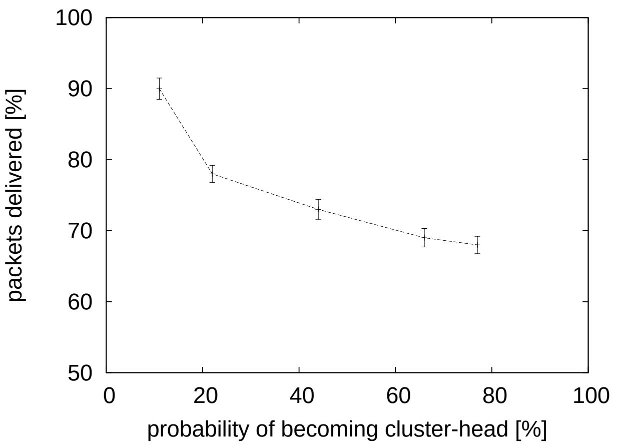

Figure 1 Data delivery with LEACH protocol as a function of the probability of becoming cluster-head (95% confidence interval).

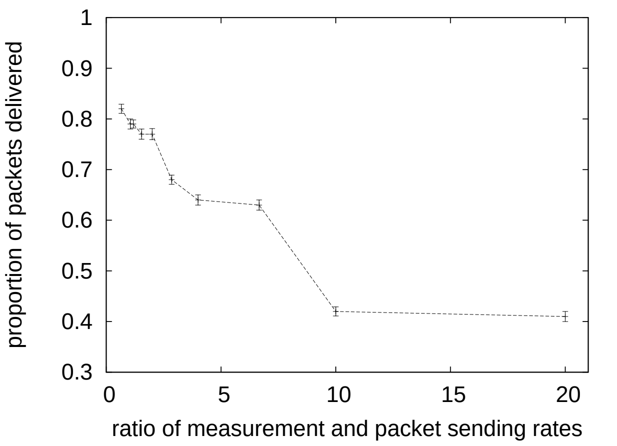

Figure 2 Data delivery with LEACH protocol as a function of the packet sending rate.

We also observed that packet loss depends on the ratio of sensor measurement and data sending rates. Subordinate nodes can only communicate in the time slot assigned by the cluster-heads. In consequence, sensor measurements that cannot be sent have to be buffered until the next transmission. During the series of experiments described below, the cluster-head election probability was arbitrarily chosen to be 0.2 and packet delivery statistics are obtained by comparing the number of packets sent by every sensor node with the number of packets received by the base station. This time sensor nodes were sending packets with a constant rate of three packets per minute but for every experiment round of 6 hours we were varying the frequency of collecting sensor measurement. As can be seen in Figure 2, if more samples have to be stored before being sent then more packets are dropped before reaching the base station. This degradation of the overall performance can be explained as follows. The channel access protocols cannot sustain long bursts of messages, even if the total data rate is relatively low. We conclude that a too long sleeping period can have negative consequences on the network lifetime. Nodes turn off transceiver periodically in order to save energy. Then sensor samples are saved and sent in a longer data burst when the node wakes up. When the sleeping period increases, data delivery performance worsens. Lost packets have to be resent which results in additional energy drain from the sensor node. So some of the energy saved in a low power mode is consumed when lost packets are resent.

Another issue that we observed during experiments was the loss of synchronisation due to the duty cycle of the subordinate nodes. With FTSP, each sensor node sends periodic synchronisation messages and if LEACH initiates sleeping mode for subordinate sensor nodes, they cannot receive those packets and quickly become unsynchronised. That is why in our experiments with LEACH, low power mode was disabled.

We address these encountered problems in two new cross-layer communication protocols. They are designed with four goals:

- to minimise the overhead of the synchronisation protocol
- to minimise the awake time of the transceiver and in consequence its energy dissipation
- to achieve good time synchronisation
- to achieve robust protocol performance across a range of values for the duty cycle of the nodes.

To achieve them we propose to use a so-called cross-layer design approach where the traditional layered protocol architecture is violated by cutting across traditional layer boundaries with the purpose of performance optimisation, resource preservation or error resilience. Compared to the traditional layered protocol architecture, the network synchronisation can be obtained in a more energy efficient way when the synchronisation service is integrated in the routing layer. Several time synchronisation protocols which use cross-layer information sharing between Network, MAC and PHY layers have been proposed Karl and Willig (2007); Miao et al. (2009). In those scenarios, the advantage of granularity of the network stack is still retained. However, time information estimated by one layer can be shared with other layers. Therefore, this design approach is also easier to incorporate in new applications.

Time information is required by many protocols and applications in WSNs (as discussed in section 1). There is thus a need to estimate it efficiently and to share it across the network stack. Because it is very rare for synchronisation not to be necessary in WSNs, we advocate integrating the synchronisation service into the routing layer. Next two sections describe two novel routing protocols which enable energy efficient time synchronisation of nodes and use a cross-layer design approach:

- **CLEAR** – Cross Layer Efficient Architecture for Routing, which works well in cluster-based WSNs.
- **RISS** – Routing Integrated Synchronisation Service, intended for use in non-hierarchical WSNs.

We implemented and tested the energy efficiency and time precision of these solutions in different environments. Both protocols were deployed on sensor nodes using TinyOS with the aim of comparing their performance with other similar techniques. Both protocols not only provide high precision of time synchronisation for low energy cost, but also achieve an excellent performance over a variety of network topologies.

4 CLEAR

In this section we describe the CLEAR protocol which integrates time synchronization into the network layer and is designed to work well in hierarchical, cluster-based network topologies.

4.1 A description of CLEAR

We could solve the experienced LEACH shortcommings described in Section 3.2 with a traditional layered network solution. It would involve a packet acknowledgement mechanism to prevent data loss and a separate duty-cycle mechanism for FTSP to keep all of the nodes synchronised. This solution however has some drawbacks. First of all, it increases data traffic by adding ACK messages. Besides, independent sleep modes for routing and synchronisation schemes are energy inefficient because more energy is dissipated due to the frequent awaking and putting into sleep of the nodes. Also the duty-cycle mechanism for FTSP may require additional computations and can be complicated to implement.

The problems stated above led us to consider a solution that requires interaction of network stack layers. This idea motivated a variant of LEACH called CLEAR which combines both functions: routing and time synchronisation. The method we use to synchronise the clocks requires interaction with the MAC layer.

The CLEAR protocol increases the energy efficiency of LEACH by optimising the duty cycle of the subordinate nodes in the clusters. It minimises the number of times subordinate nodes need to wake-up because the exchange of time information between nodes is managed by the routing protocol and the time information is transmitted according to the same schedule as data is transmitted.

The overall scheme operates as follows. Before starting the routing of messages with CLEAR, nodes need to synchronise their clocks during an initial set-up phase. We used a method based on the periodic flooding messages originating at the base station. These contain time-stamp information which is compared by the receiver with its local clock. With this method we can attain a high precision at discrete points in time. However, a very small difference in the clock frequencies can introduce errors for the global time estimation between synchronisation points. Thus the protocol implements linear regression to compensate for frequency differences of the clock crystals.

Once the nodes' clocks are correlated during the initial set-up phase, the routing protocol can start routing the data. It elects cluster heads and other sensors join the closest cluster. Then the subordinate nodes receive the TDMA schedule from the cluster-heads and go to sleep until their communication slots occur. However, the synchronisation information has still to be updated because of the clock drifts mentioned previously. Therefore CLEAR aligns the crystals of subordinates nodes in the following way:

- A subordinate node transmits data to its cluster-head and waits a maximum of T seconds for an acknowledgement.
- Cluster-heads which receive packets from subordinate nodes check each packet's integrity and if there are no errors, stores them for retransmission to the base station. An ACK message is then sent which includes the synchronisation protocol information. Content of this packet is more detailed in next paragraph.
- After the arrival of the ACK message, the subordinate node updates its clock time on the basis of the information contained in the ACK packet. It then erases the original data and goes to sleep until the next communication slot.
- If the ACK packet is not received within T seconds, the node enters a low power mode but this time the data is stored and will be sent with subsequent sensor measurements at a future transmission time.

There are two reasons for integrating this acknowledgment scheme into LEACH. Primarily, this mechanism guarantees data delivery to the base station. Our experiments carried out in a real environment, described in Section 3.2, showed us that LEACH may encounter a high rate of packet delivery failure. According to Zhao *et al.* Zhao and Govindan (2003) even reasonable link layer loss recovery is unable to mask the high data losses WSNs encounter in a real environment. Secondly, CLEAR makes the duty cycle of the subordinate nodes more efficient. The ACK packet integrates synchronisation data and therefore subordinate nodes do not have to wake up solely for receiving time information. The packet contains two fields: the source address and the global time of network. To increase energy efficiency, the cluster-head acknowledges all packets sent by a node within one time slot with a single ACK message.

The CLEAR synchronisation method is based on the comparison of the global time value included in the ACK packet and the local clock value at the reception instance. Such a method may degrade precision because the medium access time is random. We implemented MAC layer time-stamping[3] on the send and receive side in order to reduce message transmission jitter and in consequence the synchronisation error. The cluster-head performs time-stamping when a part of the ACK message was already transmitted whereas the subordinate node records the reception time of the global time field. With this method we observed a maximum synchronisation error of 2 ms. The node which receives the ACK message updates the global time information, erases acknowledged measurements and goes to sleep. CLEAR minimises the number of times a node needs to awaken because it turns on the transceiver for both routing data and updating synchronisation information.

The cross-layer approach is present in two aspects of the CLEAR protocol. First, it uses a MAC layer time-stamping to reduce time jitter caused by the variation of channel access time. The feature requires interaction of MAC, PHY and application layers. Second, the routing protocol uses a common schedule for sending sensory measurements and also to synchronise the nodes. The schedule needs to be shared between Network and application layers. In the next section we present an overview of CLEAR performance.

4.2 CLEAR performance

In this series of experiments we used the same WSN and data delivery performance measurement as described in section 3.2. In our tests comparing LEACH and CLEAR we were primarily interested in two aspects of wireless communication: energy consumption and packet loss.

The highest packet delivery ratio we could attain with real world experiments was 90% although it could drop to as low as 40%. Therefore some mechanism of assured delivery is required and CLEAR integrates into LEACH a message acknowledgement mechanism. Thus at the price of increased channel occupancy we guarantee packet delivery to the base station. Even if some messages are lost with CLEAR, the acknowledgement mechanism ensures the delivery of sensor measurements to the base station.

We also analysed the influence of CLEAR on the network lifetime. As mentioned already, CLEAR makes the duty cycle of the subordinate nodes more efficient, and so the network can operate for longer periods than when using LEACH with the same initial energy resources. In order to predict system longevity, we measured the current consumption of the Tmote Sky sensor nodes. To do this we used the voltage measurement circuit described in the appendix.

First, the sensor nodes were operated using the LEACH and FTSP protocols. The frequency differences of the crystals used in Tmote Sky motes introduces drifts up to 40 μs. In our experiments we observed that sensor nodes became unsynchronised after 16 minutes when LEACH initiated the periodic low power mode due to FTSP messages not being received (which was described in section 3.2). Hence we could not run experiments with LEACH and low power mode for a long time. Instead, the protocols are implemented with a layered architecture and the sensor nodes are not put to sleep. On average, the nodes drained a current of 20 mA each, so with two typical AA batteries (2000 mAh) as a power source, network longevity can be estimated to be 100 hours.

In contrast, CLEAR facilitates the nodes entering into low power mode. Then the current consumption drops to about 250 μA. So with nodes sending packets every two minutes, CLEAR can extend the network lifetime for the same battery resources up to around 840 hours (when the cluster-head election probability equals 10%). This value depends on two parameters of the routing scheme: the fraction of nodes being made cluster-heads and the frequency of packet transmission. Figure 3 shows the variation of WSN lifetime as a function of these variables.

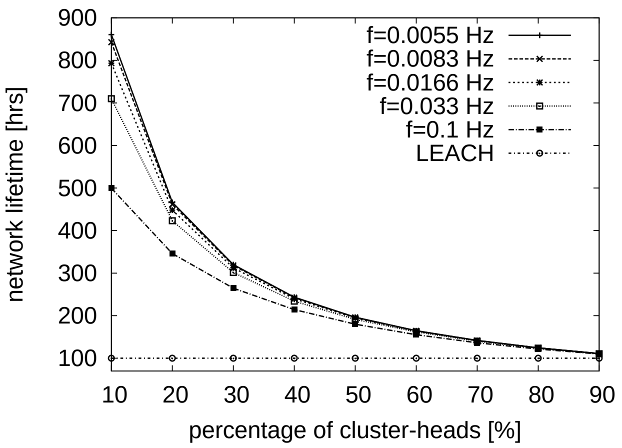

Figure 3 Network lifetime with CLEAR protocol using different packets transmission rate f.

Diminution of the cluster-head election probability has a positive influence on the network longevity. With more nodes operating in low power mode, the energy draining from the system drops. Also, frequent data transmission by subordinate nodes can limit the lifetime of the WSN.

High packet loss was encountered with the LEACH protocol when many packets had to be stored in a buffer before being sent because of the long sleeping period. It may be a critical problem for the applications requiring high measurement frequencies. CLEAR makes the duty cycle more energy efficient because nodes send packets and are synchronised during the same wake up interval. Besides it ensures data delivery to the base station so the nodes can be put into sleep for a longer time without decreasing the measurement frequency.

4.3 CLEAR: summary

The experiments performed in a real environment showed us that the traditional layered architecture may make the routing and synchronisation protocols inefficient in terms of energy consumption. We discovered also that LEACH, a hierarchical routing protocol, may encounter high packet loss. Thus we proposed to integrate synchronisation, routing, MAC time-stamping and an acknowledgment scheme into a new, cross-layer designed protocol called CLEAR. It drains less energy from the nodes compared with LEACH and in consequence extends the network lifetime. It also guarantees data delivery to the base station.

In clustered networks some nodes become cluster-heads and they are responsible for managing neighbours. The other nodes join clusters and perform tasks according to the commands sent by their cluster-heads (*e.g.* they send samples in accordance with the schedule generated by the cluster-head). Thus the hierarchy importance increases in the downstream direction because a node is responsible for managing its upstream neighbors. Also a clustered architecture requires that nodes communicate over bidirectional links *i.e.* data must be sent over the same links in both directions in these networks. For example the cluster-heads transmit the TDMA schedules and data requests in the upstream direction but the subordinate nodes send sensor samples over the same links in the downstream direction. For these reasons CLEAR is best suited to such networks, since it also requires bidirectional links. With CLEAR, a cluster-head sends the synchronisation information in the reverse direction over the same link as was used by a subordinate node to send sensor samples.

5 RISS

In a flat network, the time synchronisation can be efficiently achieved with a different approach than in a hierarchcal network where cluster nodes have the responsibility to distribute time information to subordinate nodes. In non-hierarchical networks all nodes have the same functionality and the RISS protocol demonstrates how the time synchronisation can be achieved in networks with such architecture.

5.1 An outline of RISS

In most WSN application scenarios (*e.g.* monitoring of the outdoor environment, habitat, offices, systems, buildings, and industrial sites) nodes perform the computational and transmission tasks periodically. It is rarely the case that the node stays active while it waits for an event to happen but even in those cases it must periodically broadcast a beacon message to maintain network connectivity. This periodic transmission being a common property of WSNs applications, we propose to exploit it to obtain an overall network time reference. Less overhead is required to estimate neighbour's clock frequency rather than to generate a global network time. Accordingly, since the previously mentioned monitoring applications do not require an absolute time-of-day reference, synchronisation supported by RISS is limited to the estimation of the neighbours' clock frequency.

Typically, sensory data is forwarded regularly to the *downstream* nodes (in the direction towards the base station) from the *upstream* nodes (in the reverse direction). RISS protocol extracts the time information from the periodic communication of the nodes. We chose this approach because of the following reasons. First, it reduces the protocol overhead because the downstream node can estimate the frequency of a neighbour's clock on the basis of periodic packets sent by the neighbour. So RISS does not require any additional packets and in consequence the energy cost of the protocol is reduced. Second, in WSNs data is mostly transmitted in downstream direction and it is possible that a pair of nodes can communicate directly only in that direction as they are connected by an asymmetric link Sang et al. (2010). These nodes could communicate indirectly and synchronise using multi-hop bi-directional communication. But the synchronisation error between these two nodes is minimised if they synchronise on the basis of the synchronisation data sent over direct link in the downstream direction. Finally, it may even happen that there is no multi-hop path from the downstream to the upstream node even these nodes can communicate directly in the opposite direction. In this situation, the synchronisation information may only be sent in the downstream direction.

So in order to synchronise, a node learns the clock frequency of every upstream neighbour. To do that the node uses the periodicity of the operation of the upstream neighbour and the time information added to the packet by the sender. So the task of the upstream node is to add information to the message that would facilitate the time alignment of nodes. However, we recommend adding not a time-stamp of the sending instant, but rather the time which has elapsed since the last local periodic interrupt clock (the reasons for that are described in section 5.2.1). Then the receiver after collecting multiple messages can calculate the neighbour clock frequency with respect to its local clock. Below, a more detailed description of RISS is provided.

5.2 Detailed description of RISS

In the RISS protocol every node knows the clocks' frequencies of its upstream neighbours. So, for example the base station has to evaluate the clocks of the nodes which report data directly to it. In this section we describe how a node learns clock's frequencies of its upstream neighbours.

5.2.1 Sender operation

The task of the upstream sensor nodes is to perform some periodic operation (*e.g.* to awaken the transceiver) at time T_i and to send a packet (Figure 4). The task period P is known to the downstream receiver and it may be decided before the deployment of the network or changed operationally and reported to all nodes.

It takes an amount of time W_i for the sender to actually transmit the packet. The random delay W_i reflects the time taken to assemble a packet and to submit it to the MAC layer and time to access the channel. This waiting time is inserted into the message just after the transmission of the SFD[4]. This process is called MAC layer time stamping and it reduces the synchronisation error due to channel access time jitter because the value of channel access time will be reflected in the corresponding W_i. After the insertion of the waiting time value, the FCS[5] field must be recalculated.

Sending information just about the random time delay (W_i) and not the clock reading at the time of transmission ($T_i + W_i$) has two advantages. Firstly, this number occupies a small range in comparison to the possible clock scope. Thus, sending it requires less energy and packet space. In our experiments we observed that the maximum nondeterministic delay of the transmission was 566 ticks using a 32kHz clock. We need 10 bits to transmit this number instead of the 32 bits required for sending the time-stamp. This is important in WSNs where most of the energy resources are consumed by the transceiver Fedor and Collier (2007). Also in WSNs, the packet space is very limited. For example in Tmote's implementation of the network stack the user has 28 bytes for application data. So saving even 22 bits of the header might have a substantial benefit. In the next paragraph we describe how the receiver synchronises with its neighbours.

5.2.2 Receiver operation

The downstream node records the information about the last Q packets received from a sensor node in order to estimate its clock frequency. This data is stored in a $Q \cdot l$ array where l is the number of upstream neighbours. For every packet received, the node must save the corresponding W_i and R_i values, where R_i is the local clock value at the reception of i-th packet. After that, the sensor node estimates the clock frequency of the most recent transmitter from the set of W_i and R_i values stored in the table using the method below. This calculation is performed every time a packet is received. We compared two different techniques to estimate the sender's clock frequency.

Linear regression:. We implemented linear regression using a standard optimised algorithm Press et al.

Variable	Meaning
T_i	Wake-up instance of the transceiver to execute i-th periodic task
P	Duty-cyle period of the transceiver
W_i	Time elapsed between wake-up of the transceiver and sending of the i-th packet
R_i	i-th packet reception instance
$\hat{R}_i$	Estimated reception instance of the i-th packet
Q	The number of past packets processed
$F_{r\to t}$	Frequency of the transmitter clock
$I(k)$	Time occurrence of an observed event expressed at the node k
G	Difference between estimated packet arrival and awake time in order to reduce packet loss

Table 1 Variables used in the mathematical formulas.

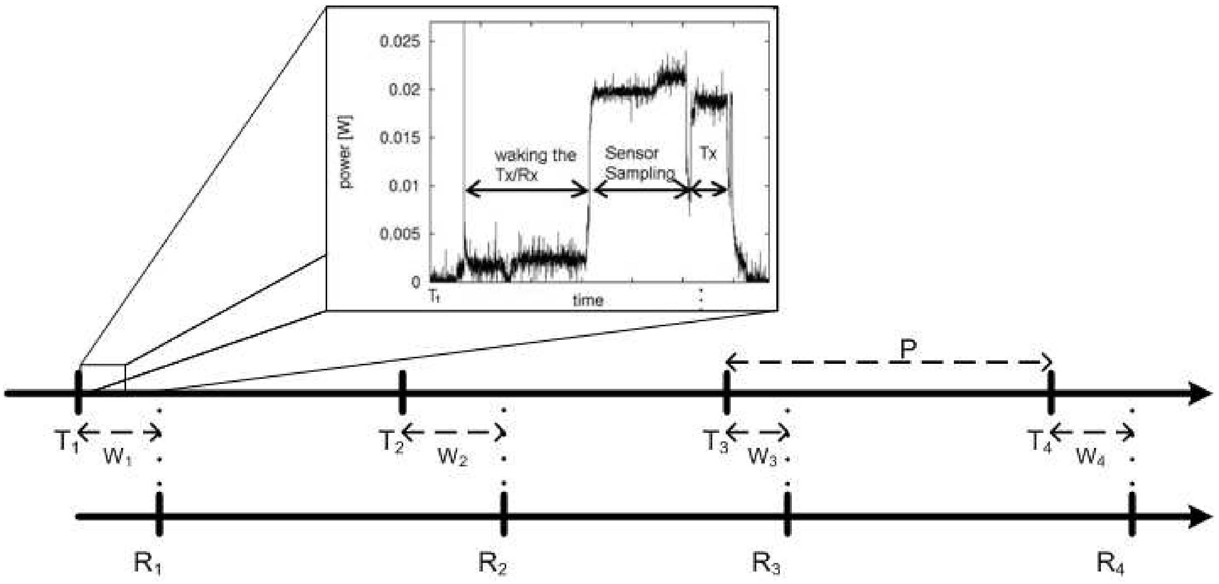

Figure 4 Time line of the operation of the receiver (bottom) and transmitter (top). Every P seconds the sender wakes up the transceiver at its local time T_i, samples the sensor, and transmits the SFD of the packet at local time $T_i + W_i$. Sender adds the value W_i to the message and turns off the transceiver. The receiver hears the SFD at the local time R_i.

(1988) [6]. So, if we want to estimate the frequency of the transmitter clock $F_{r\to t}$ with the Q most recently received packets, the sum sy of receiver time-stamps of the packets reception is given by:

$$sy = \sum_{i=c}^{c-Q+1} R_i \tag{1}$$

Every time-stamp R_i corresponds to the time of transmission of the SFD expressed using the sender's clock. If i is the packet sequence number, the sum of the SFD transmission time-stamps can be expressed as follows:

$$sx = \sum_{i=c}^{c-Q+1} i \cdot P + W_i \tag{2}$$

So the frequency of the transmitter clock $F_{r\to t}$ can be obtained with the following equation:

$$F_{r\to t} = \frac{\sum_{i=c}^{c-Q+1} t_i \cdot R_i}{s} \tag{3}$$

where $t_i = i \cdot P + W_i - \frac{sx}{Q}$ and $s = \sum_{i=c}^{c-Q+1} t_i^2$

These calculations turned out to be very resource demanding. In particular the division by large integers in equation 3 requires a considerable amount of time. Because of the time expenditure and in consequence, energy inefficiency of this method we propose another technique for estimating the frequency of the sender's clock for use in the RISS protocol.

Fast approximation:. We call the frequency of the transmitter clock expressed at the receiver $F_{r\to t}$. For every packet sent with an inter-packet period equal to P we can write:

$$R_i - R_{i-1} = F_{r\to t} \cdot (P - W_{i-1} + W_i) \tag{4}$$

or

$$F_{r\to t} = \frac{R_i - R_{i-1}}{P + (W_i - W_{i-1})} \tag{5}$$

We want to avoid the division operation in equation 5 for computational efficiency. For convenience we write $R_i - R_{i-1} = \Delta R_i$ and $W_i - W_{i-1} = \Delta W_i$. Hence, equation 5 becomes:

$$\begin{aligned} F_{r\to t} &= \frac{\Delta R_i}{P + \Delta W_i} \\ &= \frac{\Delta R_i}{P} \cdot \frac{P}{P + \Delta W_i} \\ &= \frac{\Delta R_i}{P} \cdot \frac{1}{1 - (-\Delta W_i / P)} \\ &= \frac{\Delta R_i}{P} \cdot (1 + \\ &\quad + \left(\frac{-\Delta W_i}{P}\right) + \left(\frac{-\Delta W_i}{P}\right)^2 + \left(\frac{-\Delta W_i}{P}\right)^3 + ...) \end{aligned} \tag{6}$$

which converges provided $|\Delta W_i| < P$. In this context[7], $|\Delta W_i| << P$ and so $F_{r\to t}$ is well approximated by $\hat{F}_{r\to t}$ where

$$\begin{aligned} \hat{F}_{r\to t} &= \frac{\Delta R_i}{P} \cdot \left(1 - \frac{\Delta W_i}{P}\right) \\ &= \frac{\Delta R_i}{P} - \frac{\Delta R_i \Delta W_i}{P^2} \end{aligned} \tag{7}$$

The second term is well approximated as $\Delta W_i / P$ since $\Delta R_i \simeq P$. Hence:

$$\hat{F}_{r\to t} \simeq \frac{(R_i - R_{i-1}) - (W_i - W_{i-1})}{P} \tag{8}$$

We calculate only the numerator of that expression. Defining $F'_{r\to t} = P \cdot \hat{F}_{r\to t}$ and averaging that value over the Q most recently received packets we obtain:

$$\overline{F'_{r\to t}} = \frac{\sum_{i=c}^{c-Q+1} [R_i - R_{i-1} - W_i + W_{i-1}]}{Q} \tag{9}$$

We show in sections 5.3 and 5.4 how the obtained value of $\overline{F'_{r\to t}}$ can be used for two main applications of time synchronisation in WSNs: duty cycling of nodes and for estimation of a common time reference.

5.3 Duty cycling of the node

Duty cycling is one of the most common techniques used in WSN to save the energy resources of the sensor nodes Karl and Willig (2007); Anastasi et al. (2009). As stated previously, the transceiver drains more energy than other subsystems from the sensor node. For this reason it seems reasonable to turn it off periodically. However, in order to maintain the connectivity of the network we must ensure that when the transmitter wants to forward a packet, its destination is awake. There is thus a tradeoff between keeping nodes asleep for the maximum possible time and minimising the packet loss due to the receiver not listening. Thus it is necessary that the communicating sensor nodes have the same time reference and wake up at the same instance to communicate. We can use the frequency of the sender's clock as estimated with the RISS protocol to synchronise the duty cycling of the nodes in the following way.

When the downstream node captures a sufficient number of packets from its upstream neighbour to estimate its clock frequency, it calculates the arriving time of the next packet and goes to sleep. This estimation can be done as follows. After reception of packet i, we can expect that the arrival time-stamp of packet $i+1$ is:

$$R_{i+1} = R_i + P \cdot F_{r\to t} + F_{r\to t} \cdot (W_{i+1} - W_i) \tag{10}$$

However, we cannot predict the term W_{i+1} which is stochastic. Thus we propose to replace the difference $(W_{i+1} - W_i)$ by a minimum of that value over the last Q packets $\min_{a\epsilon(0:Q-1)}(W_{i-a} - W_{i-a-1})$. We do that to minimise the packet loss due to the receiver waking up after the packet's arrival. In order to reduce the error of calculation of the neighbour's clock frequency we propose to combine the results of multiple estimates. Also, the term $\min_{a\epsilon(0:Q-1)}(W_{i-a} - W_{i-1-a})$ corresponds to a small value so we can neglect its multiplication by $F_{r\to t}$ which is very close to one. So equation 10 becomes:

$$\hat{R}_{i+1} = R_i + P \cdot F_{r\to t} + \min_{a\epsilon(0:Q-1)} (W_{i-a} - W_{i-1-a}) \tag{11}$$

The arrival of the next packet can be predicted with equation 11. However, a packet loss may still occur because the value $\hat{R}_{i+1}$ is estimated on the basis of the arrival times of the most recent Q packets. Thus, if for example the channel occupancy drops the next packet will have a smaller channel access time than the previous Q packets. Thus, it will be transmitted before estimated time $\hat{R}_{i+1}$ and not heard by the still sleeping receiver. To prevent such packet loss we propose to awake the receiver earlier than the estimated time $\hat{R}_{i+1}$ by a guard interval G which we determine empirically. For a general application scenario it would be most reasonable to make the value of G adaptive and either to increase it when the reception of a packet is missed or decrease its value when a receiver spends a long time idly listening. However, RISS can be used in networks with asymmetric links where the receiver cannot acknowledge the reception of a packet over the same link. If G were adaptive, losses of several consecutive packets could occur. In consequence, if the link is asymmetric so that the receiver cannot directly inform the sender about the reception failure, a significant amount of data can be lost. Therefore we propose to use a fixed value of G and to keep the receiver awake after a reception failure until the arrival of the next packet. With this mechanism two successive packet losses cannot occur.

We can replace the term $P \cdot F_{r \to t}$ with the value of $\overline{F'_{r \to t}}$ estimated with the fast approximation method to reduce the calculation time of the next packet arrival. Therefore, equation 11 becomes

$$\hat{R}_{i+1} = R_i + \overline{F'_{r \to t}} + \min_{a \epsilon (0:Q-1)} (W_{i-a} - W_{i-1-a}) - G \tag{12}$$

where $\overline{F'_{r \to t}}$ is obtained with equation 9.

$\hat{R}_{i+1}$ is the state variable denoting the required wake up time of the node. It is updated after every reception of a packet. When a receiver has multiple upstream neighbours, it must keep a record of estimated, future arrivals from all of them. Whenever it receives a packet, it predicts the arrival time of the next message from the same sender and goes to sleep until the projected time $\hat{R}_{i+1}$ of the next message to arrive from all upstream neighbours.

5.4 Estimation of event time correlation

Another major purpose of obtaining a common network time reference in WSN is the ability to time-stamp events. In most WSNs applications the user must know the time of occurrence of a sensed event. We propose a method of generating time-stamps which requires only a little protocol overhead. In section 5.2.2 we describe how every node can estimate the frequency of its upstream neighbours with RISS. This information may be used at each node to estimate the time-stamp, expressed in local clock units, of the received samples from an upstream neighbour. To obtain that value the receiver needs to know the time, expressed with the clock units of the upstream neighbour, elapsed between the occurrence of the sample and the instance of sending the packet containing the sample. Then the receiver is able to express that period in its local clock units and by consequence precisely estimate the instance of acquisition of the data sample. To obtain that value it must substract the calculated value of elapsed time from the instance of reception of the SFD of the packet. Thus we propose to insert into the packet a field which is updated by every node on the communication chain between the source of the packet and the base station to reflect its own local clock. Initially, the sensor node k records the time $E(k)$ elapsed between an observed event (which can be sampling of the sensor, a packet reception *etc.*) and the periodical clock interrupt ($T_i(k)$) described in section 5.2 in that field. Then the receiver (node $k+1$) computes in its local clock units the period between the event observed by the sender and the SFD of the received packet at time $R_i(k+1)$. This value can be obtained as follows. If the time value sent by the node k is $E(k)$ and the value of $\overline{F'_{r \to t}}$ is estimated at the receiver with the fast approximation method, the time of occurrence $I(k+1)$ of the observed event can be obtained with the following equation:

$$I(k+1) = R_i(k+1) - \frac{\overline{F'_{k+1 \to k}} \cdot [W_i(k) + E(k)]}{P} \tag{13}$$

where $W_i(k)$ is the value sent in the packet (see section 5.2.1).

If the receiver of the packet is not the base station, it forwards the value $E(k+1)$ to the downstream neighbour (node $k+2$). The value $E(k+1)$ corresponds to the time elapsed between the initial sensor sampling, expressed at node $k+1$, and transmission of the packet and is given by:

$$E(k+1) = \frac{\overline{F'_{k+1 \to k}} \cdot [W_i(k+1) + E(k)]}{P} + P \tag{14}$$

Then the receiver (node $k+2$) will collect the SFD of the packet and repeat the update of the time field before the transmission. So the time of occurrence $I(k+2)$ of the observed event expressed at node $k+2$ can be obtained with the following equation:

$$I(k+2) = R_i(k+2) - \frac{\overline{F'_{k+2 \to k+1}} \cdot [W_i(k+1) + E(k+1)]}{P}. \tag{15}$$

where $R_i(k+2)$ is the reception time of the packet from node $k+1$ containing the value $E(k+1)$ and $W_i(k+1)$ is the random delay of that packet. This operation continues until the message reaches the base station.

For example, if a node A observed an event e at its local time 1000 and wakes up at instance 3000 to transmit the packet at 1200 local clock units, it will insert in the packet two values: $E(k) = 3000 - 1000 = 2000$ and $W_i = 1200 - 1000 = 200$. Then the receiver node B will record 7500, the value of its local clock at the reception instance of the packet. If we assume that node B estimated with RISS the value of $\overline{F'_{B \to A}}$ (see equation 9) to be 66000 and duty-cycle period of node A equals 60000 then the node B can compute the time of occurence of event e in its local clock units using equation 13, $I(k+1) = 7500 - 66000 \cdot (200 + 2000)/60000 = 5080$.

5.5 Integration of RISS into network layer

RISS is a generic method for time synchronisation in a flat WSNs. It represents tasks of the nodes and information they need to exchange in order to obtain network synchronisation. RISS requires interaction between MAC and network layers. RISS applies a so-called MAC layer time-stamping to increase precision of the synchronisation. It also uses information about neighbours which is managed by the routing protocol. We propose that RISS should be integrated into the network layer.

Incorporating the synchronisation service in the routing protocol at the network layer results in an efficient implementation since the network architecture is managed by this layer. In the synchronisation method we propose, every node must first discover its upstream and downstream neighbours. This information can be deduced from the routing tables.

5.6 RISS performance

The purpose of our experiments is fourfold:

- to compare the effectiveness of linear regression and fast approximation methods (section 5.2.2) to estimate clock frequency
- to optimise the parameters of the RISS protocol
- to verify the robustness of the RISS protocol
- to compare RISS with other synchronisation and duty cycling methods in WSNs

We employed a commonly used sensor network platform, the Tmote Sky sensor node Polastre et al. (2005), for experiments. Each Tmote Sky node has an 8MHz TI MSP430 microcontroller and a 2.4GHz, 250kbps IEEE 802.15.4 Chipcon wireless transceiver. We built a network composed of four nodes and a single base station. Each node wakes up every 10s, transmits the sensor reading to the base station and goes to sleep. The base station collects 1000 packets from every sensor node. The time of next wake up is estimated with the RISS protocol. The base station also turns off the transceiver when the channel is free. The results of experiments were measured both in hardware and software *i.e.* at the end of a test we read from software variables the values of measured quantities and also We used a voltage measurement circuit (see appendix) to estimate the time precision and energy efficiency of the RISS protocol.

5.6.1 Comparison of linear regression and fast approximation methods

Initially we compared the computing time of both methods. The results are shown in table 2.

	linear regression	fast approximation
time execution	75.6ms	3.6ms
precision	≤ 2 clock ticks	≤ 2 clock ticks

Table 2 Time execution and precision of RISS with linear regression and fast approximation

Fast approximation method is much faster than linear regression for a similar level of time precision. Processing inefficiency may be a drawback especially if the estimated frequency is needed to duty cycle nodes. A long calculation may significantly reduce the efficiency of that service. Thus we continued the experiments with fast approximation method in order to optimise it.

5.6.2 Optimisation of RISS

The performance of the duty cycle service depends on two parameters: Q and G (see equation 12) whose ideal values are determined empirically. In order to do that we built a network composed of four nodes and a single base station. Each node wakes up periodically, transmits the sensor reading to the base station and goes to sleep. The time of the next wake up is estimated with the RISS protocol. The base station also turns off the transceiver when the channel is free.

Optimisation of Q. The first series of experiments is to estimate the ideal number of past packets (called Q) needed to predict the next packet arrival time (see equation 9). There is a trade-off between time execution and precision when choosing Q. As Q increases, the accuracy of estimation increases because a larger number of past packets is considered when estimating next packet's arrival time. However, the execution time of the algorithm (when the node is awake) increases with Q. We want to determine empirically the value of Q that maximises the energy efficiency of the duty cycling protocol. We configured the WSN as follows. Each node sends a sensor sample to the base station every 10s. The base station collects 1000 packets from every sensor node. We measure the average awake time of the base station transceiver as a function of Q. The optimal value of Q is platform dependent and we performed this experiment using TmoteSky nodes Polastre et al. (2005).

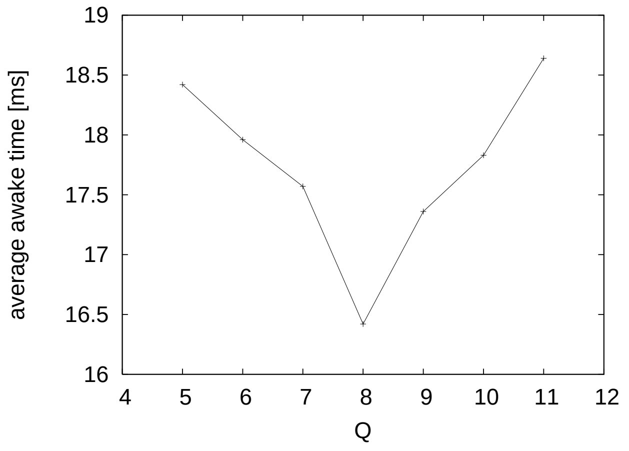

Figure 5 Average transceiver awake time as a function of Q, the number of past packets processed.

The result of this test is shown in Figure 5. The graph has a local minimum when Q equals 8. The precision of the synchronisation alorithm increases with the value of Q in the interval between 5 and 8 and therefore the avarage awake time decreases because we can predict more precisely the arrival time of the next packet. However for the values of Q grater than 8 the time required to run synchronisation algorithm becomes significant. During the execution of the synchronisation algorithm the nodes stays awake and therefore for the values of Q grater than 8 the average awake time increases with Q. It means that, for this hardware configuration, the receivers should estimate the arrival of the next packet on the basis of the time-stamps of the 8 previous messages. Then the average time spent listening to the channel, processing the packet and turning off the transceiver is kept low and is approximately 16.5ms.

Optimisation of G. Also, we carried out a series of tests to optimise the value of constant G in equation 12. It represents the time which we subtract from the estimated time-stamp of the future message and is used in order to compensate for the random component of the arrival time. The energy efficiency drops with the value of G but if we set the constant G too small, a packet loss may occur. This failure of reception disables duty cycling until the arrival of the next packet and is energy inefficient. So, there is a trade off between setting G to a low value which may increase the energy draining because of the missed packets and a larger value of G that extends the awake time for every packet. With the same network of four nodes and a base station we measure the average awake time and number of packets lost as a function of the value of G. The value of Q is set to 8 and every sensor node sends 1000 packets. The results of this experiment are shown in Figure 6.

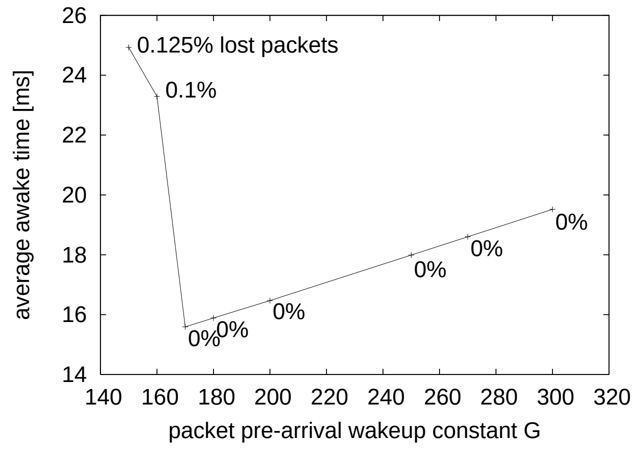

Figure 6 Average transceiver awake time as a function of the pre-awake constant (G in equation 12).

For values of G greater than 170, the average transceiver awake time increases linearly. We do not observe any packet loss in those circumstances. However, if the constant G is below 170, the rate of energy consumption increases. This is due to packets being missed by the receiver. The duty cycling is then disabled until the next message arrival. Four out of 4000 (0.1%) packets missed when G equals 160. When G is 170, the transceiver is powered up $170/32768kHz = 5.19ms$ earlier than predicted in order to compensate for the stochastic component of the packet arrival time.

5.6.3 Sensitivity of RISS

In this series of tests we want to verify the influence of application specific parameters on the performance of RISS. We test how the energy efficiency varies with the packet size. We also want to test whether the RISS protocol is sensitive to the duty cycle period.

Influence of packet size on the energy efficiency of RISS. With the network of four nodes and a base station, we set up the protocol parameters to the optimal values previously determined. We vary the size of the packet sent by nodes and we record the average awake time of the base station. The processing time of RISS increases linearly with the number of bytes of the message (see Figure 7). For the minimum packet size of 4 bytes

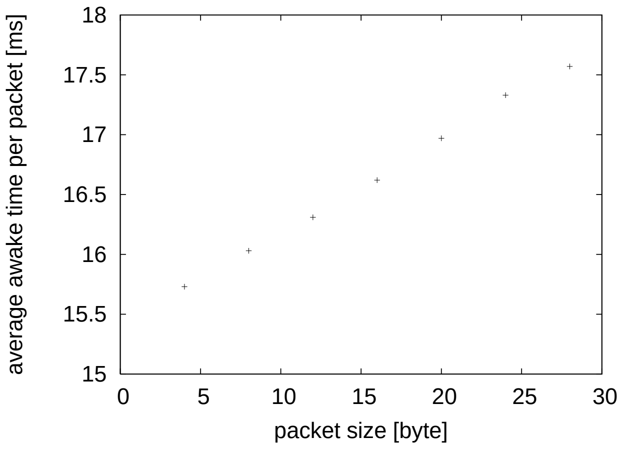

Figure 7 Average transceiver awake time as a function of the packet size.

we measured an awake time of 15.73ms and for the Tmote's maximum packet size of 28 bytes we evaluated this value to be 17.57ms. So we deduced that on average the transceiver power up time increases by 0.0784ms for an additional byte in a packet. The CC2420 transceiver has a transmission rate of 250kbps, and so takes 0.032ms to transmit one byte. The difference of 0.0784-0.032=0.0464ms is due to the processing by the duty cycling protocol of an additional byte. Thus in applications where delay is not a major concern, sensor samples should be stored and sent in packets of the maximum size in order to optimise energy efficiency. For example if a node sends a sensor sample of four bytes in an independent packet, the receiver is awakened for 15.73ms to compute the arrival time of the next packet with RISS. However if this sample is inserted in a packet which contains previous sensor samples, it would increase the processing time of the arrival time of the next packet (and thus the awake time) by only 0.18ms.

Influence of the duty cycle period on the energy efficiency of RISS. We keep the same network architecture for this test. However, this time we modify the frequency of packets and analyse the variation of the energy efficiency. Our observations are plotted in Figure 8.

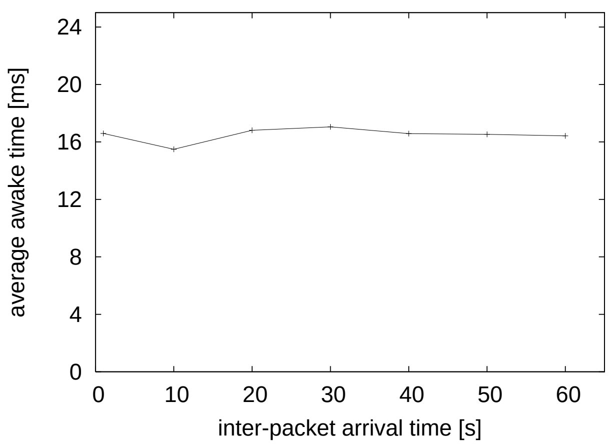

Figure 8 Average transceiver awake time as a function of the inter-packet arrival time

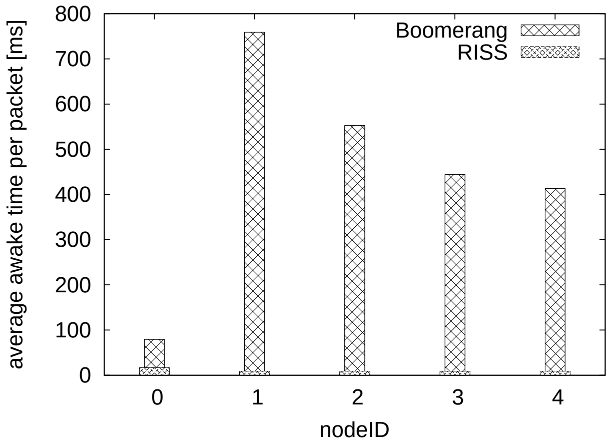

Figure 9 Average transceiver awake time with Boomerang and RISS protocol.

The energy efficiency is inversely proportional to the average awake time, which from Figure 8 can be seen to be relatively insensitive to variations in packet arrival rate. The algorithm performs well in the region where inter-packet arrival time is between one and sixty seconds. Although the performance of RISS may then drop, in real applications we will rarely increase the inter-packet period above one minute because it may lead to a loss of network connectivity, and to a lack of information about the neighbour nodes.

5.6.4 Comparison of RISS with other protocols

Comparison of RISS with other duty cycling protocols. To compare the duty cycling performance of RISS we chose as a reference the scheduling protocol which is a part of Boomerang, Moteiv's distribution of TinyOS Boomerang (Boomerang). In the first stage we deployed Boomerang on a network comprising four sensor nodes and a base station. Each node sends a sensor measurement every 10s and the *lowpower* coefficient is set to 1%, the minimum possible value for Boomerang, and thus probably the most efficient. After collecting 500 packets at the base station, we measure the average awake time of every sensor node. In the second phase of the test we installed the RISS protocol on the same network. Similarly, nodes send 500 packets with the inter-packet interval of 10s. The comparative results are plotted in Figure 9.

Each node stays asleep for longer when using duty cycling with RISS. The average awake time per node equals 449.6ms with Boomerang and 10.6ms with fast approximation method. Node 0 is the base station. We can observe that with Boomerang it consumes less energy than other nodes. The reverse is the case with RISS protocol (see section 5.1). This is due to the different approach to synchronisation. In Boomerang it is the upstream sensor node which adapts to the duty cycling operation of its downstream neighbours. We implemented the opposite solution in RISS (see section 5.1). While the difference of average awake time between the base station and the least efficient node is considerable in the case of Boomerang (679.98ms), it is very small (8.06ms) with our protocol. We attribute this to the authors of Boomerang allowing the minimum duty cycle period to be one percent. Also Boomerang requires the exchange of dedicated packets for synchronisation which extends each node's awake time.

Comparison of RISS with other time-stamping protocols. The RISS protocol may also be used to determine time-stamp of an event. The performance of the solution can be tested by measuring the difference between the protocol's estimate of an event time and its actual instance. To do this we propose the following experiment scenario. A network is composed of a single node (*node A*) and a base station (*node B*). Node B periodically collects sensor samples from node A. It also estimates node A's clock frequency with the RISS protocol. A node C (the reference broadcaster) broadcasts packets at intervals to nodes B and A. For ease of implementation we assured periodic transmission of these packets. Whenever a packet is heard by the two nodes (A and B), they record the time of occurrence of that packet. When node A sends the sensor reading to node B, it adds to the packet the value of the time which has elapsed (according to A's clock) between the packet broadcast by node C and the SFD of the message being currently transmitted. Then, node B uses that value to calculate the time of reception (according to B's clock) of the packet from node C. It compares the obtained result with the time-stamp it applied to the packet from node C using its local clock upon reception of that packet. The difference between these two values corresponds to an error of the estimation of the arrival time of the packet from the reference broadcaster. For every test we vary the frequency of the sensor packets and we find $\max_i E_i$ where E_i is the error in the i-th iteration of the estimation at the base station of the event time. We compare the precision of RISS with that of FTSP Maróti et al. (2004) which we tested with the same scenario. We chose FTSP as a reference protocol because it is one of the most prevailing WSN synchronisation protocols and also its implementation on TinyOS is provided by the authors. Our observations are shown in Figure 10.

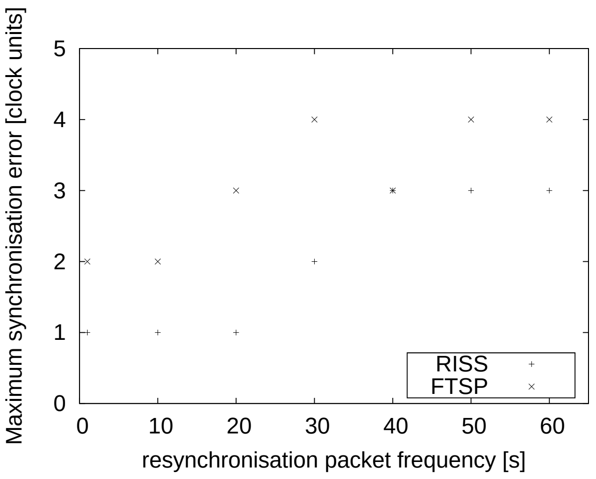

Figure 10 Maximum synchronisation error as a function of the synchronisation beacon frequency for RISS and FTSP protocols.

On that graph we plotted the observed maximum error in event time estimation for FTSP and RISS. This inaccuracy is expressed as the number of ticks of a 32kHz clock. The exactness of the method depends on the frequency of update messages. This update is done by the exchange of beacon packets in case of FTSP and by the addition of time information to application packets in RISS. The shape of the curve shows that the precision of time synchronisation is a decreasing function of the frequency of the time information exchange between nodes. We observed the smallest maximum synchronisation error of $1/32768 \approx 30.5\mu s$ when the time information was updated every second. Because the maximum synchronisation error can only be expressed as an integer multiple of clock units, FTSP and RISS perform equivalently in terms of synchronisation precision. The difference between the maximum synchronisation error of the two protocols is typically zero or one clock unit. The only exceptions in Figure 10 are for the inter-packet periods of 30 and 40s. However, the corresponding maximum synchronisation error of 2 and 3 clock units measured with FTSP protocol was encountered very rarely. Similar outliers would probably be obtained with RISS protocol with a larger set of measurements than the ten used to generate results from Figure 10.

5.7 RISS: summary

RISS exploits the cyclical operation of the nodes to establish the time reference of every sending neighbour. We showed in previous sections how this information can be used for duty cycling of the nodes and obtaining the time-stamp of sensor samples. The experiments carried in a real environment proved the advantage of RISS over other synchronisation protocols. It performs better than FTSP in terms of synchronisation precision. Also RISS is more energy efficient than Boomerang when used for duty cycling of the nodes. Finally, it is possible to increase significantly the sleeping time of the nodes by sending multiple sensor readings in one packet.

6 Comparison of CLEAR and RISS

In the previous sections we described CLEAR and RISS protocols which may be used to synchronise clocks of the nodes in WSNs. Both apply the idea of integrating the synchronisation service in the routing layer. However, they should be used in networks with different routing scenarios. CLEAR is suitable for networks which use hierarchical routing, like LEACH protocol Heinzelman et al. (2000). In this type of networks, some nodes become cluster-heads and they are responsible for managing neighbours. The other nodes joint clusters and perform tasks according to the commands sent by their cluster-heads (*e.g.* they send samples with the schedule generated by the cluster-head). Thus the hierarchy importance increases in the downstream direction because a node is responsible to manage its upstream neighbors. CLEAR adapts to this hierarchy because in this protocol nodes synchronise clocks according to the downstream neighbors.

CLEAR should be applied in hierarchical networks also because the links in these systems must be bidirectional. This is due to the fact that data must be sent over the same links in both directions in these networks. For example the cluster-heads transmit the TDMA schedules and data requests in the upstream direction but the subordinate nodes send sensor samples over the same links in the downstream direction. However the asymmetric links are frequently encountered in wireless networks Sang et al. (2010) and thus these links must be discarded from the routing tables in hierarchical networks. This can be done by eliminating the communication with nodes located in the transition region. So because a hierarchical network has to eliminate communication over asymmetric links, the CLEAR protocol can be applied in this type of WSNs. However, in non-hierarchical networks the asymmetric links problems may be overcame by sending packets over different routes in downstream and upstream directions.

Indeed, in a non-hierarchical WSNs the communication over asymmetric links is possible. It is only required that nodes are able to discover such connections and be aware of their presence. To enlarge the connectivity of the network these links can be maintained, but the data is transmitted over separate links in different directions. Thus the CLEAR protocol cannot be applied in non-hierarchical WSNs. RISS is more suitable protocol for such systems. It is characterised by a smaller overhead than CLEAR and can obtain energy efficiency when used for duty cycling of the nodes.

Although RISS performs better in terms of energy efficiency than CLEAR, the latter protocol should be applied only in hierarchical WSNS. This is due to the different ways of synchronisation used by both protocols. In RISS protocol downstream mote synchronises to its upstream neighbour and as it was argued before, in hierarchical networks it is the upstream mote that should be synchronised to a downstream node.

The proposed solutions overcome the overhead incurred by traditional layered synchronisation architectures use the so called cross-layer method where communication between nonadjacent layers is enabled. The advantage of these novel methods was tested by simulations and by experiments carried out with WSNs deployed in different environments and various communication scenarios.

APPENDIX

One of the metrics used during our experiment was the energy efficiency of the protocol. This can be expressed and measured in many ways. Experiments showed that among all a sensor node's components it is the transceiver that drains most of the power from the battery. Hence, there are two ways of comparing the energy efficiency of protocols:

- Compare an average power consumption of a node for considered protocols
- Compare an average awake time of the transceiver for examined protocols

In the appendix we describe how measurements of energy efficiency were performed.

A.1 Measurement of average power consumption by the sensor node

To measure an average power consumption by the sensor node we built the circuit depicted in figure 11.

We performed experiment on the Tmote Sky platform from MoteivPolastre et al. (2005). The Tmote Sky module is a low power "mote" with integrated sensors, radio, antenna, microcontroller, and programming capabilities. To be powered it requires a voltage source between 2.7 and 3.6V. In order to measure its power consumption we connected in series a Tmote Sky with a 1Ω resistance. The circuit is closed with a 3V regulated power supply. The power consumption of the sensor node can be deduced from the measurements of the V_r voltage on the resistance because TmoteSky and the resistence are connected in series. So, the same current is drained by the sensor node and flows through the resistance. The power consumption of the TmoteSky can be expressed as:

$$P = V_m \times I_m \tag{16}$$

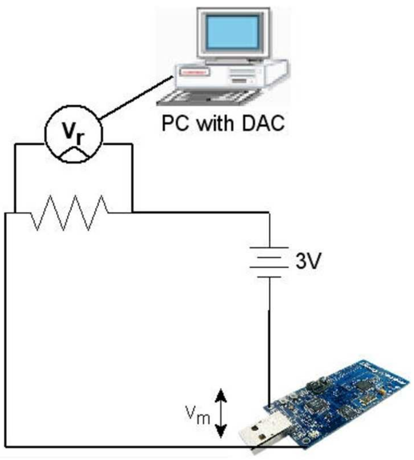

Figure 11 Scheme of the power consumption circuit.

where V_m is the voltage measured across the sensor node and I_m is the current which flows through it. These values can be deduced from the measurement of voltage V_r across the 1Ω resistance. The current I_m can be obtained with the equation:

$$I_m = \frac{V_r}{R} \tag{17}$$

where R equals 1Ω. Finally the power consumption of the TmoteSky is:

$$P = V_m \times \frac{V_r}{R} = (3 - V_r) \times \frac{V_r}{R} = \frac{3V_r}{R} - \frac{V_r^2}{R} \tag{18}$$

According to TmoteSky specification Sentilla Corporation (2006) maximum value of current I_m can be 20mA. Thus the maximum possible value of V_r can be 20mV which can be deduced from equation 17 where $R = 1\Omega$. Hence, from equation 18 we can approximate the value P of power consumed by

$$P \approx \frac{3V_r}{R} \tag{19}$$

which results in maximal relative error of 0.67%.

Hence, to obtain a power consumption of the sensor node we measured voltage V_r using NI 5112 High-Speed digitiser National Instruments (2000). The measured voltage was displayed with a LabVIEW application developed for this purpose. With the same program we could store measured samples for further treatment. In this way we could for example calculate an average power consumption. A typical screenshot of the voltage measurements is shown in figure 12. The apparent negative values are the result of software errors when displaying a large number of collected samples (10MHz sampling rate). The same High-Speed digitiser and the LabVIEW application were used to measure average awake time of the transceiver.

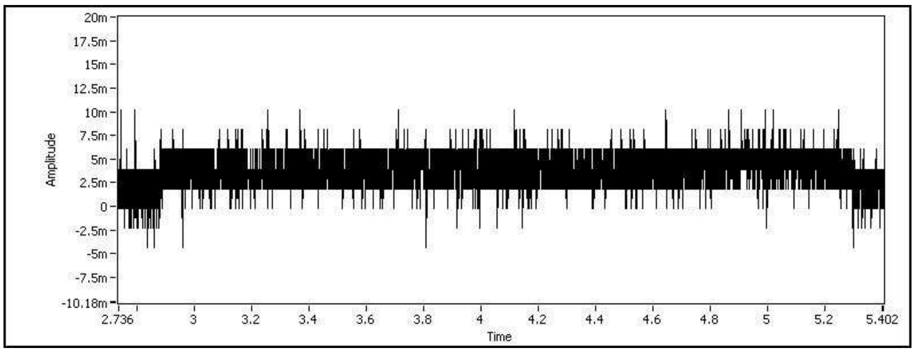

Figure 12 Sample of voltage measurement across resistance R.

A.2 Measurement of average awake time of the transceiver

In our experiments we measured the awake time of the transceiver to compare energy efficiency of different protocols. These measurements were performed in software but to verify these results we estimated the awake time using the NI 5112 High-Speed digitiser and LabVIEW application previously described. Whenever the transceiver was awake the application increased the voltage of a user general I/O pin available on the TmoteSky. After an experiment we measured the time when the pin was at the higher voltage level and compared that with awake time measurement provided by software estimation.

References

Anastasi, G., Conti, M., Francesco, M. D., and Passarella, A. 2009. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks 7,* 3, 537 – 568.

Boomerang. Moteiv corporation. www.moteiv.com/software.

Faizulkhakov, Y. 2007. Time synchronization methods for wireless sensor networks: A survey. *Programming and Computer Software 33*, 214–226. 10.1134/S0361768807040044.

Fedor, S. and Collier, M. 2007. On the problem of energy efficiency of multi-hop vs one-hop routing in wireless sensor networks. In *AINAW '07: Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops.* IEEE Computer Society, Washington, DC, USA, 380–385.

Heinzelman, W. R., Chandrakasan, A. P., and Balakrishnan, H. 2000. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8.* IEEE Computer Society, Washington, DC, USA, 8020.

Karl, H. and Willig, A. 2007. *Protocols and Architectures for Wireless Sensor Networks.* John Wiley & Sons.

Lasassmeh, S. and Conrad, J. 2010. Time synchronization in wireless sensor networks: A survey. In *IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the.* 242 –245.

Levis, P. and Gay, D. 2009. *TinyOS Programming.* Cambridge University Press.

Macedo, M., Grilo, A., and Nunes, M. 2009. Distributed latency-energy minimization and interference avoidance in tdma wireless sensor networks. *Computer Networks 53,* 5, 569 – 582.

Maróti, M., Kusy, B., Simon, G., and Ákos Lédeczi. 2004. The flooding time synchronization protocol. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems.* ACM, New York, NY, USA, 39–49.

Miao, G., Himayat, N., Li, Y. G., and Swami, A. 2009. Cross-layer optimization for energy-efficient wireless communications: a survey. *Wirel. Commun. Mob. Comput. 9,* 4, 529–542.

National Instruments 2000. *NI 5112 User Manual*, National Instruments Literature Part Number: 322628B-01 . ed. National Instruments.

Polastre, J., Szewczyk, R., and Culler, D. 2005. Telos: enabling ultra-low power wireless research. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks.* IEEE Press, Piscataway, NJ, USA.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. 1988. *Numerical recipes in C: the art of scientific computing.* Cambridge University Press, New York, NY, USA.

Sang, L., Arora, A., and Zhang, H. 2010. On link asymmetry and one-way estimation in wireless sensor networks. *ACM Trans. Sen. Netw. 6,* 2, 1–25.

Sentilla Corporation 2006. *Tmote Sky Datasheet.*, 1.04 ed. Sentilla Corporation. Available from http://www.sentilla.com/moteiv-endoflife.html [Accessed 18 January 2008].

Wang, L. and Wang, Z. B. 2007. A survey of time synchronization of wireless sensor networks. In *Wireless, Mobile and Sensor Networks, 2007. (CCWMSN07). IET Conference on.* 241 –244.

Zhao, J. and Govindan, R. 2003. Understanding packet delivery performance in dense wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems.* ACM Press, New York, NY, USA, 1–13.

Note

[1]Data fusion consists of the assembly of distributed observations into a coherent estimate of the original phenomenon.

[2]Clock drift is caused by the different frequency of crystal oscilators. The synchronisation precision decays with the time elapsed from the synchronisation instance.

[3]In a so-called MAC layer time-stamping the global time value is inserted into the MAC layer payload after the transmission of the SFD and therefore the impact of the channel access time on the jitter is minimised.

[4]Start of Frame Delimiter (SFD) is the 8-bit value (0xA7) marking the end of the preamble of an IEEE 802.15.4 frame.

[5]Frame Check Sequence (FCS) in an IEEE 802.15.4 MAC frame is a 16-bit International Telecommunication Union Telecommunication Standardization Sector (ITU-T) cyclic redundancy check (CRC) which helps verify the integrity of the MAC frame.

[6]We used variable notation according to this book. For details of the optimised implementation, please refer to the book.

[7]We determined experimentally the maximal value of ΔW_i to be 237 whereas minimum value of P is 32768 when duty cycling period equals 1s.