PhD Dissertation

International Doctorate School in Information and Communication Technologies

DISI - University of Trento

Time Synchronization and Energy Efficiency in Wireless Sensor Networks

Anton Ageev

Advisor:

Prof. David Macii

Università degli Studi di Trento

March 2010

Abstract

Time synchronization is of primary importance for the operation of wireless sensor networks (WSN): time measurements, coordinated actions and event ordering require common time on WSN nodes. Due to intrinsic energy limitations of wireless networks there is a need for new energy-efficient time synchronization solutions, different from the ones that have been developed for wired networks. In this work we investigated the trade-offs between time synchronization accuracy and energy saving in WSN. On the basis of that study we developed a power-efficient adaptive time synchronization strategy, that achieves a target synchronization accuracy at the expense of a negligible overhead. Also, we studied the energy benefits of periodic time synchronization in WSN employing synchronous wakeup schemes, and developed an algorithm that finds the optimal synchronization period to save energy. The proposed research improves state-of-the-art by exploring new ways to save energy while assuring high flexibility and reliable operation of WSN.

Keywords

wireless sensor networks, time synchronization, energy efficiency, time uncertainty

Acknowledgments

I would like to thank my adviser, Prof. David Macii. Without his help, guidance and inspiration this work would not be accomplished.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The context

Recent revolutionary changes in electronic and communication technologies have resulted in the emergence of wireless sensor networks (WSN) – the networks composed of numerous tiny radio devices equipped with miniature sensors. These autonomous devices can be embedded in various objects (and in the future they might be even implanted in living organisms) in order to form intelligent multipurpose distributed systems which acquire sensor data and process it in view of carrying out some desired actions.

Due to their advantageous characteristics (low cost, multi-functionality, small size and mobility), WSN are expected to be widely used in agriculture, industry, transportation, health care and everyday life. However, distinctive features such as limited energy and memory or unreliable communication bring up many problems before scientists and engineers working in this field. One of the crucial questions in this research domain is how to support a certain network functionality requiring identical time on all the nodes in a network. For example, if a WSN monitors a certain time-varying phenomena (e.g. propagation of fire or dangerous gases over a given area) the networked sensors must not only perform data acquisition but also tag the time to each reading. Thus, multiple measurement results collected

from different sensor nodes together with the respective time-stamps can provide valuable information about an observed process (e.g. fire propagation velocity). Also, wireless sensors must be time-synchronized in order to schedule some actions (e.g. wake-up at the same time), to communicate during assigned time slots (e.g. in time division multiple access schemes), to localize and track objects (e.g. using the time difference of arrival of the signal emitted by an object and received by three or more receivers).

The limited performance of time-keeping hardware of a typical WSN node affects deeply all network operations related to time. Indeed, a WSN node generally contains a low-cost crystal resonator that clocks an on-chip timer. In turn, timer values are used by a running program to time-stamp data, to schedule activities, etc. Since the resonator frequency is not identical on different nodes and it changes due to multiple factors (e.g. temperature, crystal aging, noise), the differences between time values grow continuously in time (the drift rate can be in the order of some seconds per day). Besides, random communication delays limit the usefulness of timestamps tagged to radio messages sent by one node to another. Consequently, the uncertainty associated with the time estimation performed by a WSN node grows significantly over time. To tackle this problem, researchers have recently developed various synchronization methods for WSN. Up-to-date literature provides a detailed classification of these methods, many of which are aimed at synchronizing the clocks of network nodes as accurately as possible. Such accuracy-oriented techniques are generally computationally intensive and require multiple communication sessions. A typical synchronization procedure relies on the transmission of several time-stamped radio packets from one node (master or root) to the others. Alternatively, network nodes can exchange the timestamps of reception of one radio packet broadcasted by a common reference node. Each of the aforementioned procedures can be repeated multiple times (periodically or sporadically)

to collect a sufficient amount of information about clock differences. Then some statistical methods can be used, such as linear regression, to compute the relative clock drift rate of every node either with respect to the master or with respect to the other nodes. Also, some round-trip time measurements, requiring additional message exchanges, can be performed to estimate and compensate communication delays. Some synchronization techniques aim at reducing both the computational burden and the traffic related to synchronization, while keeping time uncertainty in WSN within reasonable bounds. These lightweight techniques generally do not use any statistical method to compute the drift rate or even do not compute it at all. In the latter case a network node synchronizes to the time of the other nodes using just the most recent timestamps received from them and does not allow for the errors arising from the continuous growth of differences in clock readings. However, in all WSN synchronization techniques it is essential to use the radio, which is often the most power-consuming component of a wireless sensor node. Therefore, the synchronization takes a considerable amount of energy which could be used otherwise to prolong the node operational lifetime.

In an attempt to decrease the energy consumption, some of the latest research works are focused on adaptive synchronization methods. These methods change some important parameters of the synchronization procedure (e.g. its periodicity, accuracy or complexity) in response to a certain change in the condition of the network or the environment. For example, if the batteries of WSN nodes are almost depleted, synchronization operations (including radio transmissions) can be performed more rarely to save energy and prolong the network lifetime. In this case, the accuracy of timestamps degrades with time, but as soon as some additional energy becomes available (e.g. when the sunlight illuminates the solar cells of the nodes), the synchronization rate can be increased again.

1.2 The problem

Power efficient time synchronization in WSN can be performed in a variety of ways, each of which has its distinctive advantages and drawbacks under certain conditions. Consequently, there is a need to investigate possible trade-offs between different approaches, in order to achieve a reasonable synchronization accuracy and to save energy as well. From this point of view, the crucial problem of WSN domain is to choose between the accurate time synchronization, which usually takes much energy, and energy-saving synchronization, which often results in an increased time uncertainty. This problem has two important aspects explained by the examples given in the rest of this section.

1.2.1 Energy waste during nonadaptive synchronization of WSN

If resonator frequencies of WSN nodes are constant, which is true in the short to medium range of WSN lifetime under stable environmental conditions, the differences between their clock readings can be approximately considered as a linear function of time. Therefore, the rate at which those differences change can be taken as constant and its value, found by some synchronization procedure, can be used to estimate accurately the clock readings of nodes during a long time. On the one hand, this approach saves energy by keeping WSN nodes synchronized without the need to perform frequent synchronization procedures. On the other hand, drift rate estimation may require relatively complex computations involving multiple floating point operations, and exchange of numerous time-stamped radio packets. This results in an intensive use of radio modules and microprocessors, which in turn increases the energy consumption. Besides, if resonator frequencies of WSN nodes change (e.g. due to the ambient temperature variations), the computed drift value becomes out-dated and, if nothing is

done in response to ambient effects, the synchronization accuracy degrades, which also may lead to energy loss. Finally, changing environmental conditions not only worsen the accuracy of nonadaptive time synchronization, but also make it inefficient in terms of energy. For example, if a synchronization procedure is performed at a constant rate and achieves a good accuracy even under temperature variations, it wastes energy when the temperature stabilizes, because in this case the same accuracy could be achieved by less frequent synchronization. Therefore, there is a need for low-power and flexible synchronization methods, that are able to adapt to changing environmental conditions and maintain the target synchronization accuracy at the same time.

1.2.2 Energy waste during idle listening in WSN

Infrequent and lightweight synchronization can save energy at the expense of time uncertainty growth. However, in many network operation scenarios, a large time uncertainty leads to an increase in power consumption. For example, in order to periodically exchange some sensor readings, wireless sensor nodes may use synchronous wake-up schemes to activate their radio modules exactly for the scheduled communication time. Such a way of operation saves energy (since the radio modules are in a sleep mode most of the time) but requires time synchronization between communicating nodes, because their radio modules must wake up simultaneously. In this case, the less accurately the nodes are synchronized the larger the difference between their wakeup times becomes. Therefore the nodes should spend more time on idle listening or waiting until all of them are ready to communicate (i.e. until all radio modules are active), which leads to an increase in energy consumption. This example shows the necessity to find an optimal method that minimizes the energy spent on time synchronization while maintaining time uncertainty withing reasonable limits to assure reliable

communication.

1.3 The solution

In order to solve the problems stated above, at first we performed detailed analysis of uncertainty sources affecting time synchronization in WSN. Then we developed an adaptive time synchronization technique, which monitors the clock readings of WSN nodes and checks whether the differences in those readings exceed a certain threshold value. This value represents the target time uncertainty, and if the actual time differences of most nodes are within the corresponding limits, the synchronization period is extended to save power. Otherwise, the synchronization period is shortened. We proved that this technique may achieve significant energy savings and it works reliably even under rapidly changing environmental conditions such as air temperature.

Also, we have observed that if the maximum difference between resonator frequencies of network nodes is known, the worst possible clock differences can be estimated as a function of the synchronization period for periodic synchronization schemes. These maximum differences directly affect the lower bound of the waiting time necessary to ensure that all radio modules are active in synchronous wake-up strategies and, as a consequence, these differences affect the power consumption of WSN nodes. We have derived an expression for the optimal synchronization period minimizing the power consumption of a generic WSN node, which periodically performs measurements and exchanges the results with its neighbors. We implemented a power saving time synchronization algorithm, tried it on real WSN nodes and proved experimentally the correctness of the expression mentioned above.

1.4 Innovative aspects

We propose a novel approach to the power efficient time synchronization for resource-limited WSN. This approach is based on a comprehensive study of the relationship between energy consumption and time uncertainty. We developed a new adaptive time synchronization algorithm, which saves energy in WSN. Also, we investigated the benefits and cost of WSN time synchronization in terms of power consumption, and derived an optimal synchronization period, which allows to save energy in the applications employing radio scheduling. The proposed research significantly improves the state-of-the-art because it paves the way to new low-power management techniques for WSN, while assuring high flexibility and timely behavior.

1.5 Structure of the thesis

The further text of the dissertation is structured as follows. Chapter 2 outlines the state-of-the-art of time synchronization in WSN with an emphasis on power efficiency issues. Chapter 3 presents a detailed analysis of uncertainty sources affecting time synchronization in WSN as well as the results of quantitative estimation of different uncertainty contributions. Chapter 4 describes the developed adaptive synchronization method, the power efficient time synchronization algorithm based on that method and the experimental results. Chapter 5 shows how the time uncertainty affects the power consumption of wireless sensors using synchronous wake-up schemes. Then the chapter describes the derivation of the optimal synchronization period resulting in the minimum power consumption and presents the experimental results. Finally, Chapter 6 summarizes the research work and emphasizes its usefulness for WSN applications.

Chapter 2

State of the art

2.1 Time Synchronization in WSN

Emerging and anticipated applications of wireless sensor networks suggest that it is essential to have a common time on the nodes of such networks. Home control, building automation, industrial automation, medical, environmental, military and scientific applications need networks of low-cost and low-power wireless sensors which periodically take measurements, process and transmit them. In most cases these measurements must be taken synchronously or time-stamped and arranged in time. Moreover, often the time itself must be measured by wireless nodes in order to obtain some valuable information. Depending on the application, the usage of time can refer to three main cases: time-related computations, coordinated actions and data logging.

2.1.1 Time-related computations

Time synchronization is essential in WSN that observe a certain time-varying phenomenon [59, 24]. In this case different wireless sensors measure one or more physical quantities related to that phenomenon, and the time is one of those quantities. Observation results can also be represented

by some other quantities, which are derived from the measured quantities. Obviously, the accuracy of measurements affects the correctness of the calculated result. Since the clocks of wireless nodes can be regarded as sensors measuring time, the time synchronization is nothing else as the calibration of those sensors [52]. Such calibration enables to collect accurate and consistent measurements of time from different network nodes, and finally to obtain truthful observations. For example, in a common localization method based on the acoustic ranging, a network node produces a sound, which is received by the other nodes. The nodes have to compute the sound propagation time and multiply it by the sound speed in order to estimate the distance to the sound source. Clearly, such computations can be done only if the clocks of the nodes are synchronized.

2.1.2 Coordinated actions

Time synchronization is necessary if many WSN nodes have to perform some actions simultaneously at a scheduled time. For example, in order to save energy, the radio modules of wireless nodes are generally activated only for the period during which the message exchange is planned [54, 36]. When a communication session is finished, the nodes have to switch off their radios and schedule the time of the next communication session. Without time synchronization, the nodes fail to activate their radios at the same time, which may lead to message losses. Also, multiple wireless sensors, deployed in a large area, may need to perform simultaneous observations of some environmental effect. Such observations allow to take a "snapshot" of the values of a certain quantity in many spots of the area at a certain time. Of course, simultaneous activation of sensors on different nodes requires the time synchronization.

2.1.3 Data logging

Time synchronization allows to arrange in time the readings of multiple wireless sensors, and use these readings to reconstruct the sequence of events happened with the monitored object [75, 19, 30, 37, 41]. For example, wireless sensor nodes can be deployed on some structure and continuously measure ambient vibrations [32, 66]. Analysis of time-stamped sensor readings facilitates preventive maintenance and early detection of dangerous changes in the state of the structure. Besides, if some damage has already happened, time histories based on the sensors observations can help to discover the cause of the damage and prevent it from happening in the future. Another example of WSN data logging is the case when a WSN monitors the state of health of patients in a hospital, home or outdoors [43, 67]. The knowledge of how their vital signs vary in time can help to take right and timely triage decisions. Moreover, the time-referenced data can be used in patients records.

2.1.4 The need for special synchronization methods in WSN

Maintaining the common time on wireless sensor nodes entails many difficulties. First, a typical WSN node reads time values from the timers of its on-board microcontroller or microprocessor. The timers are usually clocked by a low-cost crystal oscillator, whose frequency differs from the nominal frequency, and this difference is not the same on network nodes. This is a major cause of continuously growing differences between clock readings of different nodes. Moreover, the oscillator aging, changes in temperature and power supply voltage, mechanical shock, vibration and ionizing radiation result in significant frequency instability. Of course, network nodes can periodically adjust their clocks, e.g. by setting their timers to the time values received from some reference clock. However, wireless communica-

tion delays may be in the order of some milliseconds and include random components, which makes the simple clock adjustment inefficient. Thus, multiple factors do not allow to achieve an identical time on the network nodes in a simple way. In order to keep the differences between clocks of WSN nodes within acceptable bounds, special WSN synchronization methods have to be designed and tested.

Time synchronization in WSN is an active area of study and every year researchers propose some new synchronization algorithms. In fact, time synchronization is among the top ten research directions in WSN domain [60]. Besides, the need to synchronize sensor nodes is related to the other very important issues in wireless sensor networks. Target tracking and localization are among notable cases of research related to time synchronization, which is proved by the examples of WSN applications given above.

2.2 Clocks in WSN

A typical WSN node consists of five main hardware modules: (1) microcontroller, (2) radio transceiver, (3) one or more sensors, (4) memory chip and (5) battery (Fig. 2.1). These modules are mounted on one or more circuit boards together with some auxiliary mechanical and electronic components. Time is kept by the timer module and oscillator circuit, which are among the on-chip components of the microcontroller (Fig. 2.2). The oscillator circuit, generally driven by an off-chip crystal resonator, generates the clock signal for the timer module. The timer module includes the timer/counter register (we will call it timer for brevity), which increments or decrements with the rising edge of the clock signal, and generates interrupts on overflow. In addition, the timer module may have one or more capture/compare registers to record the time of some events or to generate

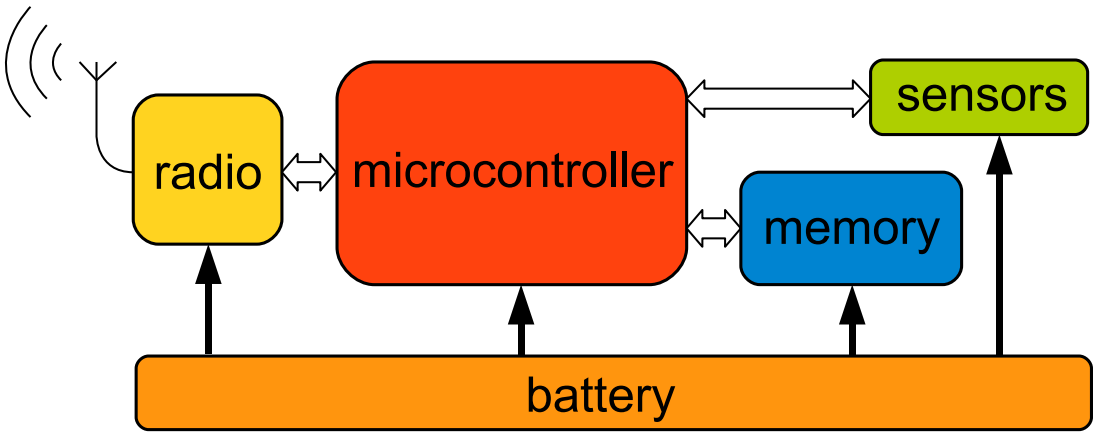


Figure 2.1: Architecture of a typical WSN node. Shaded arrows represent energy transfer, unshaded arrows represent data and command flow.

interrupts at specific time intervals, and pulse width modulation module to generate square wave signals with various average power. The timer actually counts time of the wireless sensor node, and timer values can be directly read by a certain instruction in the program, running on the microcontroller. Also, the timer is used to schedule tasks, and WSN nodes usually wake up on the timer interrupt, execute some routine and then may enter a low power mode. However, the width of a typical timer may be not enough to count time throughout the node operation. For example, if a 16-bit timer is clocked by a 32768 Hz signal (the frequency of the crystal resonator often used in wireless sensor nodes) it overflows every 2 seconds. Naturally, this interval is extremely short by comparison with possible network lifetime, which may vary from a few days to several months. To keep time longer, its value should reside in a storage cell composed of a timer register (for low-order bits) chained with a variable (for high-order bits). The value of the variable should be incremented by the interrupt generated when the timer register overflows. Such a combined timer can be accessed by means of a timer abstraction often provided in operating systems for WSN. For example, TinyOS component library includes timers with vari-

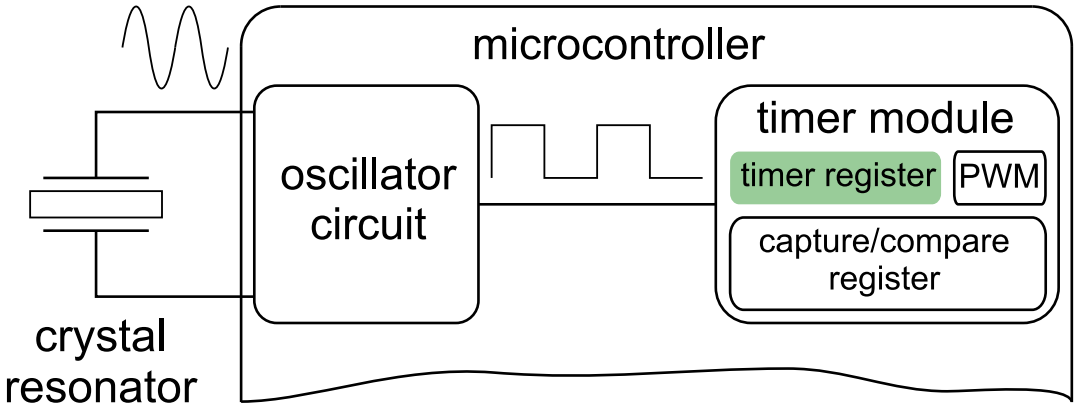


Figure 2.2: Clock system of a typical WSN node.

ous widths (e.g. 32 or 64 bits), while the microcontroller of TelosB (a well known WSN module, which can run TinyOS) has 16-bit timers [45, 65].

In many WSN papers the timer value is often meant by the terms "clock reading" or "clock indication" or "time value" or "time-stamp", and all of which are used interchangeably [61, 52, 16].

2.3 Synchronized clocks in WSN

The state of being synchronized in WSN is generally associated with the common time on two or more network nodes. The notion of common time may refer to the case, when the difference between the time measured by two different nodes is below a given target uncertainty. In this case, if the accuracy requirements are strict, either the time keeping devices must be of a very high quality (e.g. sapphire resonators) or the clock readings must be always observed and adjusted. Of course, such a way of maintaining the common time is redundant and costly. Alternatively, dissimilar clock readings of different nodes can be transformed in a way which enables to locate them on a common time scale. This transformation can be done by any wireless node, which is able to estimate the clock offsets of the other nodes with respect to its own clock within a target uncertainty. In

many research papers related to WSN, synchronization is referred to as the ability to perform a time transformation in order to achieve a common time scale.

2.4 Types of synchronization in WSN

The synchronized state of a wireless sensor network can be achieved in many ways, depending on the application requirements, available devices, energy and memory resources. Classification of time synchronization in WSN is often represented by several pairs of contrasting distinctive features [52, 61].

• Time synchronization can be proactive or reactive. Proactive time synchronization is performed continuously in order to support the synchronized state of the network, which may be necessary for some future activities. For example, a single reference node can periodically disseminate its time to the other nodes, which will allow them to estimate their clock offsets with respect to the reference time. Reactive synchronization, on the contrary, is done only when it is required. For example, unsynchronized sensor nodes can record the time of a certain event and send the time-stamps to a sink. After the data collection, the sink node can perform some synchronization procedure and transform the time-stamps of the sensor nodes to its own time. This is an example of what it is called post-facto synchronization [16].

• WSN nodes can be synchronized either to some external time reference, or by means of some intra-network synchronization procedures. A GPS receiver can serve as an external source of time for a wireless network, where the time values can be disseminated by a special node attached to the GPS receiver. In this case, network nodes are able to transform their time values to the reference time values and vice versa, thus achieving the

time synchronization. Alternatively the network can be synchronized internally, i.e. using the clock of some node as a reference clock.

• The synchronization can be network-wide or related only to some part of a network. In the latter case, almost all nodes may be in a low-power mode, while several special nodes stay synchronized and operate in an active mode. These special nodes wake up the other nodes and supply them with time information, if necessary. Also, the synchronization may be required only for those sensor nodes which actually observe some event (e.g. rise in temperature).

• WSN nodes can be synchronized to a certain guaranteed accuracy. Such a deterministic synchronization takes a significant amount of energy, which is often reasonable, especially in safety-critical applications. Alternatively, a required synchronization accuracy can be achieved with a certain probability. The probabilistic synchronization allows to save power at the expense of an increased chance of having unsynchronized nodes.

2.5 Basic synchronization procedures in WSN

2.5.1 Time transfer

A fundamental procedure which is performed in any WSN synchronization algorithm is the transfer of time value from one node to another. The simplest way to do this is to read the clock, put the time value in a radio packet and send it. Upon the reception of a time value t_A from node A, node B can record its local time t_B, thus obtaining a pair of time-stamps. If the communication between the two nodes was instantaneous, node B could estimate the difference d between its time and the time of node A by subtracting one time-stamp from the other:

$$d = t_B - t_A \tag{2.1}$$

However, time-stamping operations as well as communication and processing tasks take some time, which can be some orders of magnitude larger than the clock period (e.g. communication may take 10 ms while the clock cycle can be 1 μs). Therefore, in most cases, one simple time transfer is not enough to synchronize the nodes, unless the accuracy requirements are very loose.

2.5.2 Pair-wise synchronization

Pair-wise synchronization procedure (also called round-trip synchronization) is widely used to synchronize two WSN nodes (Fig. 2.3). During his procedure, node A sends a packet with a time-stamp t_1 to node B. Upon the packet reception, node B records its time t_2. This value is equal to the sum of t_1, communication delay C and the offset between the clocks of the nodes d:

$$t_2 = t_1 + C + d \tag{2.2}$$

Then node B sends a response packet with a time-stamp t_3 to node A. This second packet contains the value t_2 as well. Finally, node A makes a time-stamp t_4 when the second packet is received. If the communication latency is symmetric, i.e. equal in both directions, the value of t_4 is given by:

$$t_4 = t_3 + C - d \tag{2.3}$$

Subtraction of (2.3) from (2.2) allows to find the clock offset d:

$$t_4 - t_2 = t_3 + C - d - t_1 - C - d \tag{2.4}$$

$$d = \frac{t_3 - t_1 - t_4 + t_2}{2} \tag{2.5}$$

Also, the sum of (2.3) and (2.2) can be used to calculate the communication delay C:

$$t_4 + t_2 = t_3 + C - d + t_1 + C + d \tag{2.6}$$

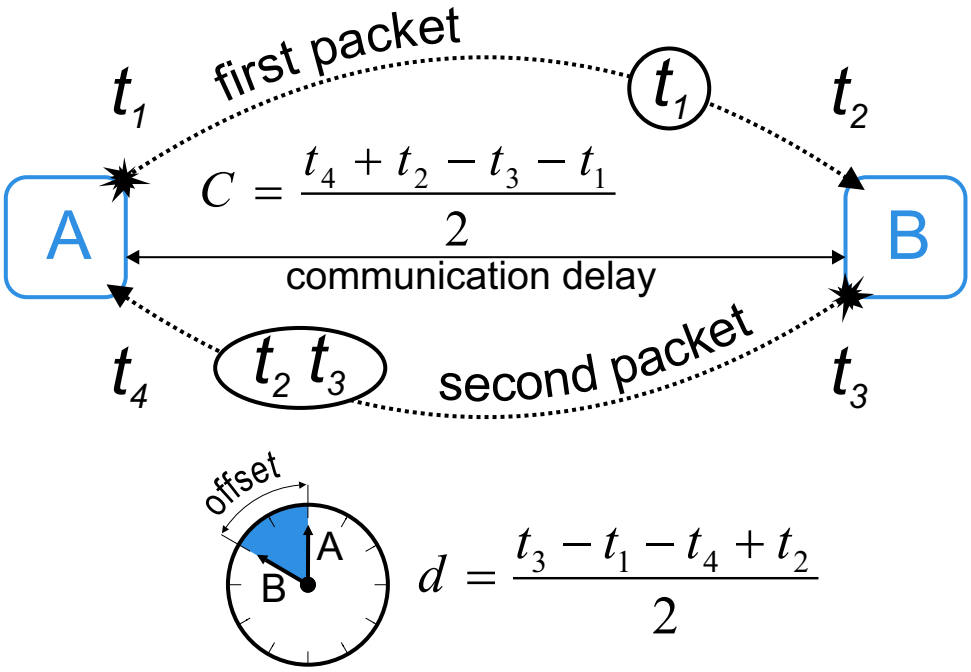


Figure 2.3: Pairwise synchronization.

$$C = \frac{t_4 + t_2 - t_3 - t_1}{2} \tag{2.7}$$

After the abovementioned message exchange is done, node A has all values necessary to find d and C. As soon as the value of d is known, node A can estimate the time of node B.

2.5.3 Reference broadcasting

Pair-wise synchronization assumes that the communication delay C is the same in both directions and the clock offset d is constant during the message exchange. However, this is not necessarily true because communication delays have random components (which will be discussed in detail in the following chapters) and the absolute values of clock offsets continuously grow due to the difference between oscillator frequencies of the nodes. Therefore, there is a significant uncertainty on the offset value computed by (2.5).

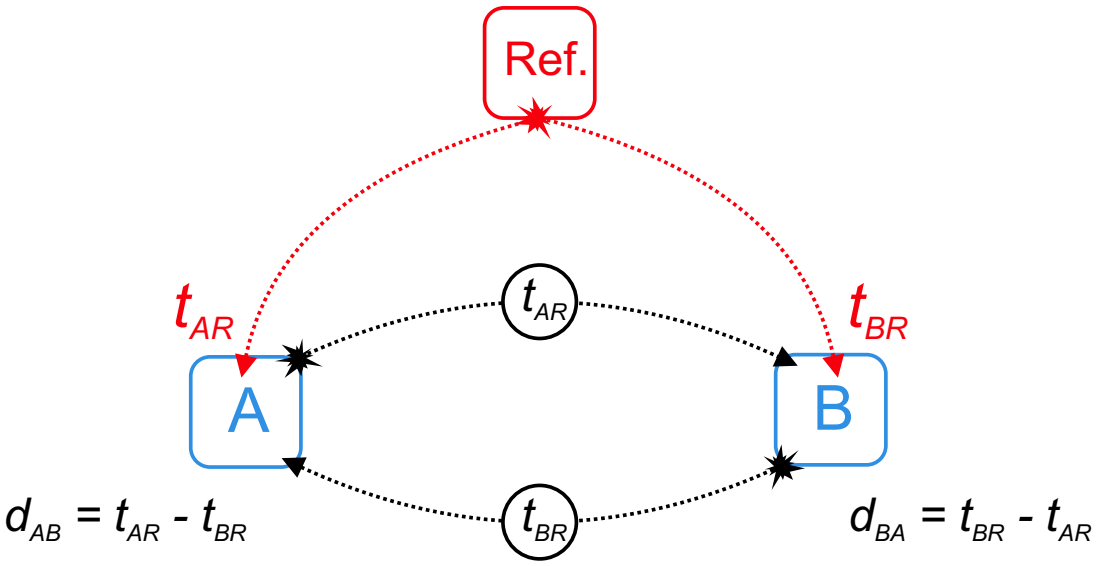


Figure 2.4: Reference broadcasting.

Such uncertainty can be reduced significantly by means of another well known WSN synchronization procedure, often referred to as *receiver-receiver synchronization* or *reference broadcasting*. A group of WSN nodes, which follow this procedure, have to receive one radio message from a special reference node. The receivers record their time values when the reference message arrives, and then each of the receivers broadcasts the corresponding time-stamp to all the nodes within its communication range. After reception of such time-stamps, a node A can estimate the offset of its clock with respect to the clock of some other node B as follows:

$$d_{AB} = t_{AR} - t_{BR} \tag{2.8}$$

where the time values t_{AR} and t_{BR} are recorded by node A and B respectively, and both these time-stamps correspond to the same event – the reference message reception (Fig. 2.4). This offset computation is much more accurate than in the pair-wise synchronization, because the corresponding time uncertainty is not affected by random delays at the sender site. However, there is still a significant time uncertainty, since the mes-

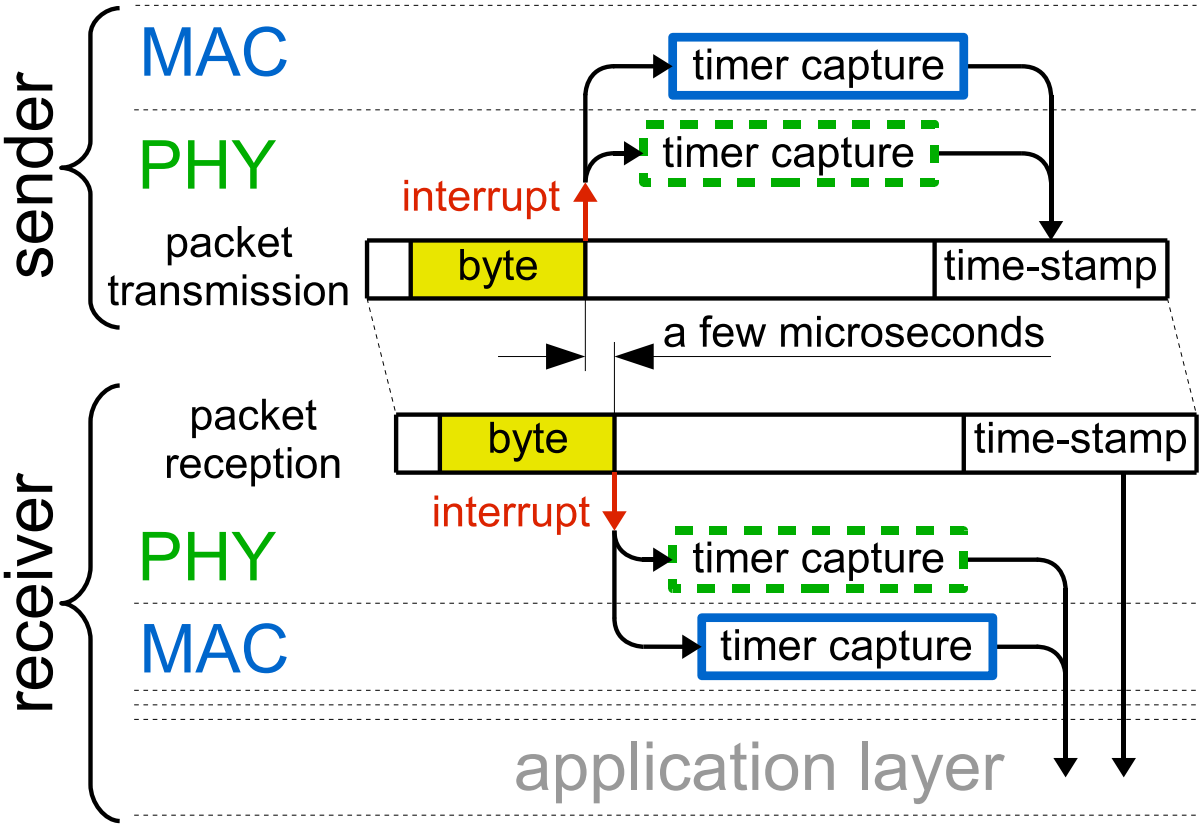


Figure 2.5: Time-stamping of a radio packet at MAC and physical layers.

sage reception takes some time, which changes from node to node. As a consequence, the absolute values of the two relative clock offsets, computed by any pair of receivers, differ.

2.5.4 MAC-layer time-stamping

The uncertainty related to communication can be further reduced by means time-stamping performed at the Media Access Control (MAC) sub-layer [22, 42, 35]. In this procedure, a sending node records the time of some event that happens at the beginning of the packet transmission, and then appends that time value to the packet, while the latter is being transmitted as shown in Fig. 2.5. That event is usually a signal, generated by the radio module, when the transmission of a certain byte is completed (e.g. SFD

signal of CC2420 radio [63]). The timer value is recorded by a special interrupt handling routine implemented at the MAC-layer of the sending node. Then the same routine sends the recorded time-stamp to the physical layer, where the time-stamp is attached to the end of the packet, or inserted somewhere after the aforementioned byte. This happens, of course, after this byte is completely transmitted. A receiver records its local time when the reception of the same byte is completed, which happens only a few microseconds after the byte has been transmitted by the sender. When the whole packet is received, the receiver has a pair of time-stamps, which are taken almost simultaneously on the two nodes. Therefore, the receiver can estimate its clock offset with respect to the sender to a high accuracy simply by subtracting one time-stamp from the other as in (2.1). Thus, many WSN nodes can be accurately synchronized to the reference time if they receive and time-stamp only one radio packet, containing a MAC-layer time-stamp of the node with the reference clock.

2.5.5 Physical layer time-stamping

Time-stamping at the MAC-layer often involves modifications in the communication protocol stack to meet the specific application requirements. For example, the well known WSN operating system TinyOS allows to time-stamp radio packets at the MAC sublayer using 32-bit timer values. If the time-stamps of a different length are required, it is necessary to change the corresponding TinyOS modules (e.g. CC2420TransmitP.nc, used to control transmission operations of CC2420 radio). This approach is invasive to the original implementation of MAC sublayer, and it may lead to the change in execution time of some important operations. Alternatively, time-stamps can be done at the physical layer, without modifying the communication protocol stack, as described in [18]. In this case, the radio must have an on-chip timer, whose value is captured by the radio

(dashed rectangles in Fig. 2.5) and can be read by the MCU of a WSN node (e.g. as in Freescale MC13192 transceiver [20]). Physical layer time-stamping is generally more accurate since it does not need interrupt service routines that may cause jitter.

2.5.6 Clock drift estimation

WSN nodes exhibit different clock speeds due to the differences between the frequencies of their crystal oscillators, which results in continuous growth of the absolute values of clock offsets. Therefore, the uncertainty of time synchronization begins to grow immediately after two nodes A and B compute their clock offsets. This causes the need to estimate time differences periodically, and the accuracy of the resulting synchronization typically depends on how often that estimation is done. More efficient synchronization can be performed under the assumption that the trend of the offset growth is linear. In this case, for example, it is possible to estimate the change of the time offset $d = t_B - t_A$ per unit of time of node B. This quantity is the relative clock drift rate ν, which can be found using just two values of the clock offset d_1 and d_2 observed recently at t_{B1} and t_{B2} according to the time of node B:

$$\nu = \frac{d_2 - d_1}{t_{B2} - t_{B1}} = \frac{(t_{B2} - t_{A2}) - (t_{B1} - t_{A1})}{t_{B2} - t_{B1}} \tag{2.9}$$

If ν is known, any time-stamp t_{BX} of node B can be transformed to the corresponding time t_{AX}:

$$t_{AX} = t_{BX} - d_X = t_{BX} - d_2 - \nu(t_{BX} - t_{B2}) \tag{2.10}$$

where the offset d_X is computed as the sum of the most recently known offset d_2 and the estimated offset change during the respective time interval $t_{BX} - t_{B2}$ according to the clock of node B (Fig. 2.6). There is always some uncertainty in any time measurement performed by a WSN node, which

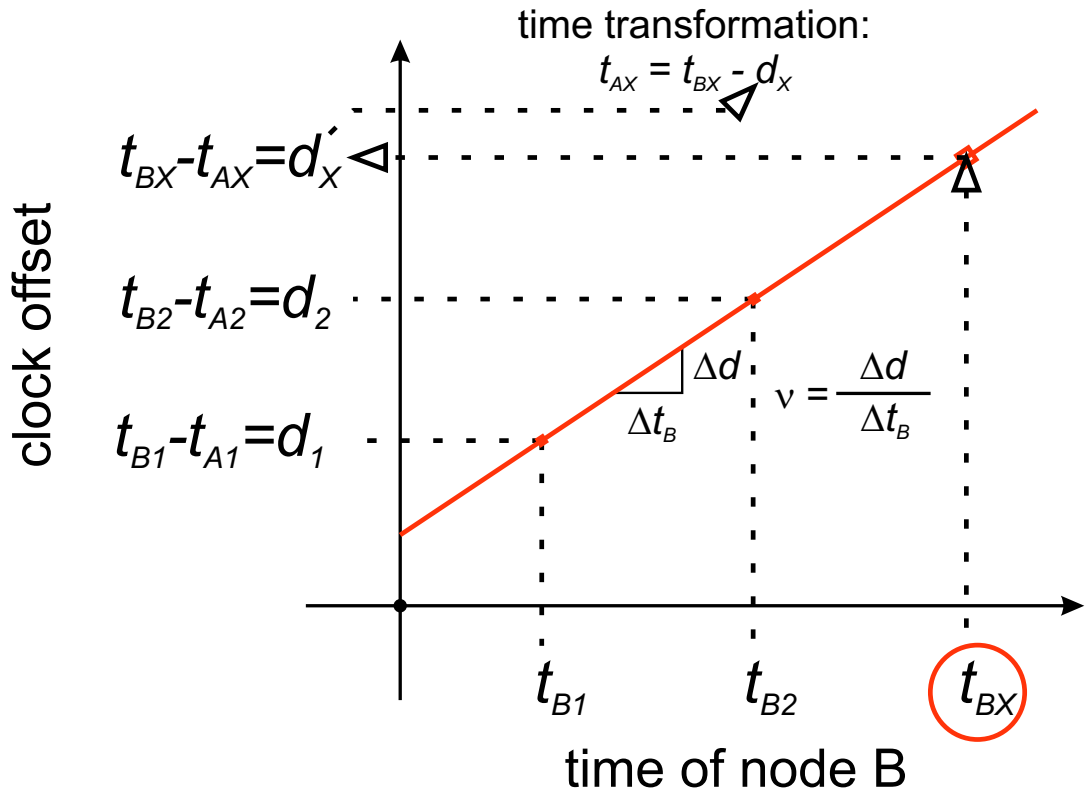


Figure 2.6: Time-stamp transformation using the relative drift rate.

affects the accuracy of the computed relative drift rate. This issue can be addressed by means of linear regression, which is indeed used in recent high-accuracy time synchronization algorithms for WSN [17, 42]. In order to synchronize a WSN node, this technique requires several pairs of time-stamps, which are obtained after the node receives several synchronization messages from a time reference over some time interval. Time stamps are used to compute the corresponding clock offsets, and then the linear regression is performed on this set of values to build a best-fit line (Fig. 2.7). The best-fit line is used to predict offset values in the future, thus keeping the clock of the node synchronized to the reference clock.

If the oscillator frequencies of WSN nodes were constant, the difference between the clock readings of any two nodes would grow at the same rate.

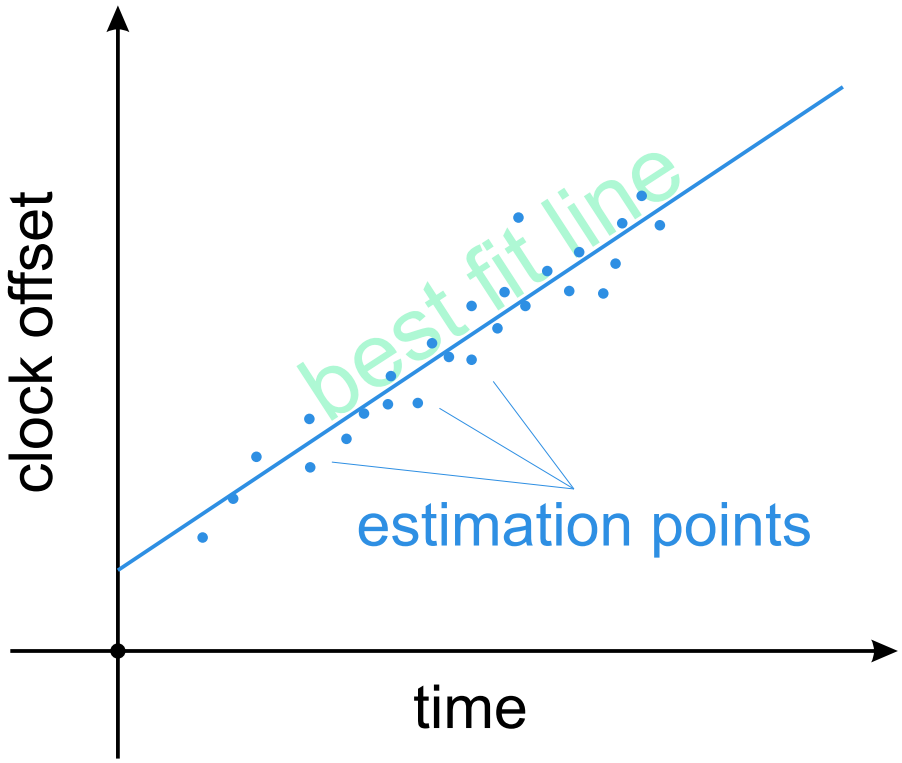


Figure 2.7: Best-fit line.

In this case it would be sufficient to find the clock drift rate once and then always use it to transform the reading of one clock to that of the other, thus maintaining the common time. However, due to the aging and variable environmental factors, the crystal oscillator frequency changes over time. Therefore, whatever accurate method is used to synchronize WSN nodes, the synchronization procedure must always be repeated after some time.

2.6 Up-to-date methods of WSN synchronization

Traditional synchronization protocols for wired networks [e.g. Network Time Protocol [44] (NTP) or Precision Time Protocol [3] (PTP)] as well as some specific wireless protocols (such as IEEE 1588v2 [3] and IEEE 802.1AS [23]) are usually not suitable for WSNs due to the limited hard-

ware and energy resources available on typical WSN nodes. In order to address such problems, in recent years several WSN-specific synchronization protocols have been proposed. However, the most known approaches to time synchronization in WSN are focused on achieving the best possible synchronization accuracy (in the order of some microseconds) for the whole network without considering energy saving.

2.6.1 High-accuracy synchronization methods

• Time-Stamp Synchronization (TSS) protocol [51], proposed by Römer, transforms the time of one network node to the time of another node whenever they exchange time-stamped radio messages. After the reception of a message with a time-stamp, a network node estimates the time-stamp age, i.e. the time elapsed from the moment when the time-stamp was generated by the originator. Then the node subtracts the time-stamp age from the message arrival time, which is recorded using the receiver local clock. The result of this subtraction is the time-stamp value converted to the receiver time. If a message goes through a multi-hop path, the time transformation is performed at each hop. In this case, the time-stamp age is the sum of the total time during which the time-stamp resides on intermediate nodes along the path and the corresponding communication delays between each pair of adjacent nodes along the multi-hop path. The first component of that sum consists of delays measured by local clocks of intermediate network nodes. The upper bound ν_{max} of the clock drift rate is used to transform the time intervals measured by one node to the timescale of another. The second component comprises the communication delays estimated by means of the pair-wise message exchange scheme, as described in Subsection 2.5.2. The TSS operation was verified in a wired network with $\nu_{max} = 1$ ppm. The average uncertainty of time-intervals for adjacent nodes was 200 μs and increased by 200 μs per each additional

hop. TSS protocol synchronizes multi-hop networks with highly dynamic topologies at the expense of low overhead. However, the synchronization uncertainty tends to increase unpredictably with the number of hops.

• Reference-broadcast synchronization (RBS) [17] proposed by Elson et al., is based on the procedure described in detail in Subsection 2.5.3: a special reference node sends radio messages to its one-hop neighbors, which time-stamp the messages upon reception, and then exchange the time stamp values. Every node uses these values to compute instantaneous relative offsets of its clock with respect to the clocks of the other nodes (except the reference node). The nodes also compute their relative clock drift rates by means of least-square linear regression. The knowledge of the relative clock drift rate allows to transform the time of one node to that of another node even in the absence of communication between them during a long term. RBS divides multi-hop networks into one-hop clusters with mutual gateway nodes transforming the time of one cluster to that of another cluster. As experiments showed, RBS synchronized Berkeley Motes within 11 μs. Besides, the protocol operation was verified on the 802.11 network composed of Compaq IPAQs running Familiar Linux: RBS synchronized that network within 6.29 $\pm$6.45 μs, and within 1.85 $\pm$1.28 μs using kernel time-stamping. The synchronization accuracy achieved by RBS is close to the clock resolution. However, the protocol may cause high computation and communication overhead, since all the receivers of the reference message have to exchange their time-stamps and build the corresponding best-fit lines. The authors of RBS first introduced the concept of receiver-receiver synchronization, which was used later in many other WSN synchronization protocols.

• Timing-sync protocol for sensor networks (TPSN) [22], devised by Ganeriwal et al., works in two phases. During the first, *level discovery phase*, a spanning tree is created in the network, where every node is as-

signed a level number. The root of the tree has a level 0 and functions as a reference node. At the beginning of the level discovery phase, the root broadcasts a *level discovery packet*, which contains the root ID and level number. The nodes within the root communication range receive that packet and assign themselves a level number, grater by one than the level number in the received packet. Then those nodes broadcast new level discovery packets with their own level numbers. This process continues until all network nodes have their level numbers. In the second, *synchronization phase*, the root broadcasts a special packet to initiate synchronization. This packet is received by the immediate neighbors of the root, and then each of the neighbors initiates an exchange of two radio messages with the root after some random time interval. During this exchange, a node sends a message to the root, and the root sends back an acknowledgment. Both messages are time-stamped just before transmission and upon reception, and the time-stamp values are used by the node to compute its clock offset with respect to the root using the procedure described in Subsection 2.5.2. The neighbors of the node overhear its communication with the root, and then, after a random time interval, get synchronized to the node in a similar manner, i.e. by means of pair-wise synchronization. This process continues until all network nodes are synchronized. Experiments with Berkeley Motes proved that TPSN synchronizes two adjacent network nodes within 16.9 μs. TPSN can synchronize large networks to a high accuracy. However, the protocol requires the hierarchical network organization, maintaining of which takes much energy, especially when network nodes are mobile.

• Flooding time synchronization protocol (FTSP [42], designed by Maróti et al., floods the whole network with messages, which contain values of the global time, i.e. the time of an elected synchronization leader. The synchronization leader periodically broadcasts synchronization messages

containing its time-stamps. A network node records its local time upon reception of a synchronization message (using MAC-layer time-stamping), thus obtaining one reference point, which contains two time values: global time and local time. When a node collects a sufficient number of reference points, it computes the drift rate of its clock with respect to the leader clock using linear regression (see subsection 2.5.6). This node can estimate the global time and transmit it to the other nodes which are not yet synchronized. The nodes within the leader transmission range receive global time values directly from the leader, the distant nodes receive estimations of the global time from the other synchronized nodes. FTSP operation was experimentally verified on the network composed of 60 Mica2 motes. In 14 minutes, the 6-hop network was completely synchronized to the average accuracy of 3 μs resulting in a 0.5 μs per hop accuracy. FTSP achieves a high synchronization accuracy and it is tolerant to heavy changes of the network topology. However, multiple transmissions of synchronization messages, performed by many nodes, may cause collisions and, as a consequence, significant communication overhead.

2.6.2 Light-weight synchronization methods

Researchers pointed out that the synchronization accuracy to within fractions of a second was often sufficient for many sensor network applications [69]. Besides, an accurate time synchronization needs more complex computations and more frequent network communication, which leads to an increased energy consumption. Since energy is a scarce resource, especially in wireless sensor networks[14, 60], alternative energy-efficient time synchronization schemes are required. In this respect, some WSN synchronization algorithms have been developed recently, which improve power efficiency by reducing the synchronization overhead related to communication and computation [58, 69, 15, 13, 77, 57, 74].

• Tiny-sync and mini-sync (TS/MS) protocols [58], developed by Sichitiu and Veerarittiphan, form a hierarchy of network nodes where each parent and child can exchange time-stamped radio messages. The oscillator frequencies of network nodes are considered to be approximately constant during time intervals ranging from some minutes to some hours. As a consequence, the relationship between clock values t_1 and t_2 read at the same time on any two nodes is assumed to be linear:

$$t_1 = a_{12}t_2 + b_{12} \tag{2.11}$$

where a_{12} and b_{12} are the relative drift rate and relative offset of the clocks of the two nodes, respectively. If node 1 knows the values a_{12} and b_{12}, it can estimate the time of node 2 using (2.11). In order to do this, node 1 sends a time-stamped probe message at t_0 to node 2. Node 2 receives and timestamps the probe message with its time t_b and immediately sends it to node 1 in a response message. Node 1 receives and time-stamps the response message with its time t_r (Fig. 2.8). Thus, each message exchange gives to node 1 one data point composed of three values: t_0, t_b and t_r. Since the time-stamp t_b was registered after t_0 and before t_r, the time t'_b which could be observed by node 1 when node 2 registered t_b is limited by t_0 and t_r. Since t'_b could also be computed by (2.11), the following inequalities hold:

$$t_0 < a_{12}t_b + b_{12} \tag{2.12}$$

$$t_r > a_{12}t_b + b_{12} \tag{2.13}$$

At least two data points allow to compute lower $\underline{a_{12}}$ and upper $\overline{a_{12}}$ bounds of a_{12} and lower $\underline{b_{12}}$ and upper $\overline{b_{12}}$ bounds of b_{12}. This can be represented graphically as two lines with the slopes corresponding to the minimum and maximum values of a_{12} and the offsets corresponding to the minimum and maximum values of b_{12} (Fig. 2.9). The values of a_{12} and b_{12} are computed

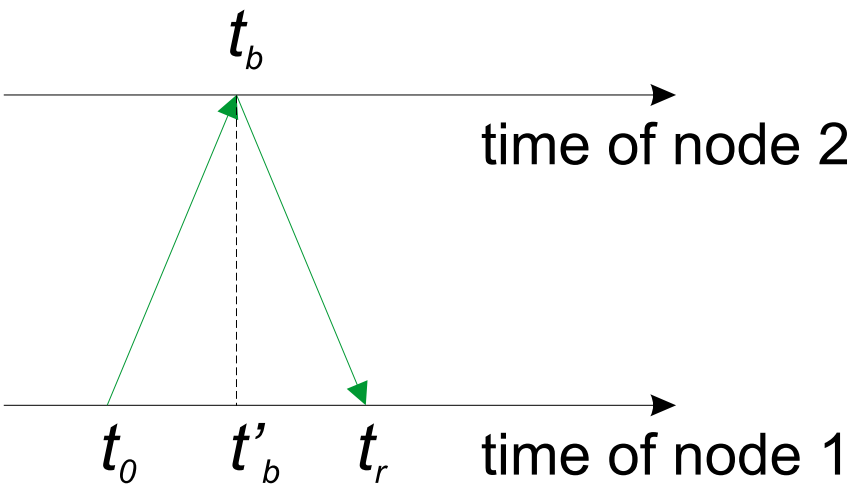


Figure 2.8: TS/MS operation: probe message exchange.

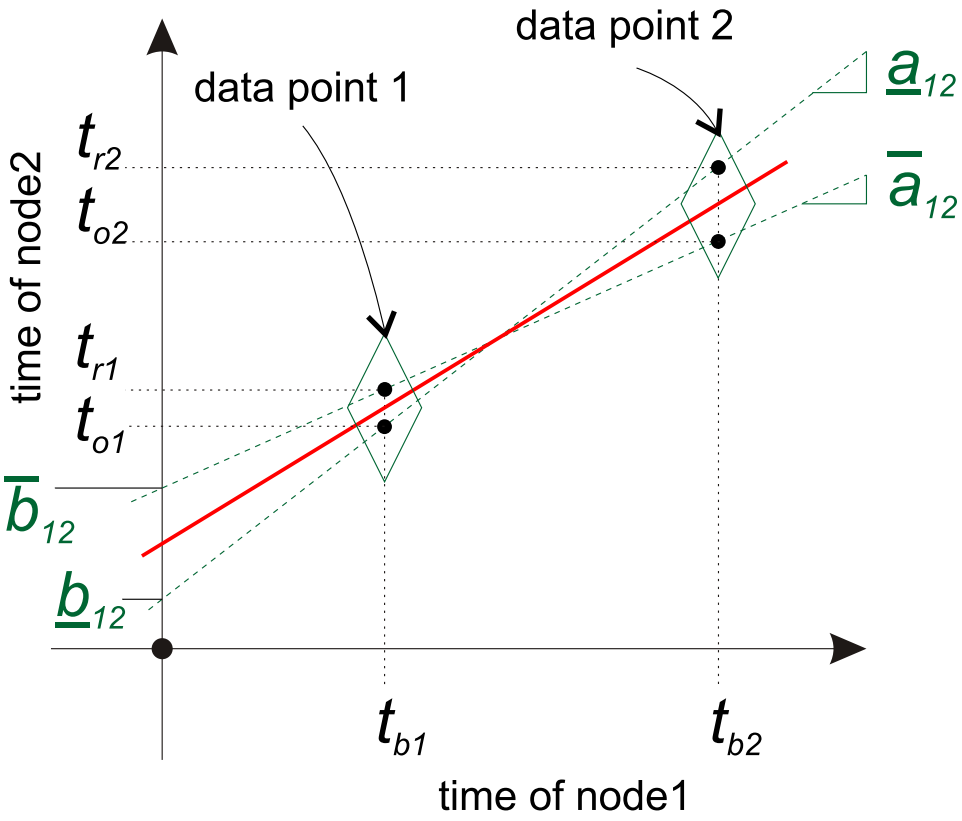


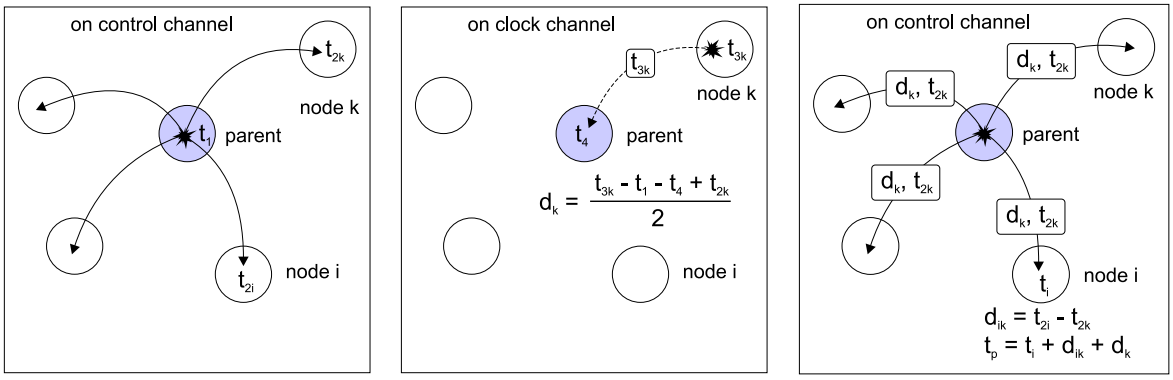
Figure 2.9: Bounds on a_{12} and b_{12}.

as the average values of the aforementioned extreme values (this is represented by the line with the average slope and offset). Node 1 receives several data points, and when a new data point is received, it is compared with the existing ones and one of the data points is discarded. Only the data points providing the best-fit line are kept in the memory. During the probe message exchange, if node 2 does not respond immediately, which will be most likely in real communication, it responds at time $t_{b2} > t_b$. This does not affect the correctness of the approach described above, but it just results in two data points: $[t_0,\ t_b,\ t_r]$ and $[t_0,\ t_{b2},\ t_r]$. Mini-sync uses more data points to achieve a higher synchronization accuracy at the expense of an increased energy consumption due to a larger number of communication events. Tiny-sync uses minimum number of data points, and saves power at the expense of reduced synchronization accuracy. Two experiments with an 802.11b multi-hop network were used to assess the performance of Tiny-sync. In the first experiment, two computers within one hop exchanged probe messages once per second during 83 minutes, which gave almost 5000 data points. The resulting offset and drift bounds were ± 945 μs and ± 0.27 ppm, respectively. The second experiment was performed similarly, but two computers were five hops away. This resulted in the offset bound ± 3.232 ms and the drift bound ± 1.1 ppm. TS/MS algorithms provide tight bounds on the relative drifts and offsets of network nodes at the cost of very low computational, memory and communication overhead. However, TS/MS use a hierarchical network organization and therefore is vulnerable to the network topology changes.

• Lightweight tree-based synchronization (LTS) scheme [69], developed by van Greunen and Rabaey, saves power by means of fewer radio messages and less complex computations necessary for the time synchronization. Two LTS algorithms perform periodic synchronization by exchanging a pair of time-stamped messages along the edges of a spanning tree, i.e. the

pairwise synchronization is performed between parents and children in the tree (see subsection 2.5.2). In the first, centralized algorithm, a reference node, which is the root of the tree, synchronizes with its single hop neighbors, then they synchronize with their children etc., until all leaf nodes of the tree are synchronized. The reference node must provide an accurate time, and must resynchronize the network periodically. The resynchronization period is calculated by the reference node depending on the desired accuracy, upper bound of the clock drift rate and depth of the tree. The upper bound of the clock drift rate in the network is considered to be known in advance. In the second, distributed algorithm, network nodes decide on their own whether they need to be synchronized. A node that requires synchronization sends a request to the closest reference node. If these two nodes are not within a single hop from each other, the request goes along some multi-hop route through intermediate nodes, which get synchronized too. The LTS functionality was verified using Omnet++, a discrete event simulator. In simulations, LTS run during 10 hours on 500 nodes placed in 120×120 m area with the target accuracy of 0.5 s. The average clock offset in the network before synchronization did not exceed 0.4 s, i.e. it was within the accuracy bounds. The centralized LTS scheme is more efficient for the network-wide synchronization, while the distributed scheme better saves power when only a part of network nodes need to be synchronized. LTS requires the construction of a spanning tree, which takes much energy if the network topology changes.

• Tsync synchronization service [15], proposed by Dai and Han, includes two components: Hierarchy Referencing Time Synchronization (HRTS) and Individual-Based Time Request (ITR). HRTS constructs a spanning tree in the network with a master-node at the root. The master node synchronizes its children, and then the same synchronization procedure is repeated at subsequent levels in the hierarchy. Two independent radio



(a) (b) (c)

Figure 2.10: TSync operation: parent node broadcasts announcement message (a); randomly chosen node k replies (b); parent node broadcasts synchronization message (c).

channels are used for parent-child communication in the tree: control channel, which is common for all nodes, and clock channel, which is unique for each node. At the time t_1 against its timescale, the parent node broadcasts a timestamped announcement message on the control channel to its children (Fig. 2.10(a)). The announcement message also contains the number k, randomly chosen by the parent among the numbers of its children. After the transmission of the announcement message is finished, the parent switches to the dedicated clock channel of node k. Every child i records its time t_{2i} upon the reception of the announcement message. Then, only the chosen node k sends a time-stamped response message on the clock channel at the time t_{3k} against its timescale (Fig. 2.10(b)). This response message also contains t_{2k} — the time recorded by node k when it received the announcement message. The parent node receives the response message at the time t_4 according to its time, and estimates the clock offset d_k of node k relative to the parent time using the pair-wise synchronization procedure (see subsection 2.5.2). Then, the parent switches to the control channel and broadcasts one synchronization message containing the clock offset d_k and the time-stamp t_{2k} of the announcement message ar-

rival, received from node k (Fig. 2.10(c)). Every child node i receives the synchronization message and uses the value t_{2k} to compute the offset of its clock d_{ik} with respect to the clock of node k:

$$d_{ik} = t_{2i} - t_{2k} \tag{2.14}$$

Finally, every child i adjusts its clock to the parent time t_p:

$$t_p = t_i + d_{ik} + d_k \tag{2.15}$$

where t_i is the current time of a certain node i. ITR performs synchronization in the same way as HRTS but on demand of a particular node. The authors of TSync evaluated its accuracy by running it on a WSN composed of five GPS-enabled Nymph nodes with MANTIS operating system. The reported values of mean synchronization error are 21.2 μs for two nodes within one-hop and 29.5 μs for two nodes located tree hops away. TSync performs only three message exchange on each level of the constructed hierarchy, which saves power. Although the usage of dedicated radio channels involves higher implementation complexity, Tsync can be adapted to WSN that use only one channel. However, TSync relies on a hierarchical communication, which makes this service vulnerable to network topology changes.

2.6.3 Adaptive synchronization methods

Lightweight synchronization algorithms such as TS/MS, LTS and Tsync often include separate ad hoc synchronization methods, each of which is the most suitable in a certain scenario, depending on the required accuracy or number of active nodes in the network. However, these parameters can change rather quickly in wireless sensor networks. For example, a WSN that continuously monitors a chemico-bacteriological laboratory may activate additional sensor nodes when a dangerous pollutant is detected in the

air. The activated nodes can help to locate the source of pollution more precisely, since they will collect a larger amount of data and at a higher rate. Moreover, the network may need a better time synchronization accuracy in order to compute the velocity and direction of the pollutant diffusion. The described case shows that it is desirable to have time synchronization schemes that combine various specific algorithms and apply them selectively depending on the situation. Such flexible techniques should be able to keep track on variable conditions and synchronize the network nodes in the most energy-efficient way. Some adaptive synchronization schemes are emerging, which save energy by keeping the synchronization events as rare as possible while maintaining the desired synchronization accuracy [49, 6, 56, 21].

• An adaptive time synchronization protocol (authors do not designate it by any name), devised by PalChaudhuri et al. [49], uses the minimum possible number of synchronization messages to achieve a required accuracy with a certain probability. In this scheme, each of the dedicated network nodes sends time-stamped radio messages to a set of receivers. The receivers register the arrival time of those reference messages, and use linear regression to compute the offset and drift rate of their clocks with respect to the sender clock. Then they send the computed values back to the sender, which uses this information to find its relative clock drift rate and to broadcast it in a special packet to all receivers. After that, any pair of receivers can compute their relative clock drift rates and offsets. Assuming that the distribution of the synchronization errors is Gaussian, the authors prove the following relationship between the probability that a synchronization error $\boldsymbol{\epsilon}$ is less than the upper bound ϵ_{max} and the minimum number of synchronization messages n:

$$P\left(|\epsilon| \leq \epsilon_{max}\right) = 2erf\left(\frac{\sqrt{n}\epsilon_{max}}{\sigma}\right) \tag{2.16}$$

where σ is the standard deviation of the distribution. Also, the authors derive an expression for the maximum synchronization error γ_{max} that can develop in the network given the synchronization interval T_{sync}:

$$\gamma_{max} = \epsilon_{max} + (T_{sync} + \delta_{max})\nu_{max} \tag{2.17}$$

where δ_{max} is the maximum delay after the synchronization has been started, ν_{max} is the maximum clock drift rate among the network nodes. The described adaptive protocol is able to save power due to the probabilistic synchronization, which causes less overhead. However, the protocol may be not suitable for safety-critical applications.

• The energy efficient time synchronization protocol (ETSP), proposed by Shahzad at al. [56], saves energy by minimizing the number of synchronizations, depending on the number of network nodes requiring synchronization. This technique is based on the observation that receiver-receiver synchronization (which is used in RBS) requires less transmissions than sender-receiver synchronization (as in TPSN) when the number of network nodes is small. On the contrary, the sender-receiver approach is more energy-efficient in large and dense networks. In ETSP, a parent node synchronizes a group of its children using an RBS-like algorithm, if their number is less or equal to a certain threshold. If the number of children nodes exceeds the threshold, TPSN method is used.

• Rate Adaptive Time Synchronization (RATS) protocol [21], proposed by Ganeriwal et al., multiplicatively decreases and increases the synchronization interval (beaconing interval) within certain limits depending on how many times the synchronization error exceeds the user-defined bound. Also, this protocol chooses an optimal time during which some number of time-stamps are sent from node A to node B. RATS achieves one to two orders of magnitude reduction in the energy spent on radio transmission and is able to synchronize WSN nodes under changing environmental

conditions.

2.7 Summary

Time synchronization is essential for the operation of wireless sensor networks. Most of the well known approaches to time synchronization for WSNs are focused on achieving high synchronization accuracy for the whole network without addressing the power consumption problem. Other solutions improve power efficiency by reducing the synchronization complexity, or by enlarging the synchronization period. Recently, also some adaptive synchronization techniques for WSNs have been developed, whose common goal is to save energy by keeping the synchronization events as rare as possible on the basis of the actual synchronization status of the network.

Despite attempts to achieve a power-efficient time synchronization in WSN, the relationship of time uncertainty and power efficiency in typical WSN applications has not been fully addressed yet. Moreover, the measurements of power consumption of real WSN nodes running synchronization algorithms have not yet been done. These measurements could give a valuable quantitative data which, combined with a deeper insight into the problems of time synchronization, will show the advantage of each WSN synchronization method over the others.

Chapter 3

Time Uncertainty in WSN

3.1 Time Estimation Uncertainty

3.1.1 Modeling of Time Estimation Uncertainty

A WSN node measures time with its MCU timer/counter register, which counts cycles of the periodic signal supplied from the crystal resonator (see section 2.2). When some event happens, a special interrupt signal (e.g. a positive pulse on one of the microcontroller pins) can be generated to copy the timer value to the capture register. After the timer value is captured, it can be used by the program running on the MCU. Assume that the timer is reset and starts counting at real time t_0. The timer value N is captured after an event happens at real time t_1. Thus, if no overflow occurs, the timer counts N cycles during the real time interval $\Delta t = t_1 - t_0$. The WSN node can estimate that time interval as following:

$$\Delta t_{est} = \frac{N}{f_0} \tag{3.1}$$

where Δt_{est} is the time interval estimate and f_0 is the nominal oscillation frequency of the crystal. However, the actual frequency of the resonator differs from f_0, and this difference changes over time due to various environmental factors, crystal aging and noise. Also, the timer measures the

time in terms of an integer number of clock cycles (i.e. time measurement result is discrete), which limits the accuracy of the estimation (3.1). Finally, when an event happens, the timer value is not captured instantly, but after some delay. Taking into account all these factors, the result of 3.1 can be modeled as follows:

$$\Delta t_{est} = \frac{1}{f_0} \int_{t_0}^{t_1+\boldsymbol{\lambda}+\boldsymbol{\gamma}} f(t)dt \tag{3.2}$$

where:

- $f(t)$ is the instantaneous frequency of the resonator.
- $\boldsymbol{\lambda}$ is the random delay between the moment when the event happens and the moment when the corresponding interrupt signal is generated.
- $\boldsymbol{\gamma}$ is a random error due to the discrete representation of time by the timer value (discretization error).

The component $\int_{t_0}^{t_1+\boldsymbol{\lambda}+\boldsymbol{\gamma}} f(t)dt$ models the captured timer value, i.e. exactly N clock cycles occurs during the real time interval $[t_0, t_1 + \boldsymbol{\lambda} + \boldsymbol{\gamma}]$. Fig. 3.1 illustrates the meaning of $\boldsymbol{\lambda}$ and $\boldsymbol{\gamma}$. Obviously, $\boldsymbol{\lambda} \geq 0$, since the interrupt is generated only after the event has happened. The sign and range of $\boldsymbol{\gamma}$ depends on whether the time capture is done asynchronously to the clock signal or not. Asynchronous capture is done immediately as soon as an interrupt signal arrives. In this case, $\boldsymbol{\gamma} \leq 0$, because the integer number of clock cycles counted by the timer is always equal or less than the real number of cycles at the moment when the interrupt happens, and $\int_{t_0}^{t_1+\boldsymbol{\lambda}+\boldsymbol{\gamma}} f(t)dt = \left\lfloor \int_{t_0}^{t_1+\boldsymbol{\lambda}} f(t)dt \right\rfloor$, where $\lfloor\rfloor$ represents the floor function. Therefore, $\boldsymbol{\gamma} \in [-d_{cycle}, 0]$, where d_{cycle} is the duration of the current clock cycle. Asynchronous capture may lead to a race condition. Alternatively, the capture can be synchronized with the next clock edge occurring in the middle

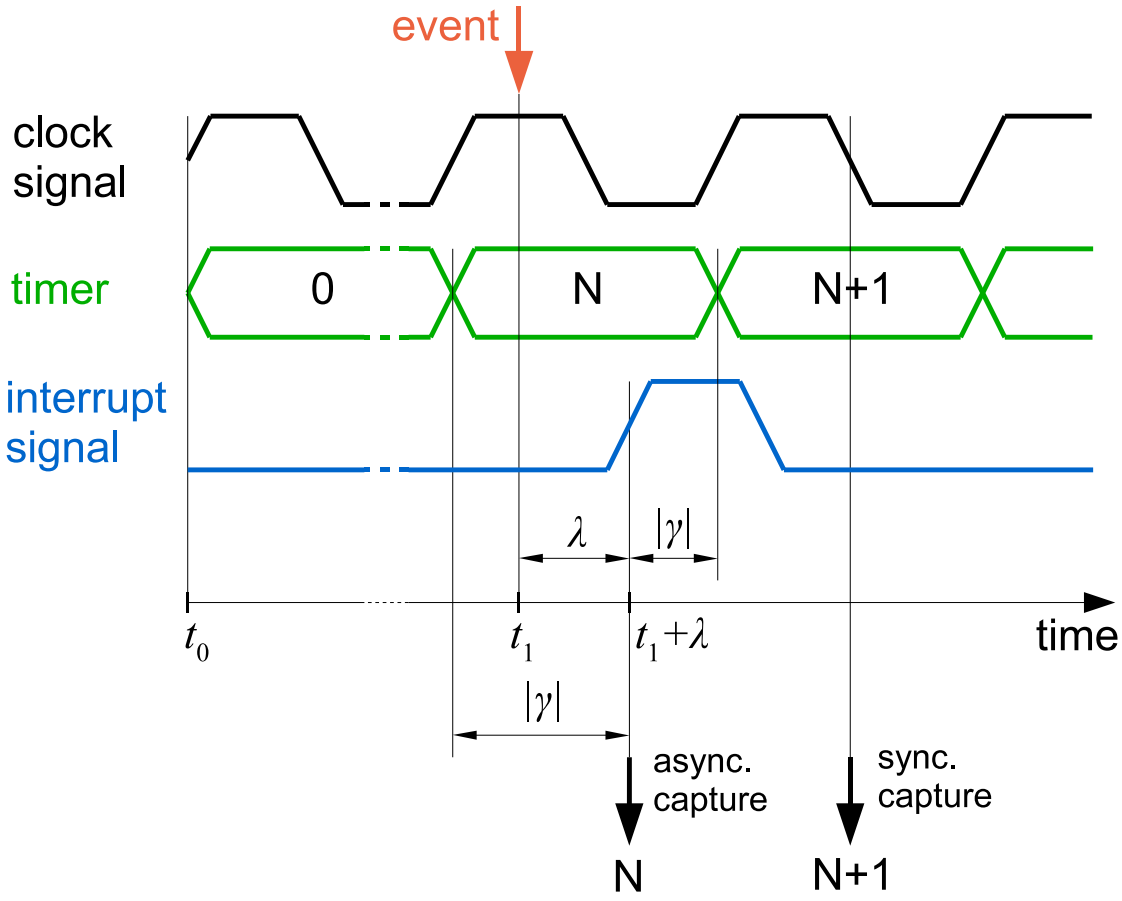


Figure 3.1: Time capture.

of a clock cycle (i.e. when the timer value does not change). In this case, $\gamma \leq 0$ if the interrupt signal arrives in the first half of the current cycle. Indeed, the integer number of cycles captured on the middle clock edge is still the same as in the first half of the cycle and is less than the corresponding real number of cycles: $\int_{t_0}^{t_1+\lambda+\gamma} f(t)dt = \left\lfloor \int_{t_0}^{t_1+\lambda} f(t)dt \right\rfloor$. When the interrupt happens in the second half of a clock cycle, the synchronous capture is performed in the middle of the next cycle. As a consequence, the captured timer value is more than the real number of cycles at the moment when the interrupt happens. In this case $\gamma \geq 0$ and $\int_{t_0}^{t_1+\lambda+\gamma} f(t)dt = \left\lceil \int_{t_0}^{t_1+\lambda} f(t)dt \right\rceil$,

where $\lceil\rceil$ represents the ceiling function. Thus $\boldsymbol{\gamma} \in [-\frac{d_{cycle}}{2}, \frac{d_{cycle}}{2}]$ when the capture is synchronous.

The uncertainty δ of the time interval estimation (3.1) is:

$$\delta = \Delta t_{est} - \Delta t = \frac{1}{f_0} \int_{t_0}^{t_1+\boldsymbol{\lambda}+\boldsymbol{\gamma}} f(t)dt - \Delta t \tag{3.3}$$

The instantaneous resonator frequency $f(t)$ can be modeled as:

$$f(t) = f_0 + \Delta f_{man} + \Delta f_{oper}(t) + \Delta f_{age}(t) + \Delta f_{fluc}(t) \tag{3.4}$$

where:

- Δf_{man} is a constant frequency deviation due to the imprecision in the angle at which the crystal is cut during its manufacturing process [53];
- $\Delta f_{oper}(t)$ is a change in frequency caused by operating conditions such as temperature, input voltage, humidity and vibration [31, 70];
- $\Delta f_{age}(t)$ is a frequency deviation related to the crystal aging [31, 70];
- $\Delta f_{fluc}(t)$ is a frequency deviation due to the random short-term frequency fluctuations such as white phase modulation (WPM) noise, flicker phase modulation (FPM) noise, white frequency modulation (WFM) noise, flicker frequency modulation (FFM) noise, and random walk frequency modulation (RWFM) noise [8, 12, 70].

Vendors of quartz crystal resonators generally indicate the frequency stability specification, which combines the effect of all factors and is given as a single number representing the worst-case relative frequency drift in the allowed operating conditions [31]. The frequency stability and relative frequency drift are expressed in terms of parts per million, e.g. $\pm$100 ppm. The relative frequency drift is also called the normalized frequency deviation, fractional frequency change or fractional frequency error or just

frequency drift [28, 2, 68]. All these terms stand for a difference between the actual and nominal frequencies divided by the nominal frequency. In order to emphasize the relationship between the instantaneous frequency and relative frequency deviation due various factors, the expression (3.4) can be written as

$$f(t) = f_0(1 + \frac{\Delta f_{man}}{f_0} + \frac{\Delta f_{oper}(t)}{f_0} + \frac{\Delta f_{age}(t)}{f_0} + \frac{\Delta f_{fluc}(t)}{f_0}) \tag{3.5}$$

or as

$$f(t) = f_0(1 + y_{man} + y_{oper}(t) + y_{age}(t) + y_{fluc}(t)) \tag{3.6}$$

where $y_{man} = \frac{\Delta f_{man}}{f_0}$ is the normalized frequency deviation due to the crystal manufacturing process and $y_{oper}(t) = \frac{\Delta f_{oper}(t)}{f_0}$, $y_{age}(t) = \frac{\Delta f_{age}(t)}{f_0}$, $y_{fluc}(t) = \frac{\Delta f_{fluc}(t)}{f_0}$ are instantaneous normalized frequency deviations due to operating conditions, crystal aging and random short-term frequency fluctuations respectively.

The frequency stability can be approximately considered as a sum of maximum values of y_{man}, $y_{oper}(t)$, $y_{age}(t)$ and $y_{fluc}(t)$. However, commercial manufacturers often assume a crystal operation life of only a few years, thus ignoring aging during a longer term [31]. Military specifications generally indicate the frequency stability related to the operation in an extended temperature range and the impact of factors such as humidity, shock and vibrations.

The constant frequency drift y_{man} of commercial quartz crystals can be in the order of 20-50 ppm [53]. The major part of $y_{oper}(t)$ is the drift in the order of 10-20 ppm caused by the temperature. The temperature-dependent frequency drift y_{temp} is not linear with temperature. For AT-cut crystals y_{temp} can be approximated as

$$y_{temp}(T) \approx K_1(T - T_0)^3 + K_2(T - T_0) + K_3 \tag{3.7}$$

where T is temperature and K_1, K_2, K_3, T_0 are unique to each resonator [53, 9]. Each crystal cut angle has a particular curve of the fre-

quency deviation versus temperature and the shape of that curve does not change with time, i.e. y_{temp} is constant if the temperature is stable [31]. The frequency drift caused by the crystal aging $y_{age}(t)$ changes with time, and new crystals age faster than old ones (e.g. 5 ppm in the first year of operation, and then 3 ppm [31]). In stable environmental conditions, aging typically exhibits logarithmic dependence on time [68, 71, 70, 47, 9, 48]:

$$y_{age}(t) = A_1 \log(1 + A_2 t) \tag{3.8}$$

where coefficients A_1 and A_2 are usually determined from the least square fit using the measurement results. However, after an initial stabilization period (usually 30 days) the aging is often considered as approximately linear function of time and is specified in ppm/year [29, 71]. The effect of random frequency fluctuations is generally estimated by means of the Allan variance over a certain time interval [8, 70]. The order of magnitude of the corresponding standard deviation estimated over 1 s is some nanoseconds for low cost XO [70, 2].

The rough estimate of the drift magnitudes given above allows to say that during a medium-term of operation the frequency deviation is mainly caused by the crystal cut imprecision and deviation of temperature from the standard ambient temperature (+25°C ±3°C). In this case, the resonator frequency approximates to:

$$f = f_0(1 + y_s) \tag{3.9}$$

where y_s is a frequency drift due to the combined effect of manufacturing imprecision and temperature offset, i.e. $y_s = y_{man} + y_{temp}(t)$. In stable environmental conditions y_{temp} is constant and therefore the relative frequency drift y_s can be considered as constant. Using (3.9) in (3.3) and neglecting the impact of the errors due to discretization and delayed time capture

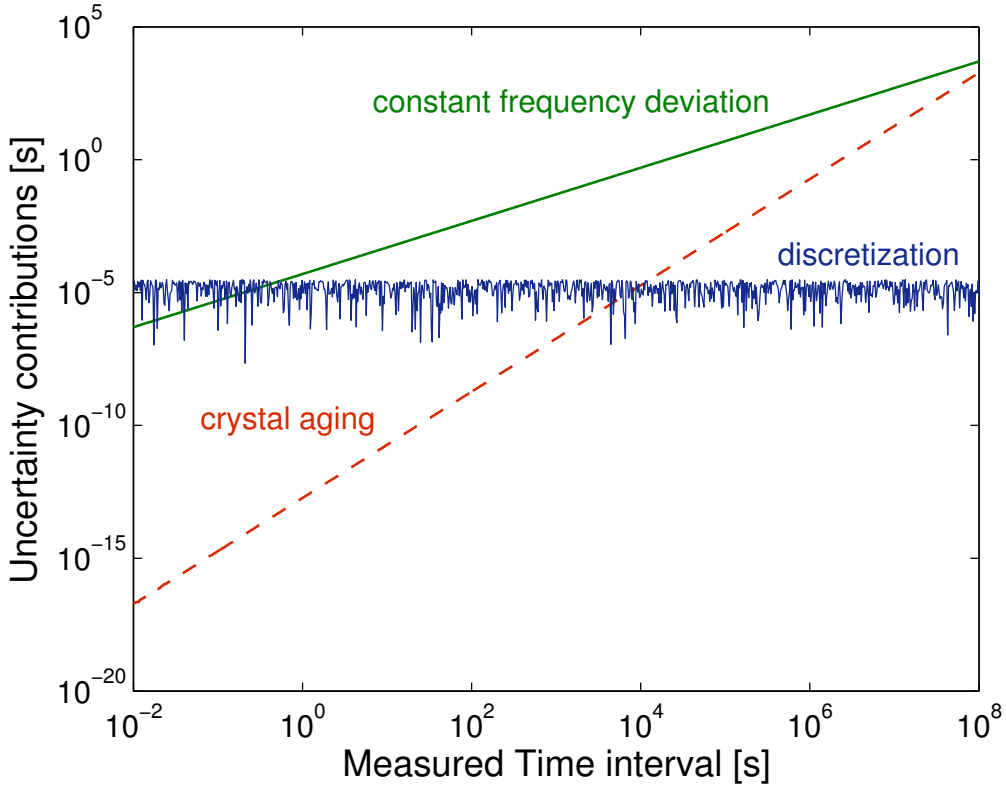


Figure 3.2: Individual contributions of time measurement uncertainty due to the constant deviation of the oscillator frequency (solid line), crystal aging (dashed line) and time discretization (noisy pattern). All terms are plotted as a function of the measured time intervals on a logarithmic scale.

gives an approximate expression for the time estimation uncertainty:

$$\delta \approx \frac{1}{f_0} \int_{t_0}^{t_1} f_0(1 + y_s)dt - \Delta t \tag{3.10}$$

Taking into account that y_s is constant and $\Delta t = t_1 - t_0$, (3.10) can be simplified as:

$$\delta \approx y_s \Delta t \tag{3.11}$$

i.e. the difference between an interval estimate and the corresponding real time interval grows approximately linearly with the interval duration. In order to highlight this point, let us assume that y_s=50 ppm, the frequency

drift change due to aging is 2.5 ppm/year and f_0=32.768 kHz. These values are in accordance with the features of the XOs commonly used on commercial WSN nodes, such as the TelosB/Tmote Sky modules. In Fig. 3.2 individual uncertainty contributions due to the discretization (noisy pattern), constant frequency offset (solid line) and crystal aging (dashed line) are shown on a logarithmic scale as a function of increasingly large measured time intervals. Notice that for very short time intervals (i.e. below 1 s) the discretization is the main source of uncertainty, whereas the aging effect becomes significant only after about 3 years. In all the intermediate cases, the measurement uncertainty is dominated by the frequency offset. Since the typical operational lifetime of a WSN node is in the order of some weeks, and the periodic synchronization operations are not repeated very frequently in order to save power, the expression (3.11) can be used to model the overall time estimation uncertainty in most of the situations of practical interest.

3.1.2 Clock Drifts Measurements

In order to evaluate the effect of resonator frequency deviation on time estimations of typical WSN nodes, 60 time intervals were measured during 1 hour using the timers of 10 TelosB and Tmote Sky nodes running TinyOS. The time measurements were performewd by 32-bit timers clocked at 32.768 kHz. The common reference signal used to compare the time intervals measured by the nodes was a square wave produced by a function generator Agilent 33220A. In fact, according to the instrument specifications [5], the RMS jitter of the reference waveform in pulse mode is about 100 ns, namely some orders of magnitude smaller than the intervals to be measured. The pulse period was set to 60 s. A special ad-hoc cable with a BNC connector at one end and split into 10 twin wires at the other end was used to deliver the reference pulse signal to MCU pins of the nodes

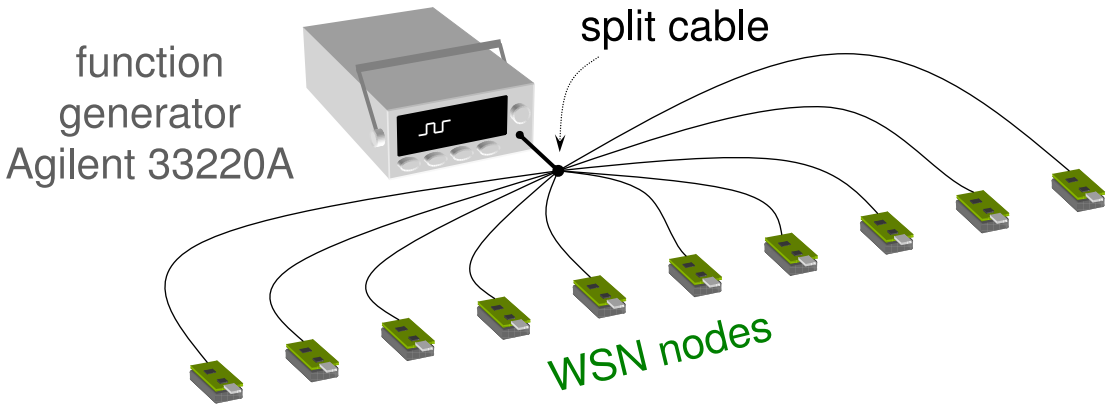


Figure 3.3: Experimental setup used to measure the clock drifts of WSN nodes with respect to a stable reference signal produced by a function generator Agilent 33220A.

(Fig. 3.3). The nodes were programmed to record their timer values at each rising edge of the reference signal. The ambient temperature was stable during the whole experiment (around +22°C) performed in a laboratory. In Fig. 3.4 the differences between the values measured by the MCU timers of the nodes are plotted against the reference values provided by the function generator. Notice that these differences exhibit essentially a linear trend with a slope that is constant for a given node. This means that during the experiment the total frequency deviation was dominated by the constant frequency deviation y_s of the employed resonator, while all the other contributions were negligible. Therefore, the time estimation uncertainty is indeed a linear function of the measured time interval under stable environmental conditions, as expected, and the approximate expression (3.11) holds in most of practical cases.

Wireless sensor nodes are often placed outdoors, and operate under varying temperature. In order to evaluate the effect of temperature on the stability of resonator frequencies of WSN nodes, another experiment was done with four TelosB nodes, two of which were exposed to directed flow of heated air. As before, a function generator Agilent 33220A produced

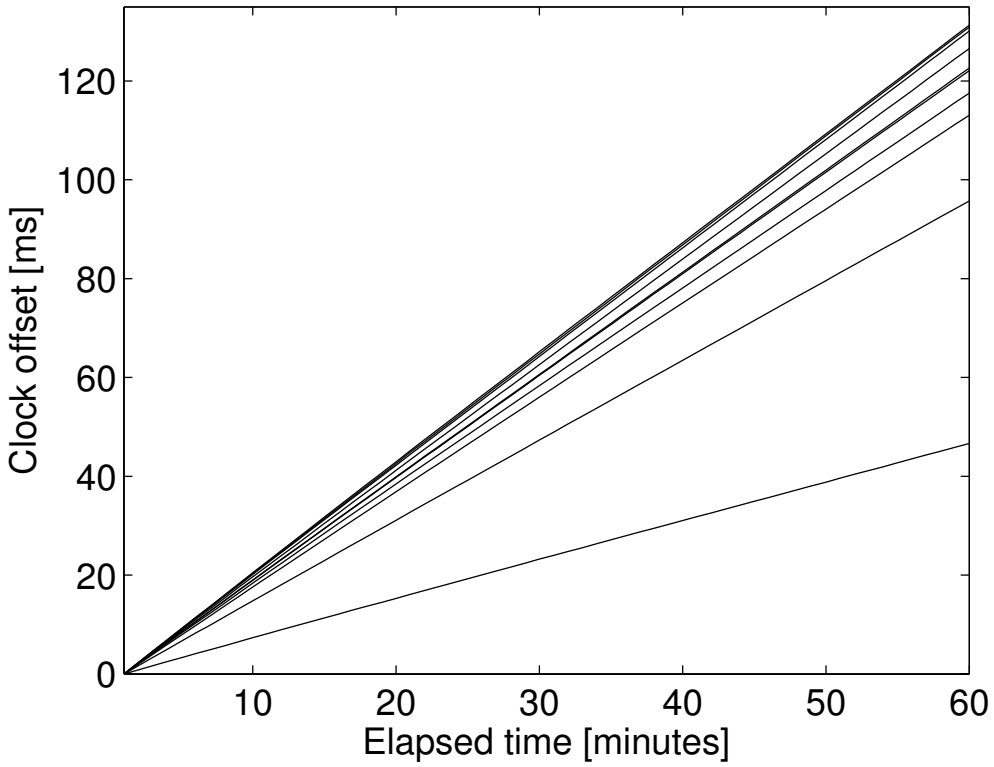


Figure 3.4: Measured offsets between the clocks of 10 TelosB and Tmote Sky nodes and the time values provided by a function generator Agilent 33220A. The nodes measure time once per minute by reading their 32-bit timers clocked at 32.768 kHz.

reference pulses with a period of 60 s, and the nodes recorded their timer values on each rising edge of the pulse signal. In addition, the nodes measured the air temperature with Sensirion SHT11 sensors [55]. First, all nodes operated under stable temperature (around +22°C). The source of heated air was activated and directed at the nodes 1 and 2 after 30 minutes from the beginning of the experiment. In the 60th minute of experiment the source of heated air was switched off and all the nodes continued to operate for 30 minutes. In Fig. 3.5 the offsets between the timer values of the nodes and the reference values provided by the function generator are shown. Notice that clock drift rates of the heated nodes appreciably change during the time of heating (interval boundaries are marked with

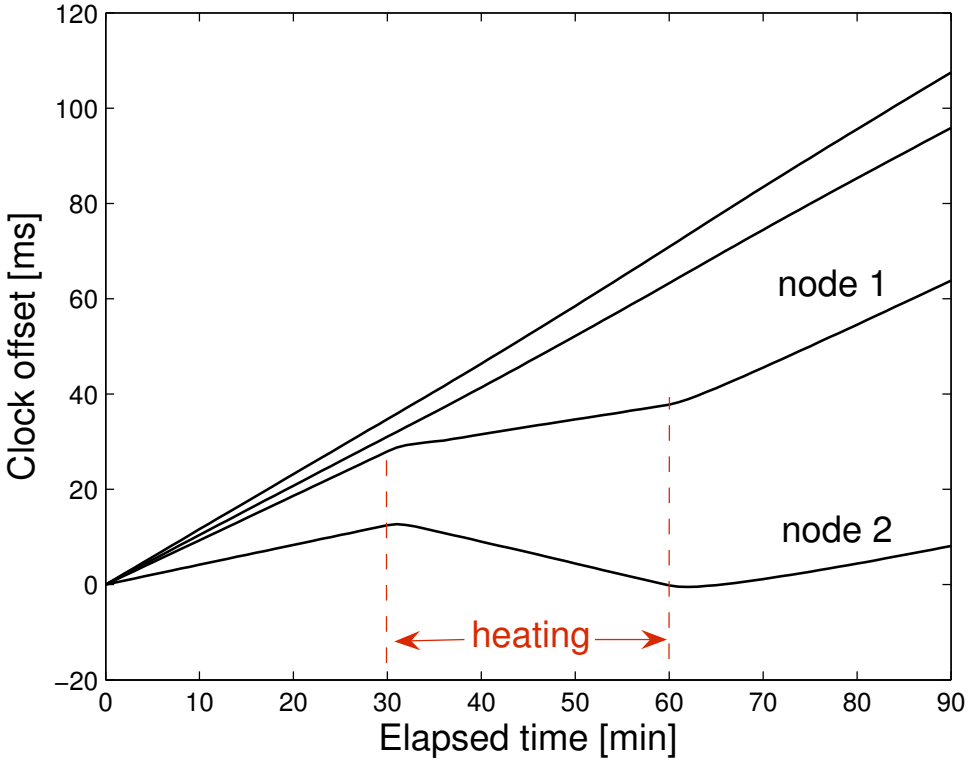
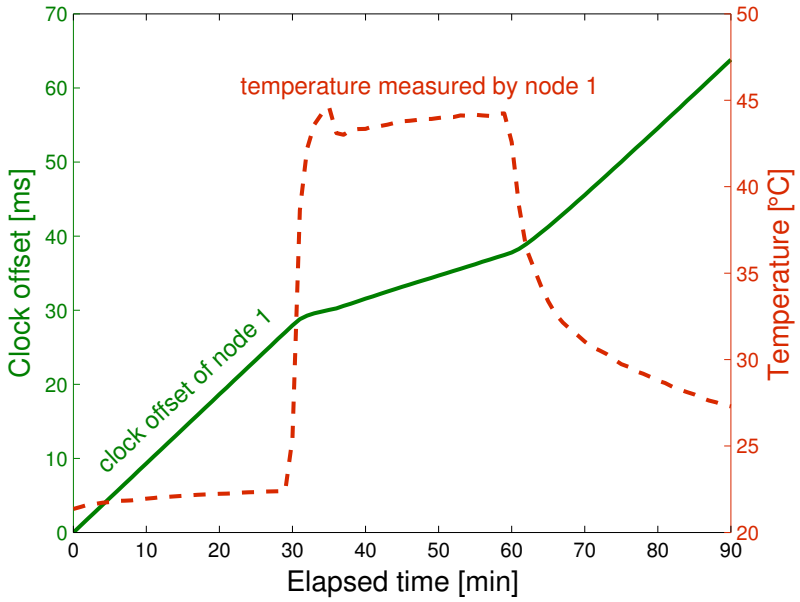


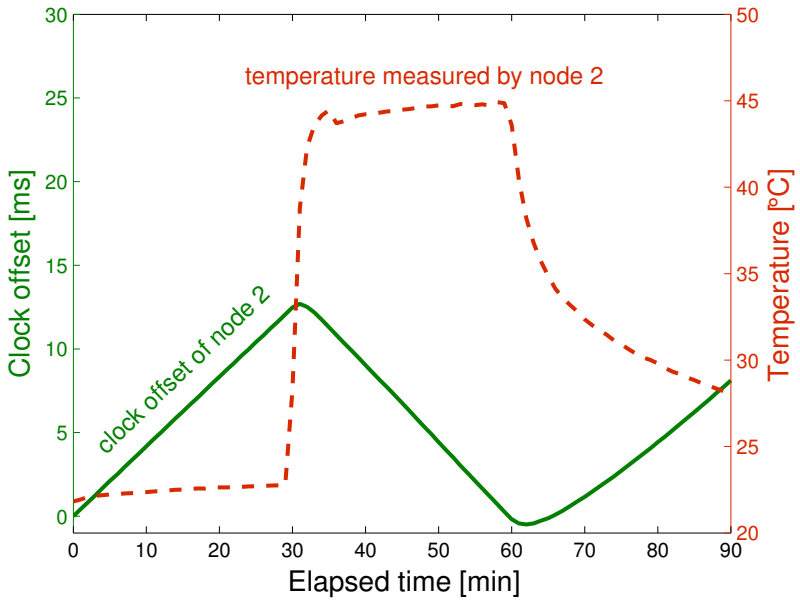
Figure 3.5: Measured offsets between the clocks of 4 TelosB modules and time values provided by a function generator Agilent 33220A. The nodes measure time once per minute by reading their 32-bit timers clocked at 32.768 kHz.

two dashed lines), while the clock drift rate of the other two nodes remains constant. Clearly, the resonator frequency of node 2 is more sensitive to the temperature, and the clock drift rate of node 2 even changes its sign during heating.

In Fig. 3.6(a) and Fig. 3.6(b) the clock offsets of the heated nodes with respect to reference time values and the air temperature measured by the nodes are plotted against the time values provided by the function generator. Although clock offsets exhibit a linear trend with time when the temperature is approximately constant (around +22°C during the first 30 minutes of experiment, and around +45°C during the following 30 minutes), they change non-linearly under varying temperature. This is obvi-



(a)



(b)

Figure 3.6: Clock offsets of two heated TelosB nodes 1 (a) and 2 (b) and the air temperature measured by the nodes plotted against the time values provided by a function generator Agilent 33220A.

ous, especially in Fig. 3.6(b) during the last 30 minutes of experiment, when node 2 gradually cools down. The results of this experiment emphasize the impact of temperature on the accuracy of time estimation performed by WSN nodes.

3.2 Time Transfer Uncertainty

When transferring a certain time value between two WSN nodes, the time of the sender will be actually available at the receiver end only after some delay. Such a latency depends on the amount of operations occurring between the moment when the sender timer is read and the instant in which the receiver time is updated. As known, in case of an end-to-end, single-hop transmission the total latency includes the following terms [34]:

- *send time*, namely the nondeterministic time spent by a transmitter for building the packet at the application layer and for transferring the packet to the MAC layer. This time depends on the interaction between the MCU and the RF module, on the processor load as well as on the OS scheduling algorithms;
- *access time*, i.e. the random, MAC-specific time spent to access the wireless channel. Such a delay depends on the network traffic as well as on the features of the communication protocol;
- *transmission time*, i.e. the time to turn an equivalent bit of information into modulated electromagnetic waves just before transmission;
- *propagation time* taken by one bit to cross the wireless channel from the sender to the receiver;
- *reception time*, i.e. is the time to transform a received electromagnetic wave into one equivalent bit of information;

- *receive time* including both the time to transfer the packet from the MAC layer to the application layer and the time to process the packet, e.g. to extract the received time value.

If a carrier sense multiple access with collision avoidance (CSMA/CA) protocol is used, the overall end-to-end latency δ_C can be modeled by a random variable resulting from the addition of three major terms, i.e.:

$$\delta_C = \delta_S + \delta_L + \delta_R \tag{3.12}$$

where:

- δ_S and δ_R coincide with the send time and the receive time, respectively;
- δ_L is the link and physical layer latency which includes one or multiple values of access, transmission, propagation and reception time depending on the number of retransmission attempts in case of busy channel or packet collision.

In order to have a clearer insight about the features of the three individual terms mentioned above, several δ_S, δ_L and δ_R values were measured using an automatic experimental setup based on an oscilloscope controlled by a PC. In particular, after identifying (from the documentation of the TelosB module [45] and radio transceiver CC2420 [63]) which pins of the radio chip can be used to generate pulses at the beginning and at the end of δ_S, δ_L and δ_R intervals during a packet transmission, we used the oscilloscope to measure the time differences between the pulse edges delimiting the various latency contributions. Notice that this approach leads to very accurate results, because software-related delays almost do not affect the measurement process.

In the following subsections, the results of the performed experiments as well as the models deduced from the analysis of the collected data are described.

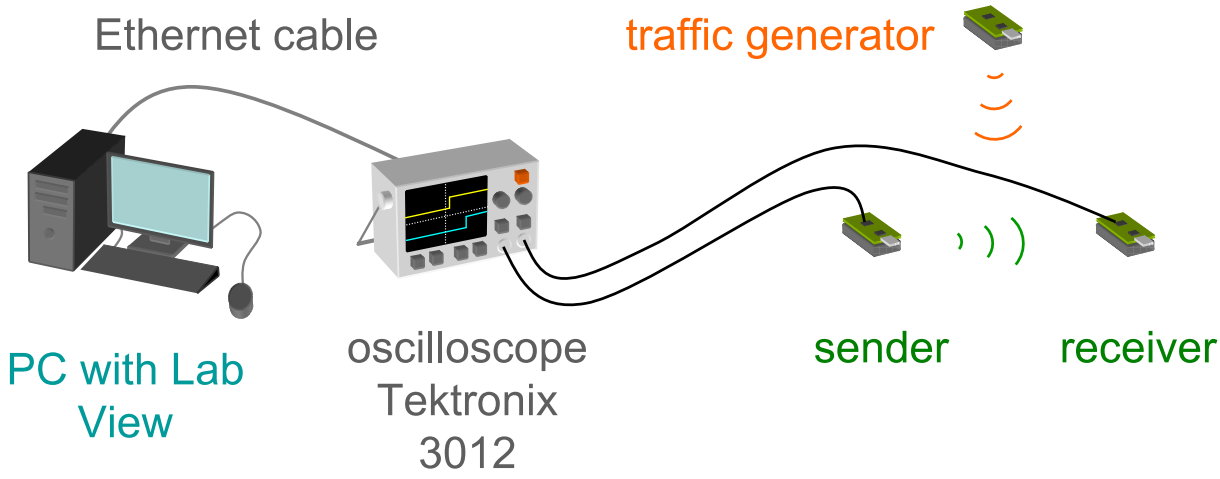


Figure 3.7: Experimental setup to measure communication delays.

3.2.1 Experimental Setup and Measurement Procedure

The basic components of the proposed experimental setup are shown in Fig. 3.7 and include not only one sender and receiver, but also a random traffic generator, namely a node that is explicitly used to emulate a known amount of traffic within the network, as it will be explained in the next subsections. All measurements were performed on TelosB nodes running TinyOS [26]. Each TelosB node is equipped with an IEEE 802.15.4 compliant 2.4 GHz radio transceiver CC2420 [63], which is controlled by a microcontroller MSP430F1611 [64] via Serial Peripheral Interface (SPI).

The sender node was programmed to send periodically a radio packet to the receiver node. Several test commands were inserted in the transmitter code to generate a special pulse on one MCU pin (send-pulse). The rising edge of the generated pulse appears on this pin (as a result of setting the corresponding bit of the port register), just before running the sending commands. After those commands are executed, the falling edge of the pulse is produced by clearing the bit of the port register. The receiver was programmed to generate a similar pulse (receive-pulse) on one of its MCU pins by suitably adding several commands to the receiver program code.

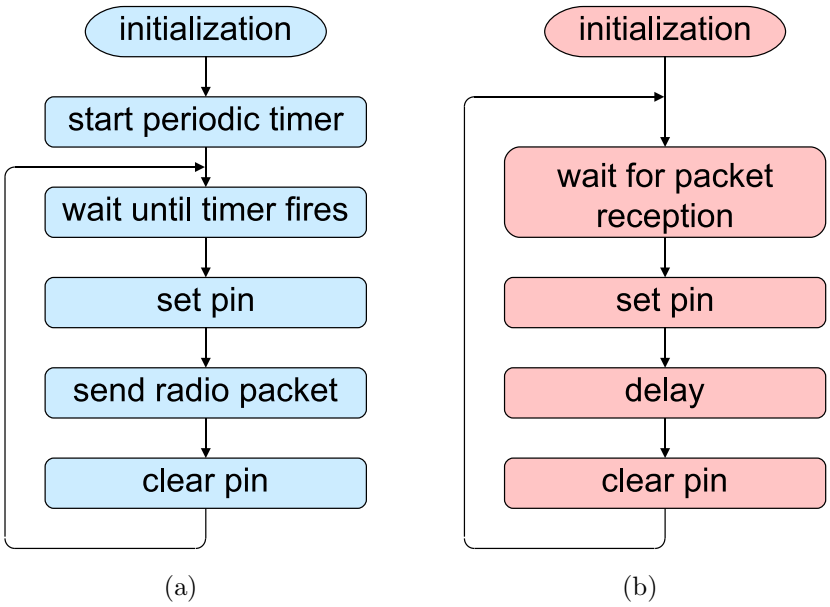


Figure 3.8: Flowcharts of the applications running on the sender (a) and receiver (b).

The flowcharts of the applications running on the sender and receiver nodes are shown in Figures 3.8(a) and 3.8(b).

Also, some low-level modules of TinyOS running on the sender node were modified to generate additional test pulse when the first channel access attempt is just about to happen (access-pulse). The send time was assumed to be equal to the time interval between send-pulse and access-pulse. Besides, the SFD signal, generated by CC2420 on the corresponding pin, was used to measure the receive time. In fact, the positive edge of SFD signal appears when the start-of-frame-delimiter byte is transmitted or received, the negative edge corresponds to the end of transmission or reception of the last byte in the packet [4, 63]. We considered the time interval between the negative edge of the receiver SFD signal and the positive edge of receive-pulse as equal to receive time. The time interval between the positive edge of access-pulse and negative edge of the receiver SFD signal was assumed to be equal to the link and physical layer latency. All test pulses and their relations to different stages of the packet transmission and

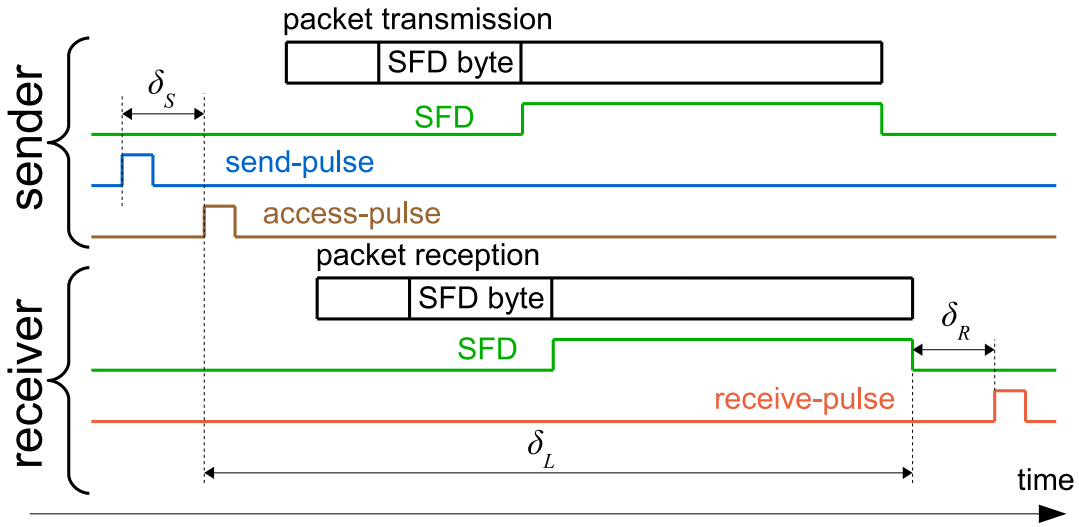


Figure 3.9: Test signals used to measure different components of the communication delay. Signal edges correspond to the boundaries between different stages of radio packet transmission and reception.

reception are shown in Fig. 3.9. The long segmented rectangle on the top of Fig. 3.9 represents the transmission of the radio packet by the sender, the bottom rectangle (identical to the upper one, but shifted to the right) represents the reception of the same radio packet by the receiver.

In the experiments the pins of either MCU or radio chip of the transmitting and receiving nodes were connected to the channels of a digital storage oscilloscope (DSO) TDS 3012 [62]. A LabVIEW application controlled the oscilloscope through an Ethernet interface. The application was specifically designed for measuring the time interval between the voltage pulses acquired on the two channels, with a pulse edge on the first channel set as a triggering event. Eventually, multiple measured time values were saved in a log file in order to be analyzed later.

A screen-shot showing two pulses measured during the transmission of a radio packet with a payload of 40 bytes is shown in Fig. 3.10. In this example, the top pulse is a send-pulse, while the bottom one is a receive-pulse. Here, the measured time difference between the rising edges of the

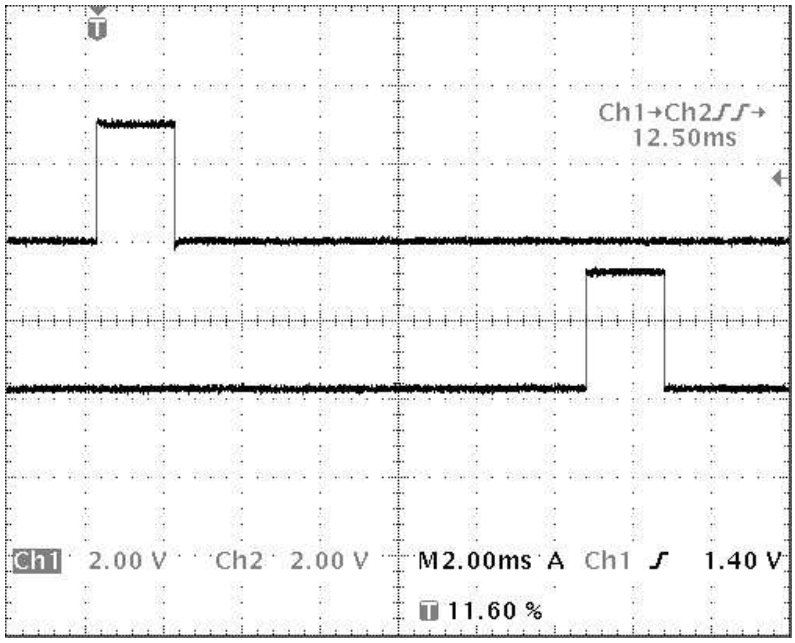


Figure 3.10: Two test pulses measured during the transmission of a radio packet with a payload of 40 bytes.

pulses is 12.50 ms and corresponds to the total communication delay. Consider that the instrumental uncertainty associated to such a measurement result is negligible, because the worst-case contributions affecting the time measurements using the chosen oscilloscope are several order of magnitude smaller than the interval under test. In fact, the delay time accuracy of the oscilloscope is ±200 ppm over any time interval longer than 1 ms [62]. Also, according to the MSP430F1611 manual [65], moving a constant to a certain destination address requires 5 CPU cycles. Therefore, since the microcontroller operates at the frequency of 8 MHz, the commands setting or clearing a port pin should take less than 1 μs.

3.2.2 Send and Receive Time

In the first set of experiments, 10 sequences of 100 packets having the same payload size (ranging from 10 bytes up to 100 bytes) were transmitted

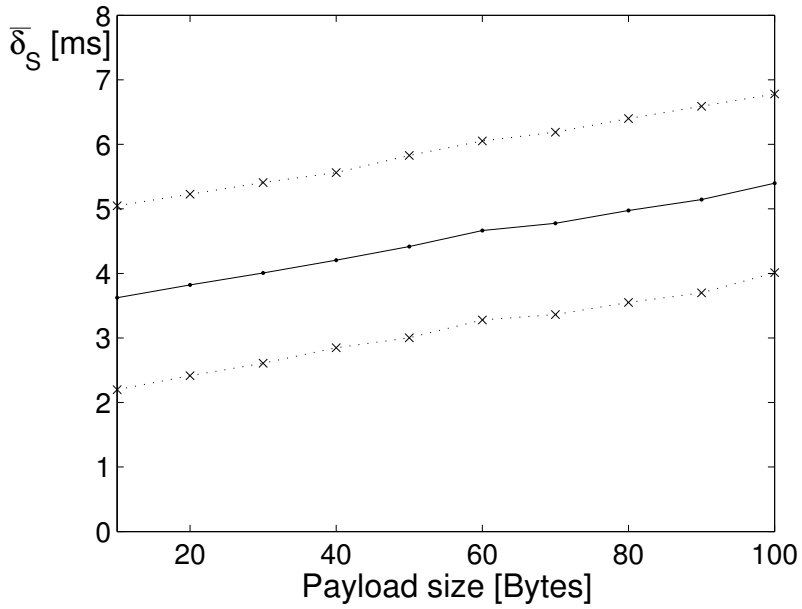


Figure 3.11: Average send time values (solid central line) and $\pm\sigma$ limits (dashed lines) as a function of the payload size. The mean and standard deviation values used to plot the picture are estimated using sets of 100 measurement results for different payload sizes.

between two nodes. The average values (solid central line) and the $\pm\sigma$ limits (dashed lines) of the estimated send time are shown in Fig. 3.11 as a function of the payload size. Notice that delay is random but it grows linearly with packet size. This is reasonable because if the MCU transfers the packet to the radio module through a serial connection (SPI), the send time depends on the packet length. As a consequence, the send time can be modeled as

$$\delta_S = \bar{t}_S(D + H) + \boldsymbol{\epsilon_S} \tag{3.13}$$

where:

- D is the payload size (e.g. expressed in bytes);
- H is the fixed number of additional overhead bytes (i.e. H=10 bytes in TinyOS 2.0);
- $\bar{t}_S$ (which is about 19.4 μs according to our experiments) represents

the average time to transfer 1 byte over the serial connection;

- $\boldsymbol{\epsilon_S}$ is a random variable including all the other possible random contributions such as the latency of the interrupt service routine reading the timer or the time spent by the MCU in running other tasks.

In general, it is not possible to provide a unique stochastic description of $\boldsymbol{\epsilon_S}$ due to the excessive number of factors affecting this term. However, according to the performed experiments, $\boldsymbol{\epsilon_S}$ exhibits approximately a uniform distribution with mean value equal to 3.4 ms and standard deviation of about 1.4 ms. This contribution is most probably dominated by the random delay introduced on purpose by TinyOS before attempting any channel access. The features of the receive time are quite similar to those of the send time. In fact, also in this case the delay grows linearly with payload size. Therefore, the receive time can be modeled as

$$\delta_R = \bar{t}_R(D + H) + \boldsymbol{\epsilon_R} \tag{3.14}$$

where $\bar{t}_R$ is the average time to transfer 1 byte of data over the serial connection between the radio module and the MCU and $\boldsymbol{\epsilon_R}$ is the random delay to process the incoming packet. In order to validate (3.14), in the second set of experiments the receive time values were measured by transmitting 10 sequences of 100 packets with the same payload size (i.e. ranging from 10 bytes up to 100 bytes) between two nodes. The corresponding average delays (solid central line) and the $\pm\sigma$ limits (dashed lines) are shown in Fig. 3.12 as a function of the payload size. Observe that the dashed lines in this case can be hardly distinguished from the line representing the mean values. This means that the receive times are mostly deterministic. In fact, the mean of $\boldsymbol{\epsilon_R}$ is about 1.9 ms, whereas its standard deviation is in the order of few μs, i.e. negligible. Such a deterministic behavior is due to the fact that the time to service the interrupt generated by the radio module is

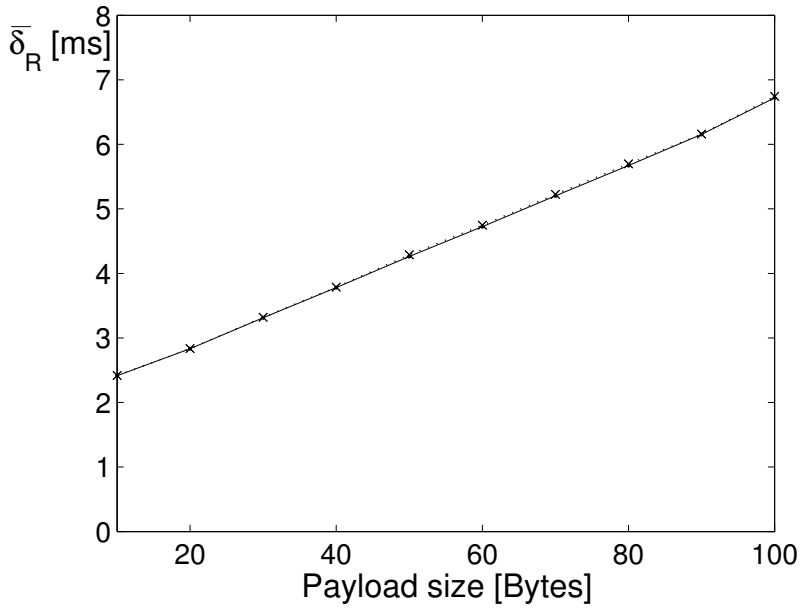


Figure 3.12: Average receive time values (central solid line) and $\pm\sigma$ limits (dashed lines) as a function of the payload size. The mean and standard deviation values used to plot the picture are estimated using sets of 100 measurement results for different payload sizes.

approximately constant because no other tasks run on the receiving node. Notice that the value of $\overline{t}_R$ is larger than $\overline{t}_S$ (i.e. 47.6 μs). This is probably due to some additional handshaking delays when the data are transferred from the FIFO buffer of the radio module to the MCU.

3.2.3 Link and Physical Layer Delay

The study of the delay associated to the link and physical layer is more involved than the analysis of the send and receive times, because δ_L depends not only on the payload size, but also on the traffic throughout the network. The qualitative description of the pure packet transfer delay after accessing the channel is shown in Fig. 3.13 and can be expressed as follows:

$$\delta_{TR} = \delta_D + [8(D+P) - 1]t_b + \delta_{TX} + \delta_P + \delta_{RX} \tag{3.15}$$

where:

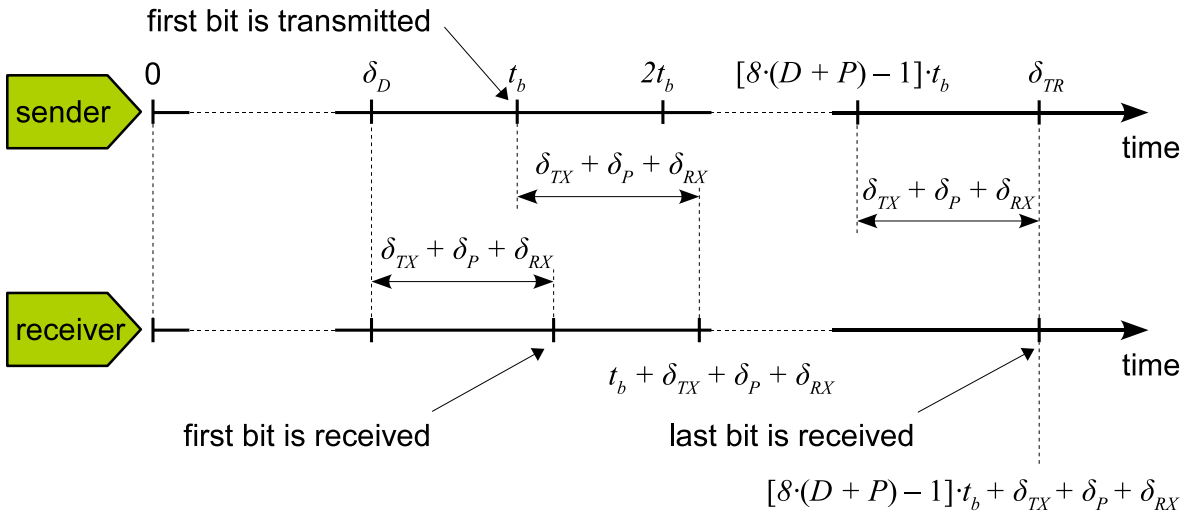


Figure 3.13: Qualitative description of the packet transfer delay δ_{TR} in case of successful channel access with no collisions.

- δ_D is the turnaround time necessary to switch between TX and RX radio states (this is approximately 192 μs for the radio modules used on TelosB nodes [63]);

- D is the payload size in bytes;

- P is the total number of additional bytes added at MAC and PHY layers just before starting the transmission (e.g. P=H+7=17 bytes in TinyOS 2.0);

- t_b is the reciprocal of the nominal bit rate (i.e. 4.0 μs if the bit rate is 250 kbit/s);

- δ_{TX}, δ_P and δ_{RX} represent the transmission time, propagation time and reception time, respectively, as defined at the beginning of Section 3.2. According to the specifications of the radio module used on TelosB nodes, δ_{RX} is about 3 μs [63]. Given that δ_{TX} must be strictly smaller than T_b, it is reasonable to assume that δ_{TX} is approximately the same as δ_{RX}. Finally, δ_P in wireless sensor network is at most in

the order of 1 μs.

Observe that the packet transfer time δ_{TR} is mostly deterministic. However, if WSN nodes rely on a CSMA/CA protocol, the total communication latency δ_L becomes random because it includes also the access time to the channel, the latencies due to the possibility of finding the channel busy and the retransmission delays following possible packet collisions. By default the MAC sublayer of TinyOS relies on an unslotted version of CSMA/CA medium access mechanism, similar to that described in the standard IEEE 802.15.4 [4]. The unslotted CSMA/CA access mechanism can be shortly described as follows. As soon as a packet is transferred from the application layer to the MAC layer, the transmitter at first waits for a random amount of basic time units called *backoff periods*. Then, the MAC layer requests the physical layer to perform the Clear Channel Assessment (CCA). If the channel is idle, the MAC layer begins to transfer a packet, otherwise the random waiting delay has to be increased and the CCA operations are repeated. Finally, if the channel is idle, but a collision occurs (i.e. no acknowledge message is received by the transmitting node within a certain timeout), all the procedure mentioned above should restart from the beginning.

A possible model describing the effect of both busy channel events and packet collisions relies on the general theoretical analysis described in [33] and [76]. If R is the random variable modeling the total number of packet transmission attempts anytime the channel is sensed idle, and NB_r for r=1,...,R is the number of times in which the channel is sensed busy before performing the rth attempt, we have that the total communication latency results from:

$$\delta_L = R\delta_{TR} + (R-1)\delta_{ACK} + \sum_{r=1}^{R}\sum_{i=0}^{NB_r}(X_i B + C) \tag{3.16}$$

where:

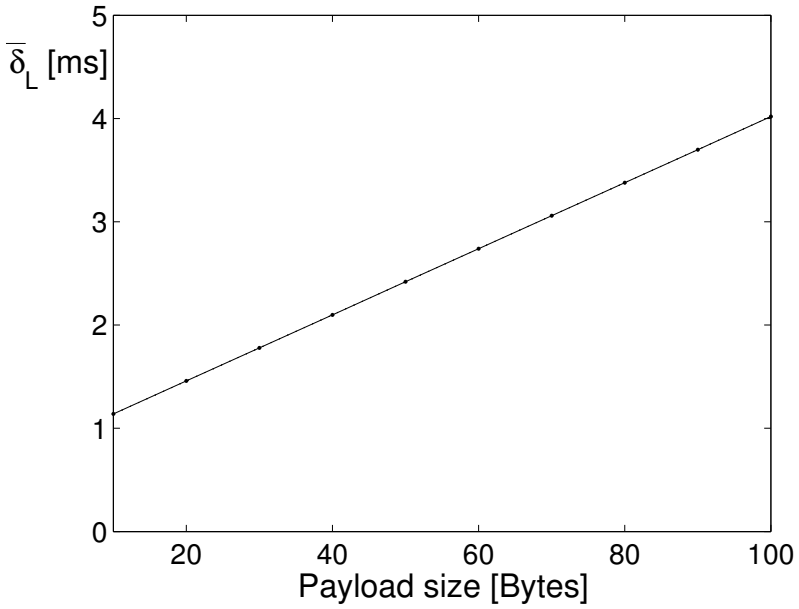


Figure 3.14: Average link and physical layer latencies as a function of the payload size in negligible traffic conditions. The mean values are estimated using sets of 100 measurement results for different payload sizes. The standard deviation of the latencies is negligible in the current example (i.e. invisible in the picture) because neither collisions, nor packet busy event occur.

- δ_{TR} is the packet transfer time resulting from (3.15);
- δ_{ACK} is the time spent in waiting for the acknowledgment message sent by the receiving node. Basically, this value can be estimated by (3.15), simply by setting D=5 bytes and P=6 bytes, respectively [4];
- B is the duration of the backoff period;
- C is the time spent to perform the CCA operation at the PHY layer;
- X_i, for i=0,...,NB_r, is a uniform random variable in the expanding range [0, $2^{BE} - 1$], with BE affecting how many backoff periods a node should wait before a new channel access attempt. In fact, BE is incremented by 1 anytime the channel is sensed busy.

According to the standard IEEE 802.15.4, B=320 μs, C=128 μs, NB_r may range from 0 to 5 and BE may range from 0 to 8 [4]. Observe that in (3.16) only the R-1 acknowledgment delays due to failed transmission attempts are included. In fact, when the packet is transmitted successfully, the final acknowledgment time is superimposed to the receive time. Therefore, if at the first attempt (i.e. for R=1), the channel is idle and the packet transfer is successful, (3.16) becomes:

$$\delta_L = \delta_{TR} + X_0 B + C \tag{3.17}$$

This expression highlights that also the link and physical layer latency grows linearly with the packet size. This is clearly visible in Fig. 3.14, in which the average communication latencies estimated over 100 measurement results are shown as a function of the payload size (from 10 to 100 bytes). Observe that in this case the $\pm\sigma$ limits are not plotted because the latencies values are mostly deterministic. In fact, in TinyOS the first random access delay is already included in δ_S. As a consequence, an initial value of BE larger than 0 is unnecessary and $X_0 B$=0. The difference between the measurement results shown in Fig. 3.14 and the communication latencies estimated using (3.17) is approximately constant and equal to 47 μs, most probably because of the differences between TinyOS communication stack and that described in the standard IEEE 802.15.4. Nevertheless, such deviations are two order of magnitude smaller than the measured δ_L values, i.e. quite small.

3.2.4 Total Communication Delay

Communication Delay as a Function of Packet Length

One-hop communication latencies between two TelosB nodes have been measured for different payload sizes ranging from 10 to 100 bytes under negligible traffic conditions (i.e. when the random traffic generator is off).

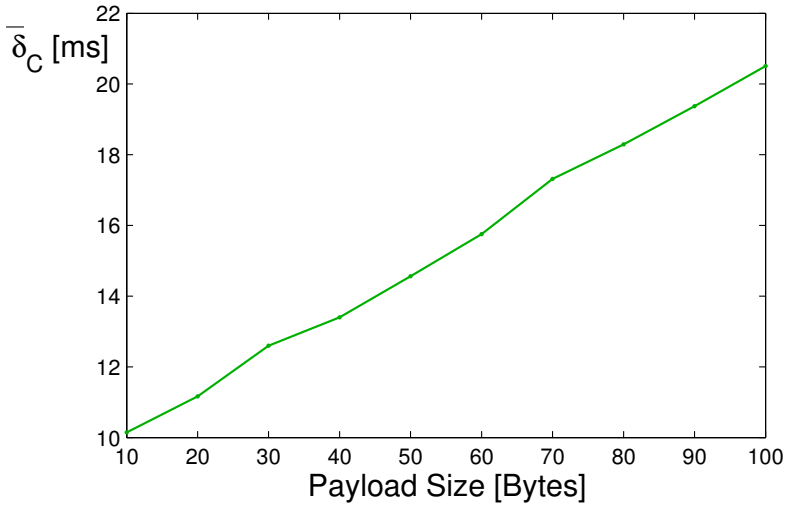


Figure 3.15: Mean values of 100 one-hop communication latencies between two TelosB nodes. The delay is plotted as a function of the payload size in negligible traffic conditions.

Due to additional bytes added at the MAC and physical layers, the whole packet length is longer. One hundred measurements have been repeated for each packet size to evaluate the effect of random contributions. The mean δ_C and standard deviation values σ_C of the measurement results are shown in Fig. 3.15 and 3.16, respectively, as functions of the payload size. Notice that the average communication delay increases almost linearly. In general, the slope of δ_C represents the average time to transfer 1 byte of data between two nodes. Notice that also in negligible traffic conditions the standard deviation of the communication latency may be quite large (in the order of some ms), but it does not depend on the packet payload size. In fact, this random behavior follows approximately a uniform distribution and it is due to the initial channel access delay.

Communication Latency as a Function of Different Traffic Conditions

The communication latencies between two TelosB modules have been measured in different network traffic conditions, after setting the payload size

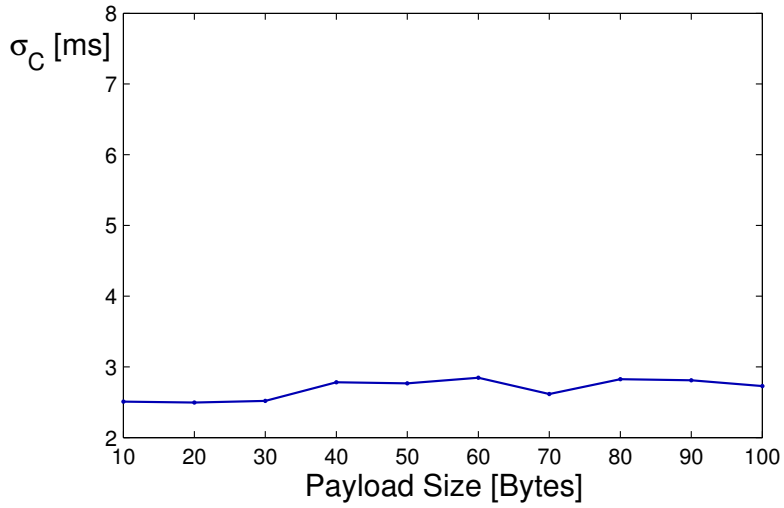


Figure 3.16: Standard deviation values of 100 one-hop communication latencies between two TelosB nodes (as a function of the payload size in negligible traffic conditions).

equal to 10 bytes. The experimental setup is the same as in the first set of experiments, but in this case the random traffic generator is on. The goal of the traffic generator here is to create radio traffic conditions similar to those in real wireless sensor networks containing a large number of nodes. Thus, in this set of experiments, whenever the sender tries to send its packet to the receiver, the communication time is affected by the network traffic. The traffic generated in our experiments is based on the model described in [33], where it is assumed that each node in the network transmits a radio packet during a time interval which is small compared to the time interval between successive transmissions. Moreover, by assuming that the packet inter-arrival times are exponentially distributed, the number of packets circulating in the network will follow a Poisson distribution (in the following referred to as Poisson traffic for brevity). The Poisson traffic is characterized by the mean offered traffic rate G representing the average number of packets arriving during one packet transmission interval. The abovementioned traffic representation was used by researchers to

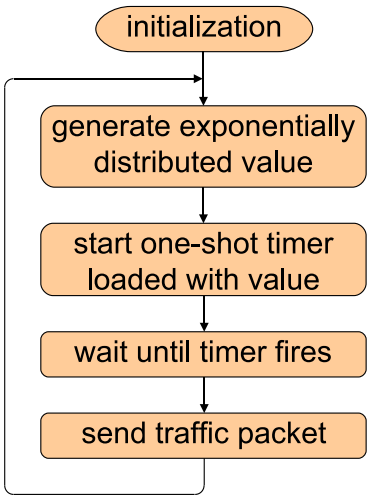


Figure 3.17: Flowchart of the application running on the traffic generator.

model delay distributions in networks based on a CSMA/CA access mechanism. In particular, the offered traffic rate G is related to the mean value β of the exponential inter-arrival time distribution with a probability density function $f(t) = \frac{1}{\beta}e^{-\frac{t}{\beta}}$ for t$\geq$0 by the following expression [76]:

$$G = \frac{\tau}{\beta} \tag{3.18}$$

where τ is the given transmission time of a single packet. According to [33], G corresponds to all packets which nodes try to transmit (including retransmissions) and the actual channel traffic is only a part of that overall number of packets. For our experiments we simplify this and assume that the actual network traffic is a Poisson traffic with a mean rate G. Starting from these assumptions, the traffic generator has been programmed to generate an exponentially distributed random sequence of packets, while the packet transmission between the sending and the receiving node occurs periodically once per second. The random intervals are generated by a routine updating the timer of the traffic generator. In this way, when the timer overflows, the corresponding timer interrupt service routine at first performs some radio sending commands, then it generates the next

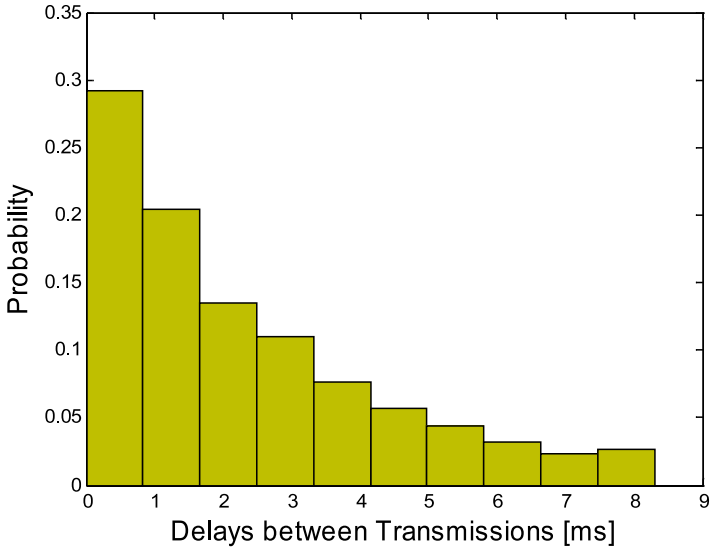


Figure 3.18: Normalized histogram example resulting from 2000 values produced by the traffic generator when G=0.46.

time interval to be loaded into the timer for the subsequent transmission. The flowchart of the application running on the traffic generator is shown in Fig. 3.17. Two thousand consecutive values actually produced by the traffic generator after setting G=0.46 have been read from the node memory to verify the exponential nature of their distribution. Notice that the normalized histogram based on the collected values shown in Fig. 3.18 exhibits quite a good exponential behavior.

Naturally, an increment in traffic should significantly reduce the probability of achieving a successful transmission. In the performed experiments, the sender is able to retransmit each packet up to 6 times in case of collision and the receiving node replies with acknowledgment packet, when a packet is successfully received. This mechanism greatly improves the data transfer reliability, but it contributes (together with the CSMA/CA algorithm adopted on the TelosB nodes) to the communication delay increase when the network traffic grows. It is important to note that the CSMA/CA mechanism has been switched off in the traffic generator in order to disable

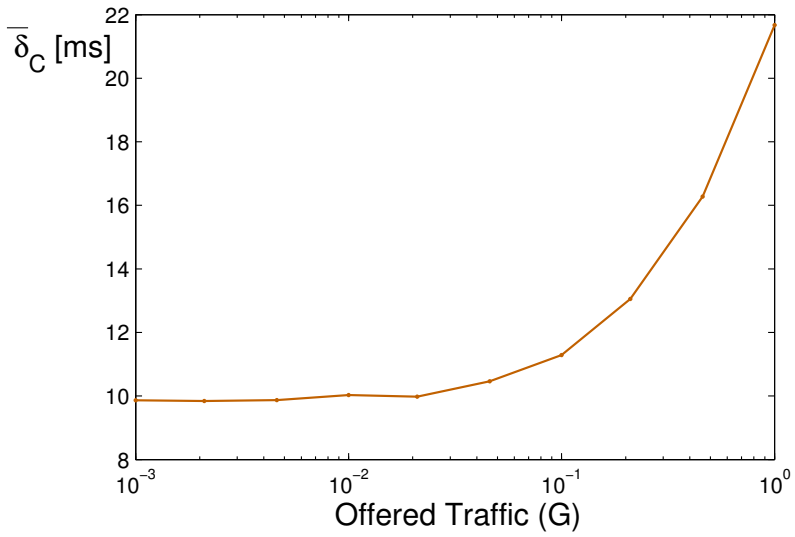


Figure 3.19: Average values of 100 one-hop communication latencies between two TelosB nodes, as a function of various values of the offered traffic parameter G on a logarithmic scale.

any retransmission attempts that could alter the required Poisson traffic behavior. In fact, the data produced by the traffic generator are assumed to emulate already the cumulative effect of the packet transmissions and retransmissions occurring in a possible large network. Thus, no additional random delays should be added to the exponentially distributed time intervals.

In order to evaluate the effect of different amounts of traffic, the average communication latencies and the corresponding standard deviation values estimated over 100 measurement results are plotted in Fig. 3.19 and Fig. 3.20, respectively, for different values of G (namely for $G = 0.001$, 0.0021, 0.0046, 0.01, 0.021, 0.046, 0.1, 0.21, 0.46, 1.0). Fig. 3.19 highlights the growth of the average communication latency as a function of the offered traffic. Observe that also the standard deviation increases at a similar rate when the channel is congested (occasionally, communication delays exceed 90 ms for $G = 1.0$) due to the overall larger number of retransmission attempts.

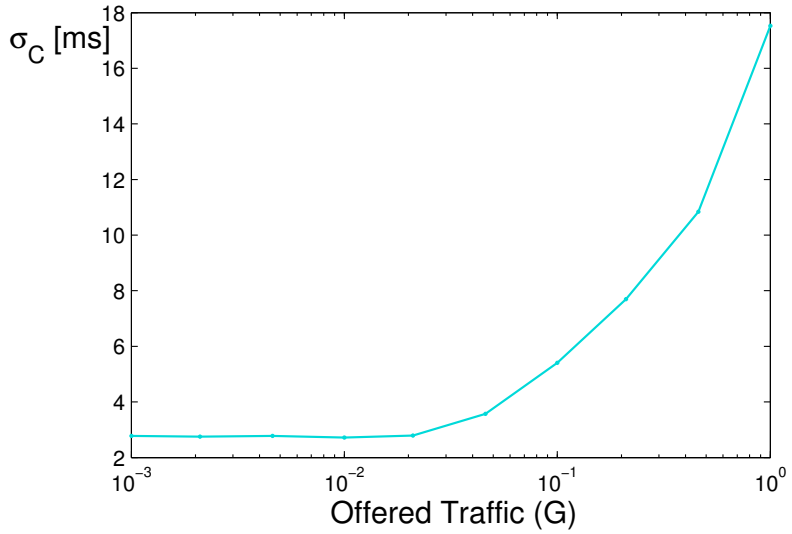


Figure 3.20: Standard deviations values estimated over 100 one-hop communication latencies between two TelosB nodes, as a function of various values of the offered traffic parameter G on a logarithmic scale.

Communication Latency in Multi-hop Links

In the third set of experiments the communication latencies between two TelosB nodes have been measured for different number of hops under negligible traffic conditions (i.e., when the traffic generator is off), after setting the payload size equal to 10 bytes. The experimental setup is the same as in the experiments described above, but in this case the sender communicates with the receiver through a variable number of intermediate "bridge" modules (ranging from 1 to 9). In this set of experiments, whenever a radio packet is received by an intermediate node, it is immediately forwarded to the closest nearby node. Therefore, the total communication latency is the time interval between the rising edge of the pulse generated by the sender (as described in subection 3.2.1) and the rising edge of the corresponding pulse produced by the final receiver.

In Figures 3.21 and 3.22 the average communication latency and the respective standard deviation values estimated over 100 measurement

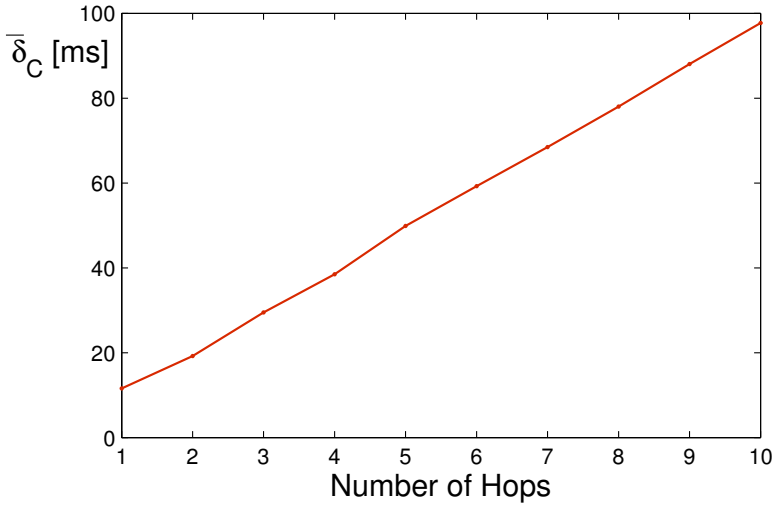


Figure 3.21: Mean values estimated over 100 communication latencies between two TelosB nodes for different numbers of hops in negligible traffic conditions ($G \approx 0$).

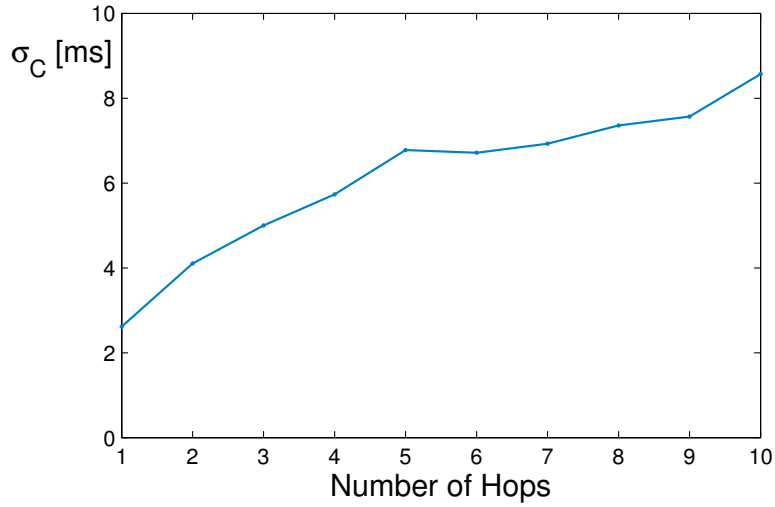


Figure 3.22: Standard deviations estimated over 100 communication latencies between two TelosB nodes for different numbers of hops in negligible traffic conditions ($G \approx 0$).

results are shown as a function of the number of hops in negligible traffic conditions (i.e., for $G \approx 0$). Notice that the average communication latency is still in the order of several ms and it grows linearly, whereas the standard deviation pattern exhibits roughly a square root trend. Furthermore,

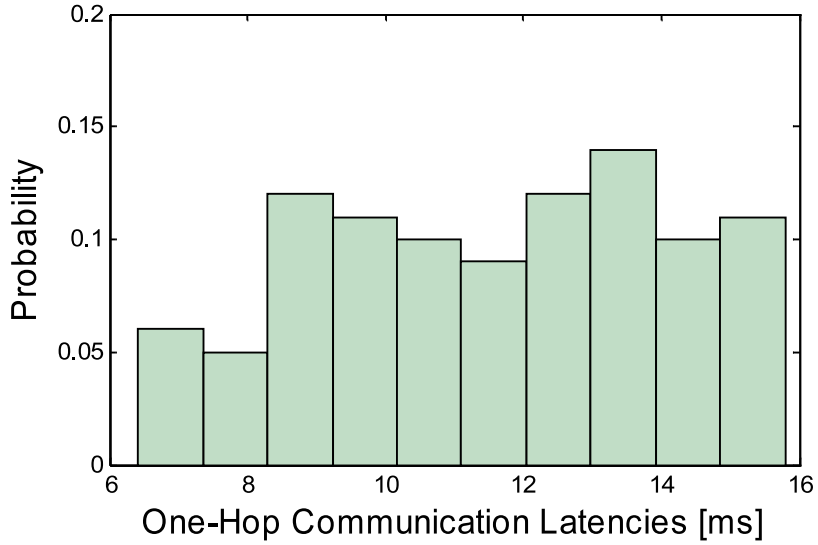


Figure 3.23: Normalized histogram of one-hop communication latencies estimated over 100 measured values.

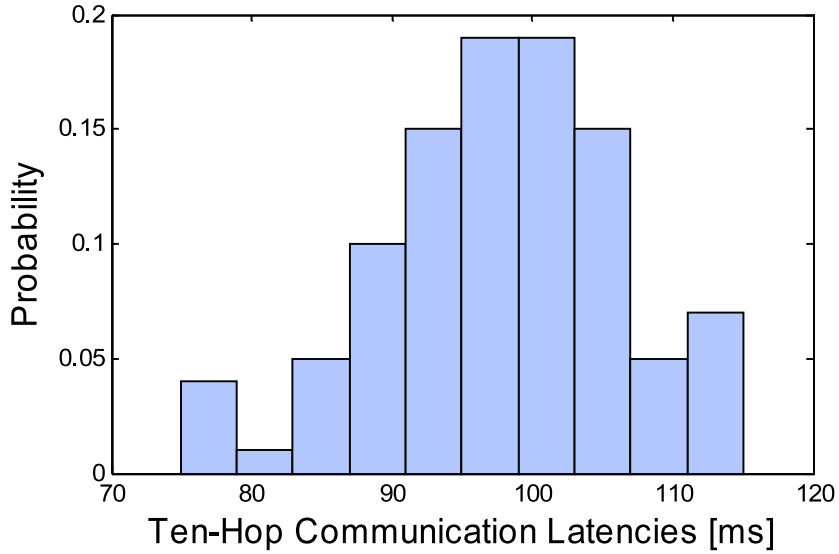


Figure 3.24: Normalized histogram of ten-hop communication latencies estimated over 100 measured values.

while single-hop communication latencies (in accordance with what stated in Subsection 3.2.2) are approximately uniformly distributed as shown in the normalized histogram in Fig. 3.23, in case of multiple hops the distribution tends to become Gaussian. This is clearly visible in the normalized

histogram shown in Fig. 3.24 in the 10-hops case. Such results confirm that, if all WSN nodes are nominally identical, also the various single-hop communication latencies can be assumed to be independent and identically distributed. This agrees with the central limit theorem (CLT), and the mean value and variance of the resulting Gaussian distribution increase linearly with the number of hops.

3.3 Summary

Modeling of time uncertainty contributions in WSN is essential to assess and compare the performance of different synchronization approaches. In this chapter the results of several experiments aimed at estimating such contributions are reported and analyzed. The obtained results provide an interesting quantitative information to determine the maximum timing constraints that can be achieved in time synchronization methods for WSN modules. According to these results, the constant frequency deviation (due to manufacturing imprecisions and temperature offset) is by far the main source of uncertainty affecting time measurements in stable environmental conditions. However, the frequency deviation may vary significantly as a result of changing conditions such as temperature. Also, the uncertainty associated to the time transfer process depends considerably on the network traffic conditions as well as on the size of the transmitted packet and the number of hops. Since the time estimation uncertainty as well as its change in time is caused by multiple factors, time synchronization algorithms should be able to adapt to those factors and maintain the required time synchronization accuracy at the expense of the minimum energy. Such an algorithm is described in the next chapter.

Chapter 4

Adaptive Time Synchronization in WSN

4.1 Benefits of Adaptive Time Synchronization

Because of multiple factors, the clock readings of wireless sensors always drift apart and a certain synchronization procedure must be performed periodically in order to keep the corresponding time uncertainty within limits. The problem of minimizing the number of periodic synchronization events is a critical issue for low-cost WSN nodes provided with limited energy, memory and computational resources. In fact, each synchronization event in WSN involves active operation of node microcontrollers, which execute instructions related to the reading and processing of time values, and radio modules, which transmit and receive packets containing time values. Consequently, the more frequently the synchronization is performed, the more energy it takes.

If a WSN node uses time values received from other nodes simply to renew the information about their clock offsets with respect to its own clock, the synchronization procedure must be repeated rather frequently to keep time uncertainty within limits. Alternatively, the offsets of clock readings can be accurately predicted thus making frequent synchronization

unnecessary and saving energy as well [38]. Indeed, in the previous chapter it has been shown that the uncertainty associated with time measurements performed by wireless sensors is approximately a linear function of time under stable environmental conditions (see section 3.1). In this case the differences in clock readings of WSN nodes grow linearly, and if the rate of this growth is found in some way, it can be used to estimate accurately the corresponding clock readings between any two consecutive synchronizations. However, this involves more complex computations (most likely, several floating point operations) and a larger number of radio transmissions giving more data, which may be necessary to use some statistical methods. Besides, it is evident that in actual applications of wireless sensors, the ambient conditions often may change rapidly [73, 50, 19], which results in a significant instability of the resonator frequencies and, as a result, a significant change in the clock drift rate. In fact, the experiments with heated WSN nodes described in Section 3.1 prove it. Thus, in changing environmental conditions the synchronization accuracy degrades and therefore, the energy-consuming clock drift estimation is not justified, since a simpler synchronization procedure may perform better, providing the same accuracy at a lower cost in terms of energy. Generally, the time synchronization must be performed more frequently under unstable ambient conditions in order to maintain the desired accuracy. On the other hand, if the environmental conditions are stable, it is reasonable to perform synchronization infrequently to save energy. These properties can be supported by adaptive synchronization schemes, which react properly to ambient effects in order to keep the time uncertainty within limits while spending the minimum energy on it.

In this chapter, a flexible adaptive strategy to perform periodic synchronizations on the basis of both target accuracy and in-tolerance probability is described. The proposed algorithm relies on the so-called Simple Re-

sponse Method (SRM) originally conceived to adjust the calibration intervals of a generic measurement instrument according to the result of the last calibration [46]. In fact, the synchronization process can be regarded as a special case of calibration, and the SRM has already proved to be effective in estimating the best calibration interval of primary time references [40]. The main aim of the proposed algorithm is not to attain the best possible accuracy of time synchronization, but to investigate the advantages of an adaptive approach focused on achieving a suitable trade-off between the target accuracy and energy efficiency.

4.2 The synchronization algorithm

4.2.1 Steps of The Algorithm

The main steps of the proposed algorithm will be orderly presented starting from the two following basic hypotheses:

1. The considered WSN is a cluster consisting of a known number of nodes. This means that each node can be reached by any other node within a single hop. Since a group of nodes can be also regarded as a sub-network of a larger WSN based on a clustered topology [4], the extension to the multi-hop case is easy, as it will be discussed later in this section.

2. The term "synchronization" used in the following is to be intended as compensation of the differences in clock readings of WSN nodes, without using any external time reference.

As far as the notation is concerned, in the rest of this chapter the following symbols will be used:

- N - number of cluster nodes;

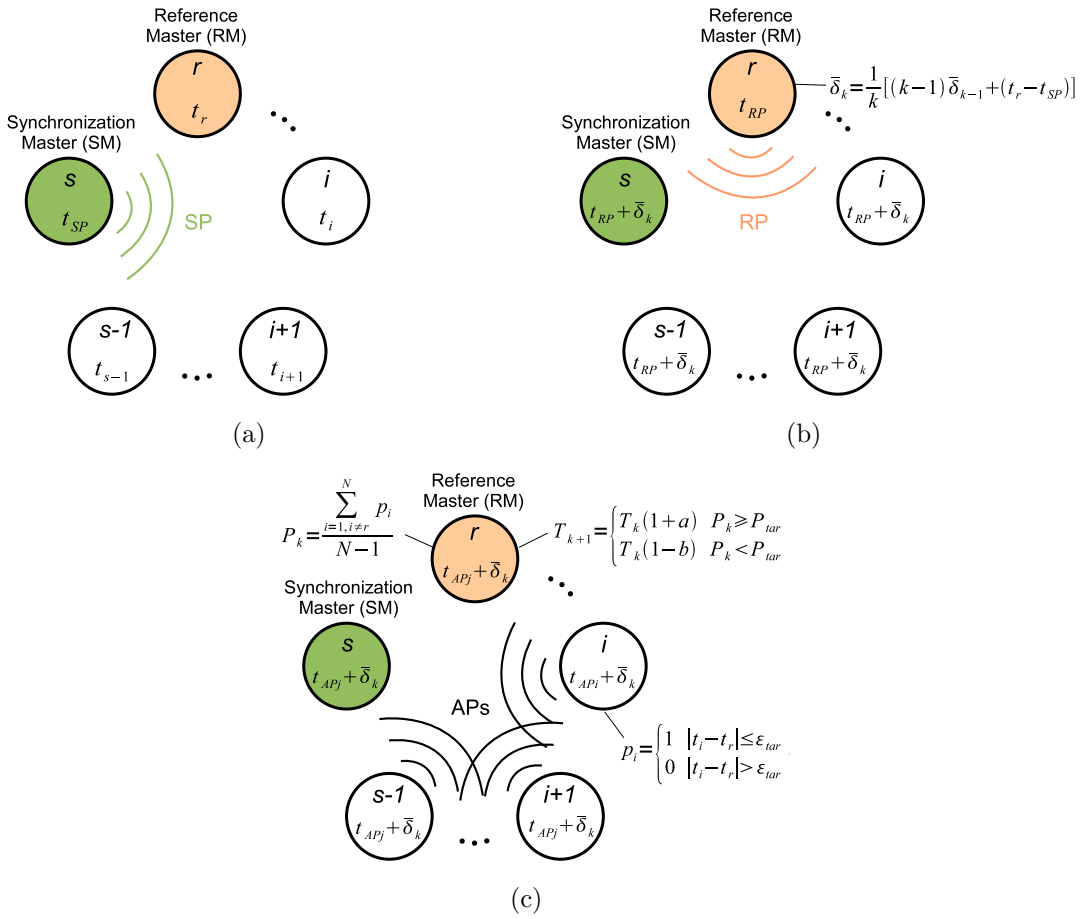


Figure 4.1: Main stages of the synchronization algorithm: broadcast of synchronization packet (a), broadcast of reference packet (b), collection and processing of acknowledgments (c).

- ε_{tar} - target synchronization error;
- P_{tar} - target End-of-Period (EOP) in-tolerance probability, namely minimum wanted probability of using an in-tolerance clock when a new synchronization is executed;
- k - index (starting from 1) representing the current synchronization cycle;
- T_k - duration of the kth synchronization interval;

- T_0 - initial duration of synchronization interval;
- $\overline{\delta_k}$ - estimate of the average latency between the transmission and reception timestamps of a packet transferred between two WSN nodes during the kth synchronization cycle;
- s (starting from 0) - identification number (ID) of the *synchronization master* (SM) of the network in the current synchronization cycle;
- r (starting from 1) - ID of the *reference master* (RM) of the network in the current synchronization cycle;
- SPR_i (for i=0,...,N-1) - binary variable that should be set to 1 whenever the *synchronization packet* (SP) (described in the following) is correctly received by the ith node;
- RPR_i (for i=0,...,N-1): binary variable that should be set to 1 whenever the *reference packet* (RP) (described in the following) is correctly received by the ith node;
- p_i (for i=0,...,N-1): binary variable to be set to 1 if the time difference between the ith node and the RM lies within the synchronization tolerance boundaries $\pm\varepsilon_{tar}$.

The three main phases of the proposed algorithm during a generic iteration are sketched in Fig. 4.1. In the first stage, shown in Fig. 4.1(a), the SM of broadcasts a synchronization packet (SP) containing:

- The values of s and r;
- The kth time interval duration T_k, after which all nodes have to perform a new synchronization;
- The previously computed time value $\overline{\delta}_{k-1}$;

- The value of the time-stamp t_{SP} that is attached to the SP at the MAC layer just before transmission.

When the other N-1 nodes of the WSN receive SP, they also time-stamp the incoming packet at the MAC layer. Such time-stamp values t_i (i.e., with ID $i \neq s$), will be used in the third stage of the algorithm (see later) to evaluate the synchronization status of the WSN nodes according to a receiver-receiver policy [17]. Upon a successful reception of the SP packet, each node will also set to 1 the corresponding local variable SPR_i. In the second stage of the algorithm, shown in Fig. 4.1(b), the RM updates the estimate of the average latency $\overline{\delta}_k$ by means of the following expression:

$$\overline{\delta}_k = \frac{1}{k}[(k-1)\overline{\delta}_{k-1} + (t_r - t_{SP})] \quad k \geq 1 \tag{4.1}$$

where $\overline{\delta}_0$=0 and t_r is the SP reception time-stamp maid by the RM. Even if the estimator (4.1) is approximate, it is easy to implement and quite fast from the computational point of view. In fact, in order to avoid overflow problems and to simplify calculations, the k values in (4.1) could be replaced by suitable powers of 2 within a limited range. After computing $\overline{\delta}_k$, the RM broadcasts a reference packet (RP) containing:

- The values of s and r;
- The interval duration T_k;
- The new value of $\overline{\delta}_k$;
- The local variable SPR_r associated to the RM;
- The SP reception time-stamp t_r ;
- The value of the transmission time-stamp t_{RP} applied to the RP.

All nodes receiving the RP adjust their clocks to $t_{RP}+\overline{\delta}_k$ and set the corresponding RPR_i variable to 1. In the third step of the algorithm, shown

in Fig. 4.1(c), each node of the WSN (except the RM) checks if both the received value of SPR_r and the local variable SPR_i are equal to 1. If this condition is true, this means that the SP was correctly received by both the ith node and the RM. Thus, the value of the binary variable p_i is given by:

$$p_i = \begin{cases} 1 & |t_i - t_r| \leq \varepsilon_{tar} \\ 0 & |t_i - t_r| > \varepsilon_{tar} \end{cases} \qquad i \neq r \tag{4.2}$$

In order to find p_i, SM can compute the difference of $t_{SP} + \overline{\delta}_k$ and t_r (since SM does not receive its own packet and cannot time-stamp it). If either SPR_r or SPR_i is equal to 0, the SP was not properly received by (at least) one of the nodes and the value of p_i can be set randomly to 0 or 1, because there is no information to perform any synchronization comparison. Then, the variable p_i (for $i \neq r$) is included into an acknowledgment packet (AP) that is broadcasted by the ith device to all the others after a preset amount of time. Of course, in order to avoid possible collisions over the wireless channel, the waiting time of the various nodes before transmission must be different (e.g. it can be proportional to the node ID number i). Note that, together with p_i, the AP message contains also:

- The values of s and r;
- The interval duration T_k;
- The value of $\overline{\delta}_k$;
- The local variables SPR_i and RPR_i;
- The SP reception time-stamp t_i;
- The AP transmission time-stamp t_{APi}.

By collecting the various APs, all nodes are informed about the overall synchronization status of the network. Depending on the values of the

received variables SPR_i (for $i \neq s$) and RPR_i (for $i \neq r$), various alternative situations are possible. Normally (i.e. if both SPR_i and RPR_i are equal to 1) the synchronization procedure is performed correctly and the EOP in-tolerance probability before running the kth synchronization results from:

$$P_k = \frac{\sum_{i=1, i\neq r}^{N} p_i}{N-1} \tag{4.3}$$

Accordingly, the duration of the following synchronization interval is set on the basis of the comparison between P_k and the target EOP in-tolerance probability P_{tar}, i.e. [46]

$$T_{k+1} = \begin{cases} T_k(1+a) & P_k \geq P_{tar} \\ T_k(1-b) & P_k < P_{tar} \end{cases} \qquad i \neq r \tag{4.4}$$

where the parameters $a > 0$ and $0 < b < 1$ rely either on engineering experience or on suitable reliability models, in accordance with the definition of SRM. In particular, it is known that the SRM parameters are related to the asymptotic EOP in-tolerance probability through the following approximate expression [40]:

$$\lim_{k\to\infty} P_k = \frac{\log(1-b)}{\log(\frac{1-b}{1+a})} \tag{4.5}$$

Notice that if b and P_{tar} are given, the value of a can be univocally derived from (4.5), assuming that $P_{tar} = \lim_{k\to\infty} P_k$.

If some RPR_i value is equal to 1, but all SPR_i values are 0, some synchronization events occurred, but the SM did not work properly. Thus, the timestamps t_i necessary to check the synchronization status of the network were not collected. In this case, the most reasonable approach is to leave T_k unchanged (i.e. $T_{k+1} = T_k$) and to perform a new synchronization after T_k. On the contrary, if all RPR_i values are equal to 0, no synchronizations occurred due to some problems of the RM. In this case the value of the

synchronization interval T_k is still kept, but a new synchronization cycle starts as soon as possible (e.g. after 1 s) using a different RM. After the end of the third stage, each node i that did not receive the RP (i.e., having RPR_i=0) may use the time-stamp of the last received AP from any other node j with RPR_j=1 to adjust its own clock to $t_{APj} + \overline{\delta}_k$. Afterwards, all nodes have to reset their local variables SPR_i, RPR_i and p_i and to set the variables s and r (identifying the roles of SM and RM for the next iteration) to r and $(r + 1)$ mod N, respectively. Eventually, all nodes are free to perform their normal monitoring activities or to enter a low-power mode until the next synchronization interval expires. As stated at the beginning of this subsection, the proposed procedure can be extended also to more complicated clustered topologies. In fact, each cluster of nodes can self-synchronize autonomously using the procedure described above, whereas adjacent clusters can be synchronized thanks to the clocks of the cluster-head nodes servicing as a bridge between clusters, when cluster-heads operate as RM.

4.2.2 Fault Condition Management

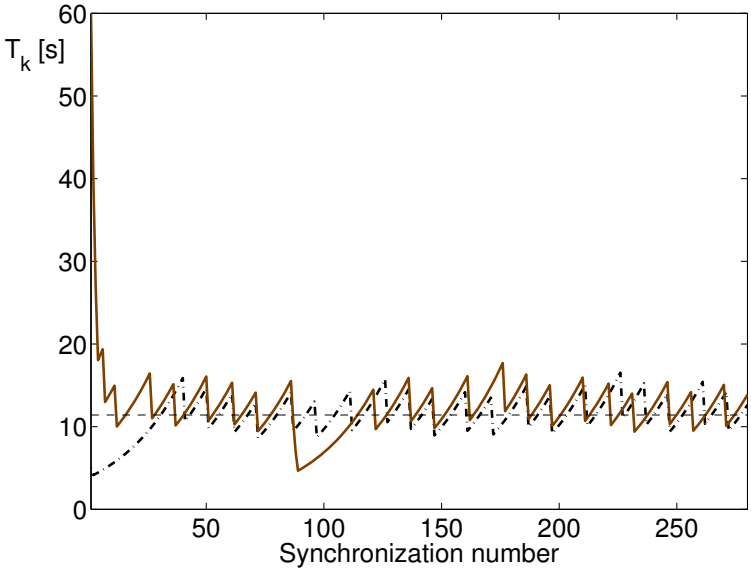
The continuous adjustment policy of the synchronization intervals requires that all the messages are correctly received by all nodes. This is a quite hard assumption. However, even if some node is faulty or temporarily out-of-range, the whole procedure keeps on working. Assume that, at a certain point of time, SM fails, but RM works properly. In this case, all available nodes are synchronized by RM and the interval adjustment policy is just temporarily suspended in the current cycle. Conversely, if RM fails but SM works, a new synchronization cycle restarts using another RM. Finally, if both SM and RM fail at the same time, the whole network must rejoin: in this case the first successful synchronization will occur after two subsequent failed synchronization cycles.

Notice that all the packet types described above (i.e. SP, RP and AP) generally contain redundant data. This is done to improve the dependability of the network, i.e. to maximize the probability that all the WSN nodes retain the same information also if some packets are lost.

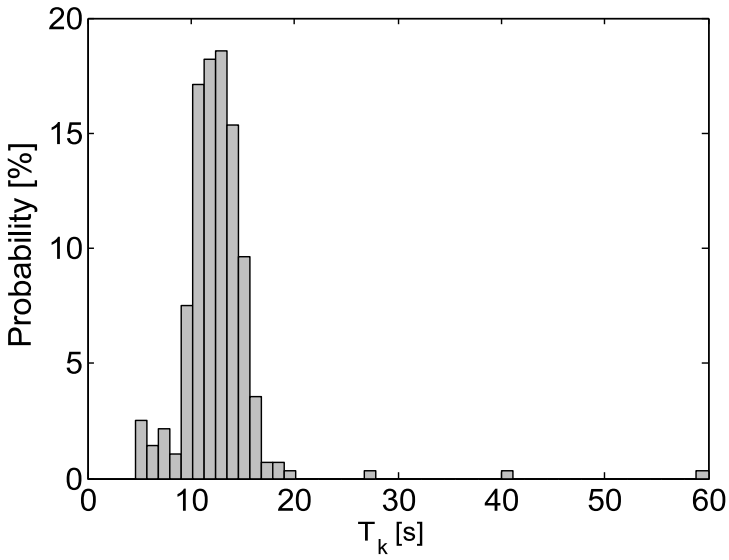
A further critical problem affecting the proposed algorithm is that if the chosen value of ε_{tar} is too stringent compared to the real clock stability, the T_k values could converge to 0, thus making the number of synchronization events unrealistically large. In order to address this issue, the minimum duration of the synchronization intervals is set equal to 1 s, which is a sensible lower bound in real applications. In this way, even if excessively stringent ε_{tar} boundaries are not met, at least the whole procedure keeps on working according to a best-effort policy.

4.3 Experimental Results

The adaptive synchronization algorithm described in Section 4.2 was implemented on some TelosB and Tmote Sky nodes using the low-level primitives of the well-known operating system TinyOS [26]. The experimental setup used to run the tests is quite simple: N=10 identical nodes were programmed to work autonomously for a given amount of time during which the nodes were synchronized to the changing RM. All synchronization messages were collected by a receiving node acting as a sniffer connected to a PC. Fig. 4.2(a) shows two sequences of T_k values as a function of the synchronization number for ε_{tar}= 0.5 ms, P_{tar}=95%, b=0.33 (the value of a instead results from (4.5)) and T_0=4 s (dashed-dotted line) or T_0=60 s (solid lines), respectively. The T_k values tend to be distributed around 12.6 s (dashed line) according to a lognormal probability density function. Such lognormal behavior is confirmed by the histogram shown in Fig. 4.2(b) for T_0=60 s and it is in good accordance with the experimental



(a)



(b)

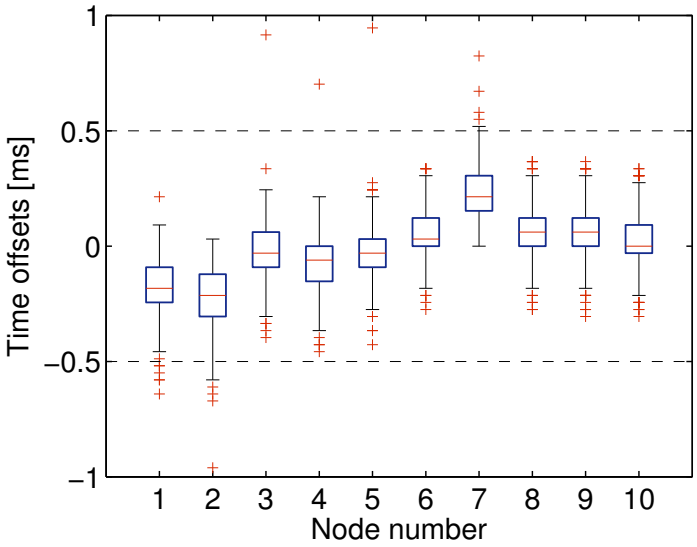
Figure 4.2: Sequence (a) and histogram (b) of the synchronization interval values T_k collected after 1 hour using N=10 TelosB and Tmote Sky nodes for P_{tar}=95%, ε_{tar}=0.5 ms and b=0.33. The solid line in (a) and the histogram in (b) refer to the case when T_0=60 s, whereas the dashed-dotted line in (a) results from T_0=4 s.

results about the calibration of Cesium atomic clocks [40].

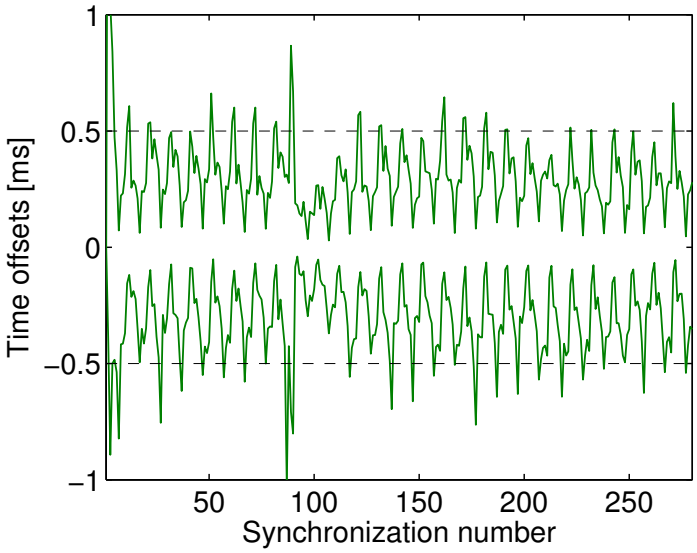
In Fig. 4.3(a) and 4.3(b) the offsets between the time provided by the SM and the time measured by any other node of the WSN just before performing a synchronization are compared with the tolerance boundaries $\pm\varepsilon_{tar}$. Fig. 4.3(a) shows the box-and-whiskers diagram of the time offsets of the various WSN nodes (i.e. for i=0,...,9). Box-and-whiskers diagrams are particularly useful to display the degree of dispersion and the skewness in the data and to identify outliers without making any hypotheses of the underlying statistical distribution. The spacing between the different parts of the "box" refer to the lower quartile, the median, and the upper quartile values. The "whiskers" extending from the lower and upper sides of the boxes show the range of the rest of the data. Outliers are highlighted as crosses.

In Fig. 4.3(b) the upper and lower limits of the confidence intervals containing the time offsets with probability equal to 95% are plotted as a function of the synchronization number. Notice that both upper and lower time offset patterns exhibit roughly a sawtooth behavior with a period equal to the number of nodes (i.e. N=10). This is due to the lack of any clock rate drift compensation in the current version of the algorithm. Nevertheless, except for few outliers, the tolerance limits are generally met. Notice that the values of $\overline{\delta}_k$ are not shown because in our experiments they converge to 0. In fact, when MAC-layer time-stamping is used, the difference between sending and receving time-stamps is in the order of some μs (see subsection 2.5.4), i.e. much smaller than the resolution of the timers employed (i.e. about 30.5 μs).

In order to test the operation of the proposed procedure in different environmental conditions, the previous experiment was repeated several times. In one of them, two different nodes were exposed to the flow of the heated air, similarly as described in Subsection 3.1.2. The sequence of T_k



(a)



(b)

Figure 4.3: In (a) the box-and-whiskers plot of the time offsets of 10 TelosB/Tmote Sky nodes just before each synchronization is shown. In (b) the 95% confidence interval of the same time offsets is plotted as a function of the synchronization number. Both graphs result from 1 hour of operation of the WSN after setting T_0=60 s, P_{tar}=95%, ε_{tar}=0.5 ms and b=0.33.

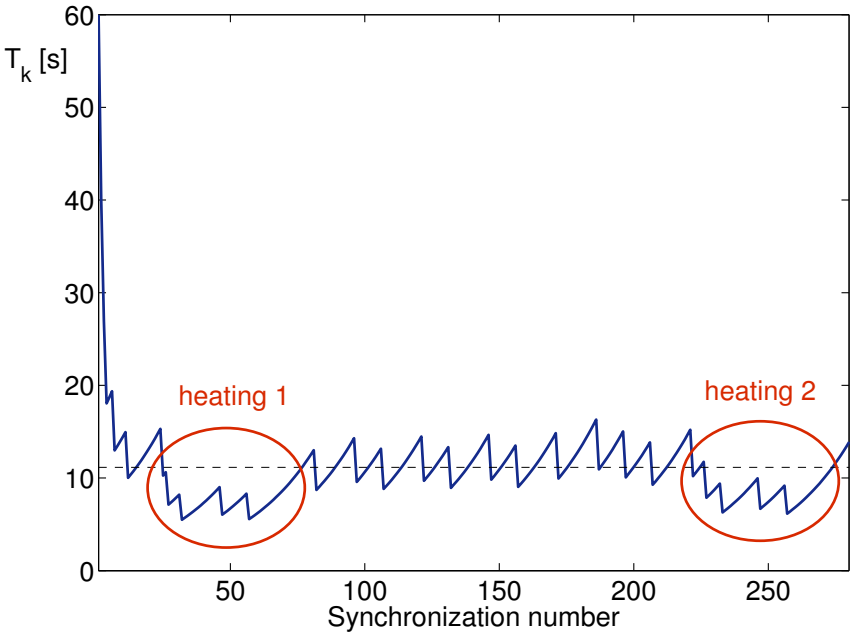


Figure 4.4: Sequence of synchronization interval values collected over 1 hour when two different nodes are heated. The parameters of the experiment are the same as before, i.e. P_{tar}=95%, T_0=60 s, ε_{tar}=0.5 ms and b=0.33.

values related to this experiment is shown in Fig. 4.4. Observe that, when the temperature difference between nodes becomes larger, the duration of the synchronization intervals tends to decrease, which is in accordance with the experimental results reported in Subsection 3.1.2. In fact, the temperature increase causes the change of the oscillator frequency, and the clock offsets of the nodes grow. Therefore, more nodes report p_i=0, and new synchronization intervals computed by (4.4) are shorter. As a consequence, the total number of synchronization events over 1 hour increases.

Further experimental results for various target synchronization errors ε_{tar} (0.15 ms, 0.5 ms, 1 ms and 2 ms) are reported in Table 4.1 for P_{tar}=90%, b=0.33 and T_0=60 s. These results highlight clearly that both the average and the standard deviation of the synchronization intervals grow considerably if a larger ε_{tar} is chosen. Thus, the value of ε_{tar} (i.e. the target

Table 4.1: Estimated average and standard deviation values of the synchronization intervals collected during 1 hour of operation of the WSN for P_{tar}=90%, b=0.33, T_0=60 s and different values of ε_{tar}. The average EOP in-tolerance probabilities are also reported in the rightmost column.

ε_{tar} **[ms]**	Number of synchronization events K	Average synchronization interval [s]	Std. deviation of synchronization interval [s]	Average EOP in-tolerance probability [%]
0.15	851	3.9	1.8	98
0.5	280	12.6	2.6	98
1	132	27.0	4.9	98
2	69	52.0	7.1	99

accuracy) can be traded for the overall power consumption, as desired. In fact, the number of synchronization events over 1 hour is reduced approximately by a factor 12 if ε_{tar} grows from 0.15 ms to 2 ms. Notice also that the estimated average EOP in-tolerance probabilities (shown in the rightmost column of Table 4.1) are a bit larger than the target value P_{tar}. This is due to the fact that (4.5) relies on the assumption of exponentially or Weibull-distributed reliability models [40], which does not hold in the case considered.

In order to prove the energy saving capabilities of the developed synchronization algorithm, additional experiments were performed using a small wireless network composed of 5 TelosB motes. A sixth wireless node acted as a sniffer, as in the experiment described above. In these experiments, the nodes deactivated their radios between synchronization events to save energy. The power measurements were performed using two digital multimeters Agilent 34411A, one used as a voltmeter and the other as an amperometer, as shown in Fig. 4.5. With this setup we repeated four experiments after setting ε_{tar} = 0.15, 0.5, 1 and 2 ms, respectively. 1000

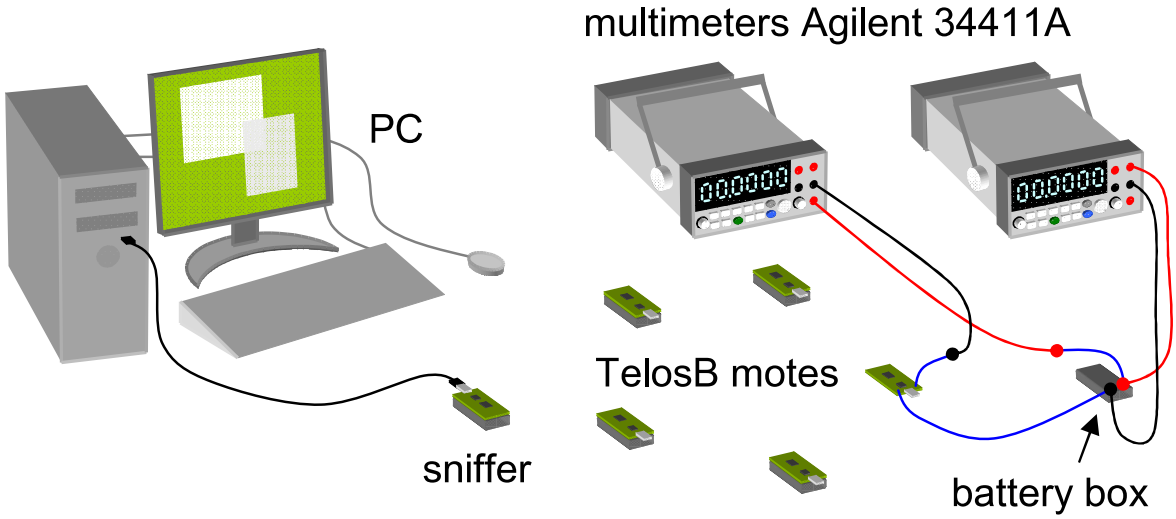


Figure 4.5: Experimental setup to measure the power consumption of WSN nodes running the developed adaptive algorithm.

Table 4.2: Average current drain, supply voltage and power consumption over 1000 measurements for different values of ε_{tar}.

ε_{tar} **[ms]**	$\bar{I}$ [μA]	$\bar{V}$ [V]	$\bar{P}$ **[mW]**
0.15	342	2.8305	0.97
0.5	96	2.8317	0.27
1	54	2.8302	0.15
2	24	2.8275	0.07

simultaneous current and voltage measurements were collected in each experiment for P_{tar}=95%, b=0.33 and T_0=60 s. The integration time of both multimiters was set to 100 power line cycles (i.e. 2 s) in order to obtain the maximum instrument resolution and to reject the normal mode noise. The average power consumption $\bar{P}$ was calculated as a product of the average current $\bar{I}$ and the average supply voltage $\bar{V}$ measured during the algorithm operation. The corresponding values are reported in Tab. 4.2. These results emphasise that our adaptive synchronization policy actually saves power of WSN nodes.

4.4 Summary

This chapter examines the advantages of adaptive time synchronization and presents an algorithm aimed at achieving the target synchronization accuracy at the minimum cost in terms of the number of synchronization events. Several experimental results show that the algorithm converges. In fact, after a transient phase, the synchronization intervals tend to exhibit a lognormal distribution around the optimal value. If no transmission problem occurs, the probability of keeping nodes synchronized within the wanted tolerance boundaries is in the order of 99%. Also, the algorithm is able to withstand deviations of oscillator frequency caused by the change in temperature, as the experiments showed. The measurements proved that the algorithm reduces the power consumption of WSN nodes. However, reducing the cost of synchronization at the expense of synchronization accuracy does not always have an optimum effect on energy saving, which is studied in the next chapter.

Chapter 5

Energy Efficient Synchronous Operation of WSN

5.1 Time Synchronization for Monitoring and Communication

As shown in the previous chapter, the frequency of synchronization events can be reduced to the minimum possible, while the target synchronization accuracy is achieved with a high probability. In this way significant energy savings can be achieved, resulting in longer lifetime of WSN. However time synchronization itself can help to save energy in WSN, and these energy savings may outweigh the cost of synchronization in terms of energy.

Whenever a wireless sensor node is not required to perform continuous sensing, communication or processing operations, all unused node components (e.g. sensors, radio modules or microcontrollers) could be switched off in order to save energy [27, 60, 11, 54, 25]. In particular, as the radio module is usually the most power–hungry component of a WSN node, reducing its duty cycle as well as the amount of transferred data can significantly prolong the battery life. Also, the energy spent by sensors may be sometimes comparable to or even more than the energy spent by radio modules [7]. If sensors and radio modules operate in a coordinated way, their

activation and deactivation must be also coordinated. Indeed, when a WSN performs network-wide measurement of a certain quantity, the sensors of the nodes must be already switched on at the moment when simultaneous measurements have to be taken. The same applies to the radio modules of the communicating nodes when they need to exchange the measurement results. Obviously, this synchronous monitoring and communication require time synchronization between network nodes. When the frequency of synchronization events decreases, the energy cost of synchronization–related activities also diminishes, but the corresponding time uncertainty grows. As a result, the time intervals in which the radio modules and sensors must be fully operational have to be extended in order to include the worst-case time uncertainty and to assure reliable inter–node communications and simultaneous measurements. Of course, this extension in turn causes an increment in power consumption. Therefore, there is a need for a trade-off between energy savings and energy consumption associated with time synchronization. In this chapter, a way to achieve such a trade-off is investigated.

5.2 Model

5.2.1 Assumptions and qualitative description

The model described in this section relies on some basic assumptions that are shortly summarized in the following:

- The WSN consists of N nodes with the same hardware and software features;

- The network has a cluster–tree topology as shown in Fig. 5.1. This topology includes the star and the linear topologies as special cases.

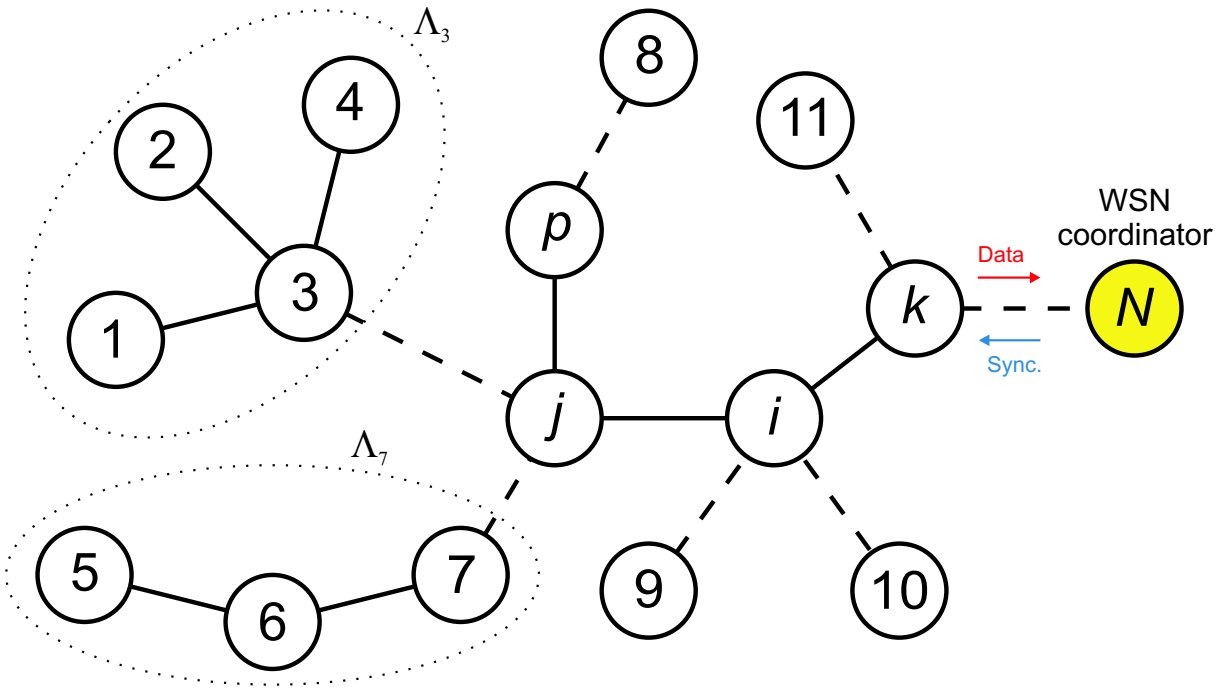


Figure 5.1: WSN topology. The root of the tree (WSN coordinator) gathers the data collected by all the other nodes and broadcasts the synchronization beacons, which are spread throughout the network.

Indeed, they are probably the most used topologies in industrial applications.

- The distance between nodes is in the order of at most 10 m, which requires a small transmission power;
- each node executes M distinct *monitoring tasks* of period T_m (with $m = 1, \ldots, M$) and one *synchronization task* of period T_s;
- The duty cycle of each node, namely the ratio between the time during which a node is running any task over the least common multiple of the time periods of all tasks is much smaller than 1. This is reasonable in low-power WSNs and assures that task scheduling is always feasible.
- The power consumption of each node in the receiving and transmitting states are approximately the same, in agreement with what reported

in [10].

All nodes are normally in low–power mode and should wake up just to run either monitoring tasks or the synchronization task. In the proposed model, the sensor data flow and the synchronization packet flow within the network are opposite as the WSN coordinator (namely the root of the tree) is both the device collecting all sensed data (e.g., before sending them to a PC or to a gateway node) and the *Synchronization Master* (SM) of the network. When a monitoring task is executed, every WSN node does the following actions:

1. it performs one or multiple measurements and collects the corresponding sensor data;
2. it switches on the radio module;
3. it transmits one packet containing its own data to its parent node;
4. it receives all messages from all child nodes possibly acknowledging each received packet;
5. it relays such messages to the parent node and waits for the corresponding acknowledgment packets (if enabled) or until a predefined time–out interval expires;
6. it enters into sleep mode again.

Notice that the amount of transferred data as well as the duration of the whole task tends to grow at least linearly with the number of descendants. In fact, if Λ_i is the subtree rooted in node i (as shown in Fig. 5.1) and L_i is the number of elements of Λ_i, then node i receives $L_i - 1$ data packets and transmits L_i data packets, respectively, with L_i ranging from 1 (leaves of the tree) to N (WSN coordinator). Of course, the total task duration also greatly depends on the channel access mechanism.

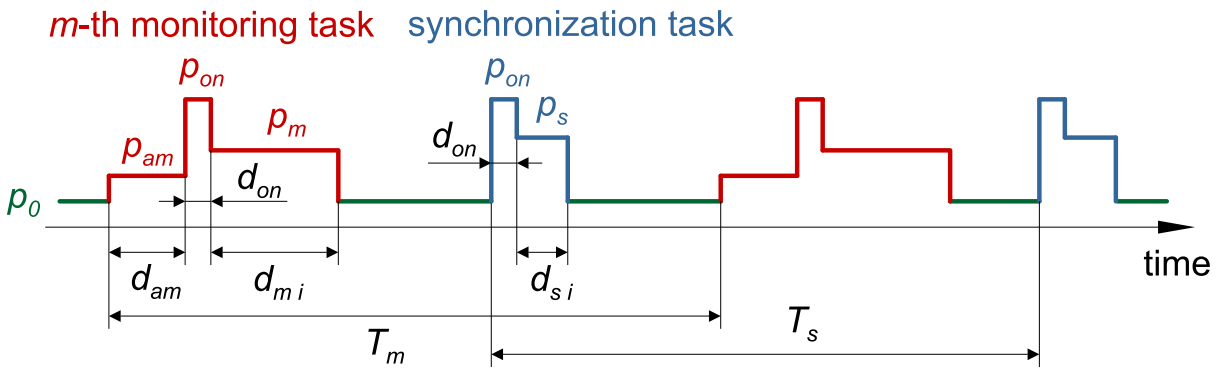


Figure 5.2: Qualitative power consumption waveform of node i in ideal conditions.

The case of the synchronization task is quite different. In fact, time synchronization is initiated by the WSN coordinator and it is further propagated throughout the network. Each node synchronizes its children by broadcasting timestamped beacons that are used to correct the children's clocks. If the beacons are timestamped at the MAC layer, no round–trip delay estimation is required, since in short–range WSNs the packet propagation time is negligible compared to clock period (see subsection 2.5.4). This greatly simplifies the synchronization process because broadcasted packets usually are not acknowledged. Thus, anytime a synchronization task is executed, every WSN has to:

1. switch on the radio module;
2. wait for the synchronization beacon from the parent;
3. correct the local clock;
4. broadcast a new beacon containing the new corrected value;
5. enter into sleep mode again.

A qualitative waveform representing the power dissipated by a generic node i (with $i = 1, \ldots, N$) according to the proposed model is shown in Fig. 5.2, where:

- p_0 is the power consumption when the radio module is off;
- p_{am} (with $m = 1, \ldots, M$) is the average power consumption associated with the sensor data acquisition process during the $m-$th monitoring task;
- d_{am} (with $m = 1, \ldots, M$) is the corresponding sensor data acquisition time;
- d_{on} is the start–up time of the transceiver [72];
- p_{on} is the power dissipated to switch on the radio module;
- p_m (with $m = 1, \ldots, M$) is the equivalent average power dissipated by each node to receive, to relay and to transmit the sensor data collected during the $m-$th monitoring task;
- $\boldsymbol{d}_{m_i}$ (with $m = 1, \ldots, M$) is the time spent by node i to complete the $m-$th monitoring task;
- p_s is the equivalent average power dissipated by each node to run the synchronization procedure and to transmit its own synchronization beacon to nearby devices;
- $\boldsymbol{d}_{s_i}$ is the total execution time of the synchronization task on node i.

All the power coefficients as well as the time parameters d_{am} for $m = 1, \ldots, M$ and d_{on} can be regarded as constants because they just depend on the hardware features of the WSN nodes as well as on the type of sensors used in the $m-$th monitoring task. On the other hand, the terms $\boldsymbol{d}_{m_i}$ and $\boldsymbol{d}_{s_i}$ are random variables which depend on the channel access scheme, the packet size, the hierarchy of node i in the network as well as the type of retransmission policy in case of unsuccessful transmission attempts. Defining a general expression for $\boldsymbol{d}_{m_i}$ and $\boldsymbol{d}_{s_i}$ is not possible

without more precise assumptions about the communication scheme. We assume that CSMA/CA is used, however, the analysis described above is absolutely general and holds even if other access techniques are used (e.g. TDMA). Also, in general both $\boldsymbol{d}_{m_i}$ and $\boldsymbol{d}_{s_i}$ have to be limited in practice. In fact, in order to prevent possible deadlocks, any task should be stopped after a given time–out interval expires. Such time–out intervals $\hat{d}_{m_i}$ and $\hat{d}_s$ must be much larger than the time spent by node i to receive and to transmit the data packets or the synchronization beacons. However, while $\hat{d}_{m_i}$ in principle depends on the position of the node in the tree, i.e. on the amount of incoming and outgoing traffic, $\hat{d}_s$ is independent of the hierarchy because one single synchronization beacon has to be received and broadcasted.

5.2.2 Effects of Time Uncertainty

WSN clocks tend to drift away from one another, because oscillator frequency may change over time due to various factors, as described in previous chapters (see sections 3.1 and 4.1). Thus, the clock offsets of the WSN nodes with respect to the ideal time after T_s (i.e. just before performing a new synchronization) can be regarded as zero–mean independent and identically distributed (i.i.d.) random variables $\boldsymbol{\varepsilon}_n \in [-\varepsilon_{max}, \varepsilon_{max}]$. Therefore, the absolute value of the maximum time offset between any pair of network nodes is $2\varepsilon_{max}$, which may have critical consequences on both communication reliability and power consumption if we assume that all nodes, normally in sleep mode, just wake up occasionally. In order to clarify this point, let us consider the link between nodes i and j shown in Fig. 5.1, where node j is programmed to transfer both local and relayed data to node i, while node i sends a synchronization beacon to node j. Without loss of generality, let us analyze the two following worst-case scenarios, i.e.

- when the clock of node i is in advance by $2\varepsilon_{max}$ with respect to node j;
- when the clock of node i is late by $2\varepsilon_{max}$ with respect to node j.

The qualitative power waveforms of node i and node j in either case are shown in Fig. 5.3(a) and Fig. 5.3(b), respectively. In the former case, the time interval in which node i is fully powered (i.e., ready to receive) is $2\varepsilon_{max}$ longer than expected. Also, node j misses the synchronization beacon (black rectangle), because it wakes up too late. Therefore, node j must wait for some other synchronization beacon or till the time interval $\hat{d}_s$ expires.

In the case shown in Fig. 5.3(b) we have the opposite situation. Indeed, the data packet sent by node j is lost because at that time node i is still in sleep mode. Consequently, in order not to miss information, node j should retransmit the same sensor data again, thus wasting energy. However, the synchronization beacon sent by node i is received correctly by node j and $\boldsymbol{d}_{s_j}$ is longer by $2\varepsilon_{max}$ compared with the ideal case. If we generalize the considerations above, the two following properties hold.

1. In order to avoid possible communication failures, either the sending node has to postpone any data transmission by $2\varepsilon_{max}$ or the receiving node has to wake up $2\varepsilon_{max}$ before the scheduled time. For simplicity, only the latter case will be analyzed in the following.
2. The data reception is actually delayed if the sender clock is slower than the receiver clock. This delay is equal to the corresponding clock offset, whose absolute value in the worst case is $2\varepsilon_{max}$.

If we assume that the clocks of different nodes do not drift away significantly during task execution, and we apply the two properties mentioned above, the maximum duration of the $m-$th monitoring task running on node $i = 1, \ldots, N$ is

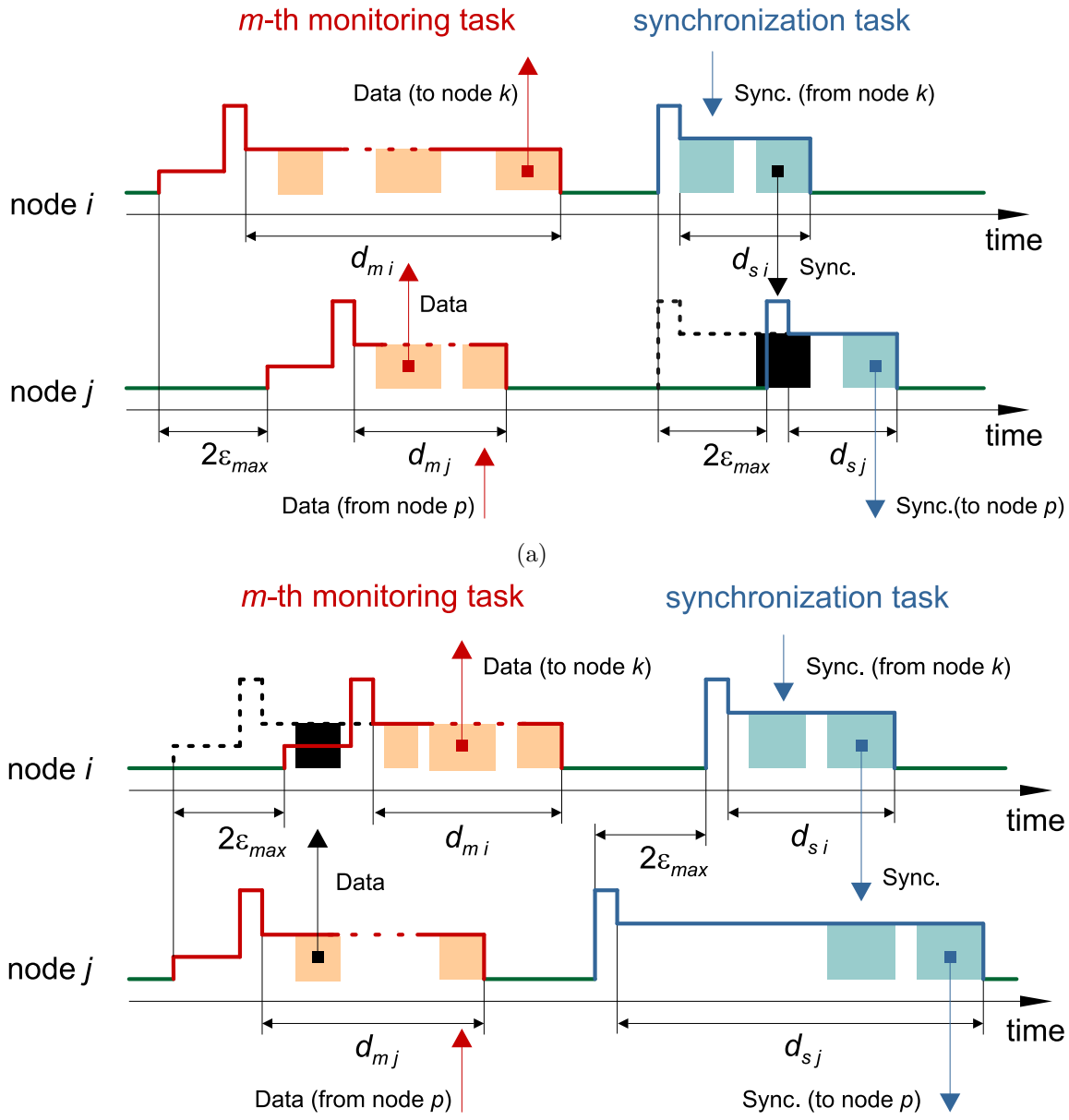


Figure 5.3: Power waveforms of two nodes in the case of 1–hop link. In (a) node i is in advance by $2\varepsilon_{max}$ with respect to node j. In (b) the opposite situation is shown. Shadowed areas represent the time intervals when data packets or synchronization beacons are transferred. The black rectangles highlight the lost packets. The white gaps between shadowed areas represent the time spent to access the channel. Time offset are purposely exaggerated for illustration purposes.

$$d'_{m_i} = \alpha_i \varepsilon_{max} + \hat{d}_{m_i} \tag{5.1}$$

where $\hat{d}_{m_i}$ is an upper bound of $\boldsymbol{d}_{m_i}$ (the time spent by node i to complete the $m-$th monitoring task), and the coefficient α_i is defined as

$$\alpha_i = \begin{cases} 0 & \text{if } L_i = 1 \\ 4 & \text{otherwise} \end{cases} \tag{5.2}$$

In fact, the leaf nodes of the WSN (i.e., those with $L_i = 1$) do not receive any packet from any other device. Similarly, the maximum duration of the synchronization task running on node $i = 1, \ldots, N$ is

$$d'_{s_i} = \beta_i \varepsilon_{max} + \hat{d}_{s_i} \tag{5.3}$$

where $\hat{d}_{s_i}$ is an upper bound of $\boldsymbol{d}_{s_i}$ (the total execution time of the synchronization task on node i), and the coefficient β_i is defined as:

$$\beta_i = \begin{cases} 0 & i = \text{SM} \\ 4 & i \neq \text{SM} \end{cases} \tag{5.4}$$

because the SM does not need to receive any synchronization beacon.

5.3 Power Consumption Analysis

The aim of this section is to determine the power consumption of a generic WSN node, in order to estimate the time interval between subsequent synchronizations which minimizes the energy dissipation of the network. To this purpose, the problem of task overlap must be preliminarily considered. Two or more instances of different tasks may overlap, i.e. one task instance may be scheduled for the time when an instance of another task is already running. If first-come-first-served scheduling is employed, the total execution time is roughly equal to the sum of the individual execution times of the various tasks. The main difference compared to the

case of non–overlapping tasks is that the radio module is switched on only once, instead of two or more times. On the basis of these assumptions, let T be the least common multiple (LCM) of periods T_s and T_m, with $m = 1, \ldots, M$. By definition, the average power dissipated by node i over T is

$$\boldsymbol{P}_i = \sum_{m=1}^{M} \boldsymbol{P}_{m_i} + \boldsymbol{P}_{s_i} + \boldsymbol{P}_{0_i} - \boldsymbol{P}_{ov_i} \qquad i = 1, \ldots, N \tag{5.5}$$

where

$$\boldsymbol{P}_{m_i} = \frac{d_{on}p_{on} + d_{am}p_{am} + \boldsymbol{d}_{m_i}p_m}{T_m} \qquad m = 1, \ldots, M \tag{5.6}$$

is the average power consumption of the $m-$th monitoring task,

$$\boldsymbol{P}_{s_i} = \frac{d_{on}p_{on} + \boldsymbol{d}_{s_i}p_s}{T_s} \tag{5.7}$$

is the average power dissipated by the synchronization task,

$$\boldsymbol{P}_{ov_i} = \boldsymbol{k}_i \frac{d_{on}p_{on}}{T} \tag{5.8}$$

represents the random amount of power which is saved if $\boldsymbol{k}_i$ task overlaps occur during T (so that the radio does not need to be turned on) and

$$\boldsymbol{P}_{0_i} = p_0 \left(1 - \sum_{m=1}^{M} \frac{d_{on} + d_{am} + \boldsymbol{d}_{m_i}}{T_m} - \frac{d_{on} + \boldsymbol{d}_{s_i}}{T_s} + \boldsymbol{k}_i \frac{d_{on}}{T} \right) \tag{5.9}$$

is the base power consumption when a node is idle and the radio module is off. Usually, the overlap term (5.8) is negligible, because $d_{on} << \boldsymbol{d}_{s_i}$ and $d_{on} << \boldsymbol{d}_{m_i} + d_{am}$ for $m = 1, \ldots, M$. Furthermore, the overlap probability is also negligible if the task duty cycles are much smaller than 1.

Observe that the total average power consumption increases linearly with $\boldsymbol{d}_{m_i}$ and $\boldsymbol{d}_{s_i}$, whereas it tends to decrease slightly when the number of overlaps $\boldsymbol{k}_i$ grows. As a consequence, if we apply (5.1) and (5.3) to (5.5) and we assume that no overlap occurs (i.e., $\boldsymbol{k}_i = 0$), we have that the upper

bound to the average power dissipated by node i over T is

$$\overline{P}_i = p_0 + \sum_{m=1}^{M} \frac{d_{on}(p_{on}-p_0) + d_{am}(p_{am}-p_0) + (\alpha_i \varepsilon_{max} + \hat{d}_{m_i})(p_m - p_0)}{T_m} + \frac{d_{on}(p_{on}-p_0) + (\beta_i \varepsilon_{max} + \hat{d}_s)(p_s - p_0)}{T_s} \qquad i = 1, \ldots, N \qquad (5.10)$$

To a first approximation, $\varepsilon_{max} = |\nu_{max}| \cdot T_s$ and ν_{max} is the worst-case clock drift rate of the WSN. After replacing ε_{max} with $|\nu_{max}| \cdot T_s$ in (5.10) and analysis of the derivative of $\overline{P}_i$ with respect to T_s, it can be easily proved that (5.10) is minimum for

$$T^*_{s_i} = \sqrt{\frac{\hat{d}_s(p_s - p_0) + d_{on}(p_{on} - p_0)}{\alpha_i |\nu_{max}| \sum_{m=1}^{M} \frac{(p_m - p_0)}{T_m}}} \qquad (5.11)$$

This expression returns the time interval between subsequent synchronizations for which, in the worst-case, the energy dissipation of node i is minimum. Of course, the expression (5.11) does not assure to minimize the power consumption in all possible operating conditions because the optimal synchronization interval is a random variable that depends also on the number of overlapping tasks as well as on the actual duration of the synchronization procedure. Besides, this expression is not valid for the leaf nodes (i.e. when $\alpha_i = 0$), because according to (5.10) their average power consumption does not correlate with ε_{max} when they neither receive sensor data from any other nodes nor have to delay packet transmissions. Moreover, the power dissipated by the leaf nodes of the network is certainly smaller than the power dissipated by the other devices (especially the root which must gather all sensor data). However, given that in practice all WSN nodes must use a common value of T_s, setting the synchronization period equal to (5.11) for $\alpha_i = 4$ provides a simple and sensible criterion to reduce the average energy dissipation of the network. In fact, (5.11) can be easily computed and applied in real-time and it is very general because

it is independent of the duration of the monitoring tasks. Furthermore, even if (5.11) tends to overestimate the optimal synchronization period, the actual amount of dissipated power tends to increase very little around the minimum, as it will be shown in Section 5.4.

5.4 Analysis through simulations

In order to validate the analysis described in Section 5.3, in the following the results of some simulations are reported. The simulator developed in C to this purpose is able to generate power consumption patterns of different length and amplitude depending on the hierarchical position of one node within the network topology. All task operations (including early wake–up and late switch–off times) are scheduled by a digital timer clocked by a virtual 32.768 kHz XO. The simulator includes the possibility to generate task overlaps. Tasking is managed by a typical non-preemptive first-in-first-out scheduler, like in TinyOS. All the parameters of the model, including the drift rate, the oscillator jitter and the number of tasks, can be arbitrarily changed by the user. However, in order to enable an easier comparison of the simulation results in different situations, most parameters of the model were fixed. For the same reason, the parameter values of multiple monitoring tasks were also assumed to be equal. Most of the time and power consumption coefficients used for simulations refer to the same TelosB nodes used for the experiments described in Section 5.5. The values of such parameters were measured by means of a digital oscilloscope Agilent DSO7032A and are listed in the Table 5.1.

The coefficients $\boldsymbol{d}_{m_i}$ and $\boldsymbol{d}_{s_i}$ have been simulated on the basis of the model described in Section 3.2, assuming that no acknowledgment packets are sent by any node in the network. The simulation results described below refer to four different scenarios. In each of them, only one of the

Table 5.1: Parameter values used in simulations for $m = 1,...,M$.

Power coefficients				
p_{on}	p_{am}	p_m	p_s	p_0
3 mW	1 mW	56 mW	58 mW	3 μW
Time coefficients				
d_{on}	d_{am}	d_m and d_s		
3 ms	72 ms	simulated using the model described in Section 3.2		

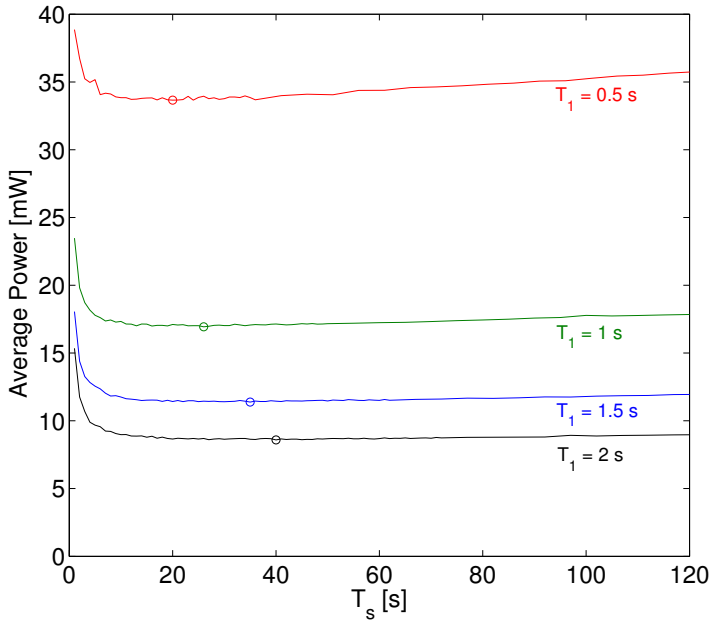


Figure 5.4: Average power, dissipated by the central node of a 10-node WSN with a star topology, as a function of T_s for different periods of a single monitoring task, $|\nu_{max}| = 50$ ppm.

parameters of the model was varied in order to evaluate its influence on the energy dissipation of a single network node. All simulations were repeated for T_s ranging from 1 s to 120 s. The corresponding average power curves are plotted in Fig. 5.4–5.7 as a function of T_s. Each point represents the average power dissipated by one node with a given position in a different WSN

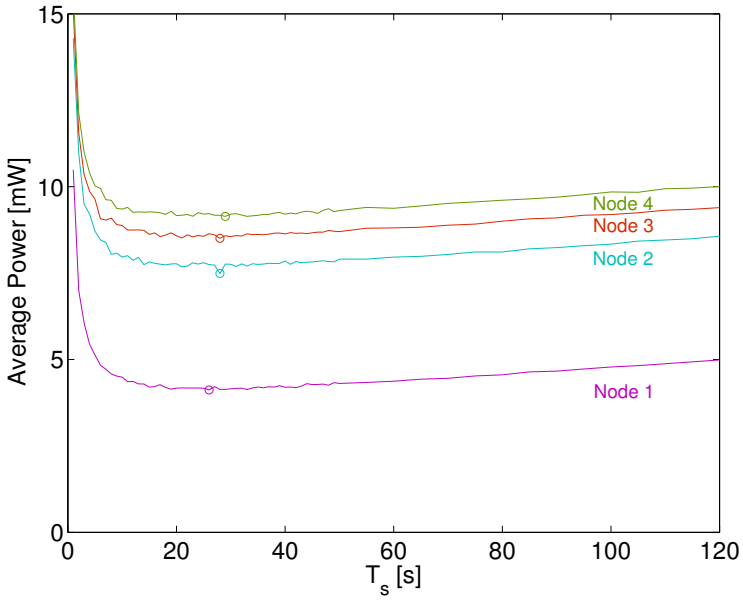


Figure 5.5: Average power, dissipated by to 4 daisy-chained devices running a single monitoring task, as a function of T_s, $|\nu_{max}| = 50$ ppm.

after 30 synchronization periods. In fact, with records of such a length, the simulation time remains within reasonable limits and the width of the 95% confidence intervals containing the average power is generally smaller than 5% of the estimated values. Fig. 5.4 reports the power dissipated by the central device of a 10–node star WSN with $|\nu_{max}| = 50$ ppm for different periods of a single monitoring tasks (i.e., $T_1 = 0.5,\ 1,\ 1.5,\ 2$ s). Fig. 5.5 shows the power consumption patterns of $N = 4$ daisy–chained devices executing just one monitoring task of period $T_1 = 1$ s, with $|\nu_{max}| = 50$ ppm. Fig. 5.6 refers to the root node of a binary tree consisting of $N = 7$ nodes. Also in this case, all nodes run a single monitoring task of period $T_1 = 1$ s, but the effect of different worst–case drift rates is analyzed (i.e., $|\nu_{max}| = 25,\ 50,\ 75,\ 100$ ppm). Finally, in Fig. 5.7 the average power consumption curves of a node running from 1 to 4 identical monitoring tasks of period 2 s are plotted. In this example, the WSN consists of $N = 5$ devices and

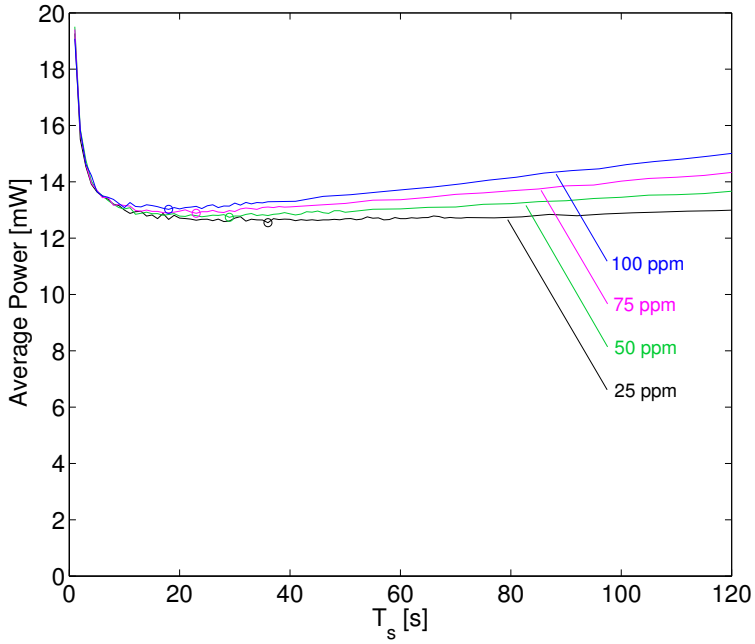


Figure 5.6: Average power, dissipated by the root of 7-node WSN running a single monitoring task, as a function of T_s for different maximum clock drift rates.

$|\nu_{max}| = 50$ ppm. In all cases a global minimum exists. This is highlighted by a small circle. Observe that the power consumption strongly depends on the position of the node in the WSN, as well as on the number and the period of the monitoring tasks. However, the corresponding optimal synchronization periods are mostly independent of the hierarchy. In fact, the minima of the curves in Fig. 5.5 are almost vertically aligned. This is in accordance with the theoretical analysis reported in Section 5.3. In Fig. 5.4 the values of the optimal synchronization period range from 20 s when $T_1 = 0.5$ s to 40 s when $T_1 = 2$ s, i.e. it doubles as T_1 increases by 4 times, in accordance with (5.11). Similarly, in Fig. 5.6 the optimal synchronization period grows from 18 s for $|\nu_{max}| = 100$ ppm up to 36 s for $|\nu_{max}| = 25$ ppm, as expected. This is reasonable, because when $|\nu_{max}|$ is small, repeating frequent synchronizations is unnecessary. Notice that $|\nu_{max}|$ affects the power growth rate, but it does not alter significantly the

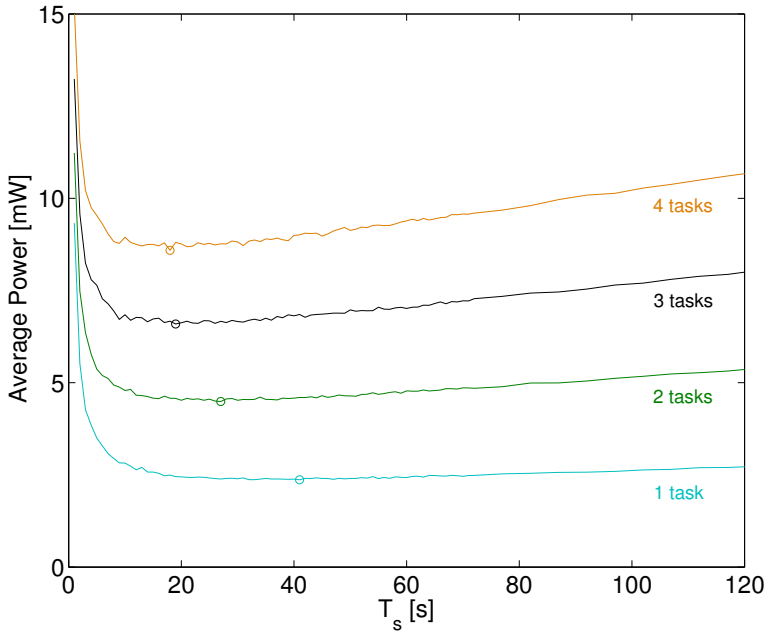


Figure 5.7: Average power consumption of one WSN node when multiple identical monitoring tasks are running simultaneously. The power is shown as a function of T_s, $|\nu_{max}| = 50$ ppm.

amount of dissipated power, because the intervals $|\varepsilon_{max}|$ are usually much shorter than $\boldsymbol{d}_{m_i}$ and $\boldsymbol{d}_s$.

On the other hand, the power consumption may increase considerably when the number of tasks grows. In the case of multiple identical tasks shown in Fig. 5.7 two different situations occur. If the node duty cycle is very small, the optimal synchronization interval tends to decrease as the square root of the number of tasks M, in accordance with (5.11). Conversely, when the duty cycle becomes large enough (e.g., for $M = 4$) the power consumption grows less than expected due to the increasing number of task overlaps. For the same reason, also the optimal synchronization interval decreases more slowly.

As a last remark, notice that, even if expression (5.11) usually overestimates the optimal synchronization interval, the consequences in terms

on dissipated power are minor because the consumption increases quite smoothly as T_s grows.

5.5 Experiment Description

5.5.1 Implementation details

In order to prove the correctness of the analysis described in Section 5.2, several experiments were performed using a small WSN consisting of 10 TelosB nodes. All nodes were programmed to run a single monitoring task and a synchronization procedure. During the monitoring task, each node reads periodically the value measured by a humidity sensor and broadcasts the collected sample to all the other devices within a time window d_1 which is proportional to the number of WSN nodes. Data transmission is performed orderly by each node depending on its identification number (ID). In this way, the probability of packet collisions is reduced. The synchronization period of the monitoring task T_1 can be set by the user.

The synchronization task was conceived not only to correct the clocks of the WSN nodes, but also to estimate the largest drift rate between the SM acting as a time reference and the other devices of the network. We decided to rely on a simplified one–hop synchronization scheme, rather than using a complete synchronization protocol, because the implementation details of many existing solutions are excessively complicated and redundant for validating the model proposed in this chapter. Also, not all synchronization protocols enable accurate clock drift estimation, which is an essential part of the planned experimental activity. The proposed synchronization procedure can operate in two alternative modes, i.e. by setting the synchronization interval T_s equal to a given constant value (*fixed interval mode*), or by adjusting the synchronization intervals in real–time according to (5.11) anytime a new value of $|\nu_{max}|$ is estimated (*automatic interval mode*). In

both cases, the basic steps of the procedure are the same and are shortly described in the following.

- When the ith synchronization period expires, the SM broadcasts a *synchronization packet* (SP). This packet is time–stamped at the MAC layer in order to reduce the effect of the sending and channel access times.

- All the other nodes time–stamp the incoming SP at the MAC layer and then use the received SP time–stamp to compensate their time offsets with respect to the SM.

- Afterwards, all nodes send an *acknowledgement packet* (AP) to the SM. The AP contains the ID of each node as well as the time–stamp value of the formerly received SP. Acknowledgment packet transmission is performed orderly by each node depending on its ID within a time window d_s in order to reduce the probability of channel contention.

- The SM estimates the relative clock drift rates of the WSN nodes (as it will be explained in the following) and chooses that with the maximum absolute value. Eventually, $|\nu_{max}|$ is used by SM to compute T_s^* (in *the automatic interval mode* only) and $\varepsilon_{max} = |\nu_{max}| \cdot T_s$, which are sent to the network nodes in the next SP. The values of ε_{max} are used by all nodes to schedule safely the synchronization and monitoring tasks according to the model described in Section 5.2.

The relative clock drifts of the WSN nodes with respect to the SM are estimated in two steps. At first, each drift rate is measured from two consecutive sets of time–stamp values on the basis of the following expression:

$$\nu_{n_i} = \frac{(t_{n_i} - t_{n_{i-1}}) - (t_{SM_i} - t_{SM_{i-1}})}{t_{SM_i} - t_{SM_{i-1}}} \qquad i \geq 2 \tag{5.12}$$

where t_{SM_i} and $t_{SM_{i-1}}$ are the sending time–stamps associated with the ith and $(i-1)$th synchronization packets, respectively, and t_{n_i} and $t_{n_{i-1}}$ are the corresponding receiving time–stamps collected by the nth node (with $n \neq SM$). Afterwards, in order to reduce the effect of possible random uncertainty contributions affecting packet time–stamping, the SM computes the moving average of the drift rate of each node over the most recent K samples, i.e. $\overline{\nu}_{n_i} = \frac{1}{K}\sum_{k=1}^{K} \nu_{n_{i-k}}$, with $K \leq K_{max}$. Eventually, the absolute value of the maximum relative drift rate at the end of the ith synchronization interval is $|\nu_{max}| = \max_{\substack{n=1,\ldots,N \\ n \neq SM}} |\overline{\nu}_{n_i}|$.

In the current implementation, the value of K_{max} is set equal to 16. Indeed, divisions by a power of two are particularly efficient from the computational point of view. Also, larger values of K_{max} should be avoided, because they could rapidly lead to the memory overflow of the SM when the number of nodes grows. Observe that the first drift rate estimates are obtained for $i = 2$, namely after the second synchronization. Thus, the initial values ν_{n_1} must be properly chosen. As the relative frequency offset of low–cost crystal oscillators is in the order of some tens of ppm, we decided to set $\nu_{n_1} = 50$ ppm on all nodes.

5.5.2 Experimental results

The average power consumption of one WSN node running the two tasks described in the previous subsection was measured in different operating conditions using an automatic procedure similar to that described in [39]. Basically, a Labview application was implemented to collect simultaneously the voltage values applied to a WSN node and the corresponding current drain. In fact, since the voltage provided by a DC source is approximately constant, each average power consumption value results from the product of the average current drain and the average supply voltage, i.e. $\overline{P} = \overline{IV}$.

Average voltage and current levels were measured with two digital multimeters (DMMs) Agilent 34411A connected to a PC (as in the experimental setup shown in Fig. 4.5). The integration time of both DMMs was set to 100 power line cycles (i.e., 2 s) in order to exploit the best instrument resolution and to maximize normal mode noise rejection. Average power measurements were repeated for various synchronization periods T_s in the range [1 s, 400 s] and for three different periods of the humidity monitoring task, i.e. $T_1 = 0.5$ s, 0.7 s and 0.9 s. In all cases the *fixed interval mode* of the synchronization procedure was selected. The collected experimental results are plotted in Fig. 5.8 (solid line). The standard measurement uncertainty associated with each value of $\overline{P}$ can be estimated using the same procedure described in [39] and is given by

$$u(\overline{P}) = \sqrt{\overline{I}^2 u^2(\overline{V}) + \overline{V}^2 u^2(\overline{I})} \tag{5.13}$$

where $u^2(\overline{V})$ and $u^2(\overline{I})$ are the standard uncertainties related to average current and voltage measurements, respectively. Both $u^2(\overline{V})$ and $u^2(\overline{I})$ result from the combination of independent random and instrument–related contributions. The effect of the former ones can be estimated by computing the experimental standard deviation of $\overline{I}$ and $\overline{V}$ (*type A* standard uncertainty) [1]. Conversely, the instrument–related standard uncertainties can be determined from instrument specifications (*type B* evaluation procedure). In the considered case the maximum standard uncertainty associated with average power values is about 2.5 mW, i.e. about one order of magnitude smaller than measurement results.

In order to check the validity of the theoretical model, the collected experimental results were compared with the average power consumption curves computed by (5.10). The maximum drift rate of the WSN clocks was estimated directly by the SM through the procedure described in Subsection 5.5.1. In particular, after a quite large number of repeated syn-

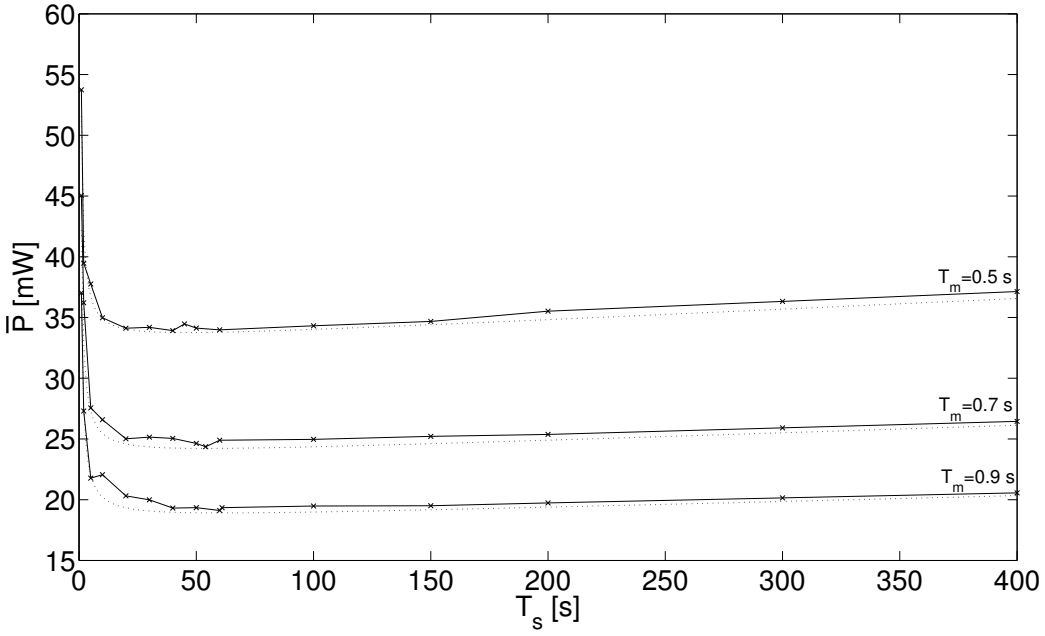


Figure 5.8: Average power consumption of a TelosB node as a function of the synchronization interval T_s for three different periods of the humidity monitoring tasks (i.e., for $T_1 = 0.5$ s, 0.7 s, 0.9 s). The estimated maximum relative clock drift rate of 9 nodes with respect to the SM is $|\nu_{max}| \approx 20$ ppm. In the picture the solid lines refer to the experimental results, whereas the dashed lines are obtained from (5.10).

chronizations, we obtained that $|\nu_{max}| \approx 20$ ppm. The theoretical average power consumption curves as a function of T_s for $T_1 = 0.5$ s, 0.7 s and 0.9 s are shown in Fig. 5.8 (dotted lines). Observe that such curves are in good accordance with the experimental patterns, thus confirming the validity of the proposed analysis. The corresponding optimal synchronization periods are $T_s^* \approx 45$ s for $T_1 = 0.5$ s , $T_s^* \approx 54$ s for $T_1 = 0.7$ s and $T_s^* \approx 61$ s for $T_1 = 0.9$ s. Unfortunately, such values are not very visible in Fig. 5.8. Indeed, due to the small value of $|\nu_{max}|$, actual power consumption tends to increase very slowly.

In principle, the optimal synchronization intervals are not constant over time, because they are very sensitive to possible variations of $|\nu_{max}|$ (e.g., due to temperature fluctuations). This is the reason why it would be

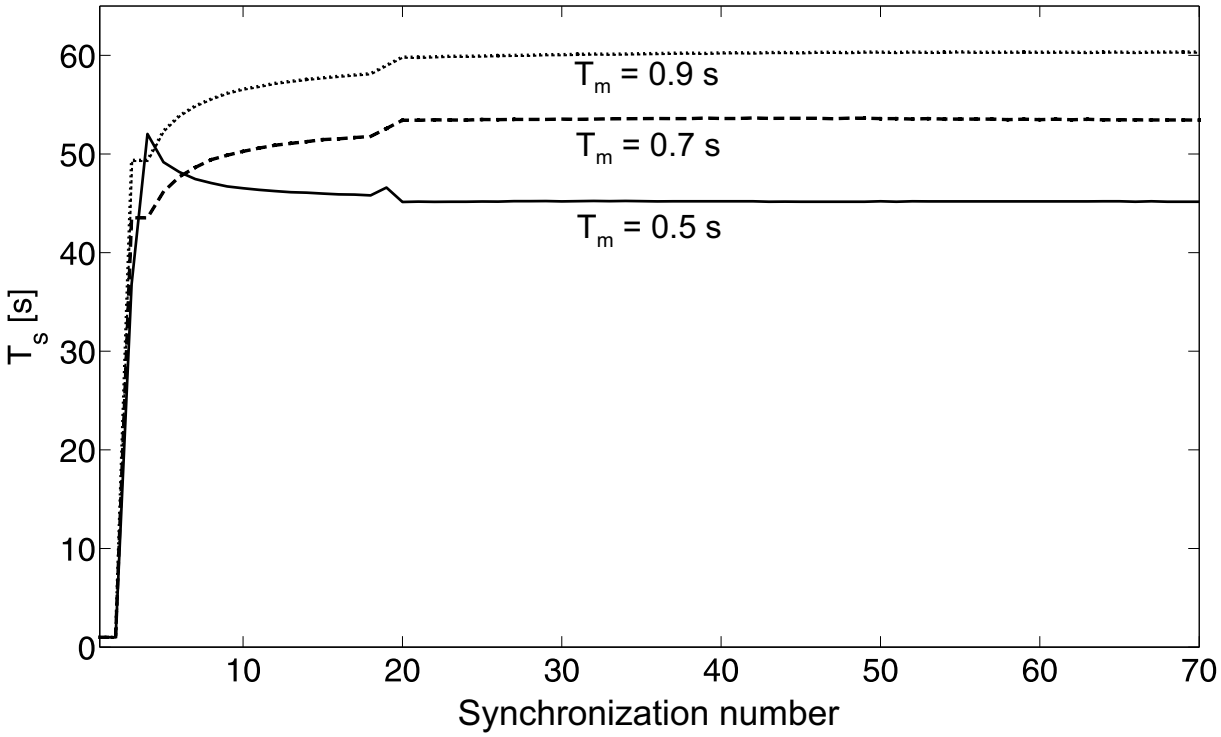


Figure 5.9: Sequence of synchronization periods estimated automatically by the SM for three periods of the humidity monitoring task, i.e. $T_1 = 0.5$ s (solid line), $T_1 = 0.7$ s (dashed line) and $T_1 = 0.9$ s (dotted line). In all cases, the initial value of T_s is 1 s and it is computed after each synchronization after estimating ν_{max}.

preferable if WSN nodes were able to adjust T_s^* in real–time. When the synchronization procedure described in Subsection 5.5.1 operates in the *automatic interval mode*, the SM changes dynamically the synchronization periods on the basis of the last estimated maximum drift rate. In Fig. 5.9, the duration of the synchronization periods computed by each WSN node for $T_1 = 0.5$ s (solid line), $T_1 = 0.7$ s (dashed line) and $T_1 = 0.9$ s (dotted line) are plotted as a function of the number of synchronization events. The initial period is set as small as possible (i.e. equal to 1 s) because when the WSN nodes are switched on, they are completely unsynchronized. Notice that after a short transient phase, the estimated synchronization periods converge to the respective optimal values. Consequently, all nodes reach

the configuration characterized by the least energy consumption.

5.6 Summary

When a WSN is used in periodic monitoring applications characterized by a small duty cycle, a quite obvious strategy for saving power is to switch off the radio modules during idle time intervals. However, this approach may cause major connection problems if nodes are not properly synchronized. Indeed, some nodes might try to communicate when others are still sleeping. In this chapter we proved that the overall average power dissipated by a node running multiple periodic monitoring tasks and a suitable synchronization protocol can be minimized, without affecting inter–node communication reliability. From an in–depth analysis of the problem mentioned above, a simple analytical expression [i.e., equation (5.11)] returning the value of the optimal synchronization period has been obtained. Multiple simulation and experimental results confirm the correctness of the proposed model. Even if (5.11) depends on several quantities, most of them are approximately constant and can be measured quite easily or derived from node specifications. The most critical parameter for the estimation of the optimal synchronization interval is the absolute value of the maximum relative drift rate of WSN clocks. However, if this term is estimated by the same procedure performing clock synchronization, all WSN nodes can adjust autonomously the optimal synchronization period, thus steadily working in the condition where the average power consumption of the network is minimum.

Chapter 6

Conclusion

6.1 Thesis Summary

In this dissertation we have investigated the trade-off between the need to perform time synchronization and the need to save energy in wireless sensor networks. First, we have estimated quantitatively the impact of different factors on the accuracy of time synchronization. Also, we have studied how the time synchronization overhead can be minimized, while keeping uncertainty within required limits under changing environmental conditions. Then we have studied the case when the time synchronization is critical to power saving in WSN. On the basis of that study, we have proposed a strategy to achieve a minimum energy consumption during synchronous operation of WSN nodes. We have proved the effectiveness of our approach through multiple simulations, measurements and experiments with common wireless sensors.

6.1.1 Quantitative Analysis of Time Uncertainty

The oscillator frequency deviations and communication delays are the most significant factors affecting the accuracy of time estimation in WSN. The frequency deviation, which causes the growth of time uncertainty, can be

considered approximately constant during typical WSN operation intervals and under stable ambient conditions. We have quantified the time uncertainty by measuring the clock drifts of widely used wireless sensors with respect to a reference clock. The measurement results confirm that the offsets between WSN clocks change almost linearly with time. The rate of that change, which is equal to the slope of the corresponding linear function, can be computed and used to estimate the clock offset at any moment of time. However, the real environmental conditions may rapidly change in an unpredictable way, causing the change in oscillator frequency. In fact, we have read the timers of WSN nodes when air temperature varied both fast and slowly with time. The measurements confirm that the clock drift changes considerably with temperature. As a consequence, an accurate and energy-consuming drift rate estimation (e.g. using linear regression) may become ineffective in WSN.

Communication delays cause the aging of time information delivered to wireless sensors. Indeed, if a sending node inserts its clock reading into an outgoing packet, this reading arrives at a receiving node only after some delay. Therefore, if the receiver time-stamps the packet upon reception and computes the offset between its clock and the sender clock by subtraction of one time-stamp from another, there is a significant uncertainty in that computation. Although MAC-layer time-stamping can reduce the time uncertainty to negligible values compared to clock resolution, this time-stamping method depends greatly on the specific features of WSN hardware and MAC-sublayer, and therefore may not be always feasible. Consequently, in many cases the estimation of communication delays is necessary. We have estimated the amount of time uncertainty introduced by the communication of common wireless sensors in various realistic situations. In particular, we have measured the communication latencies and their considerable change under different packet lengths, traffic conditions

and number of hops in a multi-hop network.

6.1.2 Adaptive Time Synchronization Algorithm for WSN

We have investigated how time synchronization parameters can be adjusted in response to changing environmental conditions during WSN operation. In particular, we have developed an adaptive algorithm which performs periodic synchronization of wireless sensors to the time of a special node dynamically reelected as a synchronization master. In order to save energy, the synchronization period is extended when a required percentage of WSN nodes are synchronized with a target accuracy. If the number of synchronized nodes is below the threshold, the synchronization period is shortened to maintain the target synchronization accuracy. The synchronization period is adjusted in accordance with so called *Simple Response Method*, which was also used in the past to calibrate primary time references. The experimental results have shown that the achieved synchronization period converges to the optimal value, also when the ambient temperature changes.

6.1.3 Energy Saving Strategy for Synchronous Wakeup of WSN

We have studied the relationship between time uncertainty and power consumption in WSN employing synchronous sleep and wake-up schemes. We have shown that, although generally the power consumption of WSN can be reduced at the expense of an increased time uncertainty, in many cases the higher uncertainty may cause additional energy waste. In fact, a WSN node has to prolong the period when its radio module is active to be able to receive messages from the other nodes with faster or slower clocks. We have derived an expression for the optimal synchronization period, resulting in minimum power consumption of wireless sensor nodes and timely

activation of their radio modules. Simulations have verified the validity of that expression for WSN with various topologies and running several tasks, each of which requires synchronous wake-up. Also, we have measured the power consumption of WSN nodes. In fact, the WSN nodes run a periodic synchronization procedure, and their average power consumption has been estimated under different values of the corresponding synchronization period. Besides, we have implemented a power efficient time synchronization algorithm, which dynamically computes the optimal synchronization interval. The experimental results have shown that the computed value of the optimal synchronization period converges to the theoretical one.

6.2 Applicability

The results of this work can be used in agricultural, commercial and industrial sensor networks collecting various environmental parameters during relatively long terms. For example, the designed synchronization algorithms can support the operation of wireless sensors measuring temperature and humidity in vineyards, granaries, warehouses with hygroscopic materials, vine cellars, diaries and tea-packing factories. In particular, in order to time-stamp measurements in the applications listed above, the developed adaptive algorithm can maintain a required accuracy of time synchronization without an excessive overhead under changing ambient conditions (e.g. due to atmospheric effects or some manufacturing processes). Also, the proposed synchronous wakeup strategy can be used to activate environmental sensors and radio modules in time to avoid energy waste during long idle time intervals. As a result, both algorithms can achieve significant energy savings, thus prolonging the WSN lifetime. In general, the approach to energy-efficient time synchronization described in this thesis provides the basis for the development of other synchronization

schemes, which can be used under extreme conditions. For example, the wireless sensors monitoring radioactivity in a contaminated area will need more reliable synchronization, considering the impact of ionizing radiation on the stability of their oscillator frequencies. Besides, the high accuracy of synchronization is critical in such cases, because the consequences of differences in clock readings may be disastrous (e.g. if the direction or speed of movement of a radioactive material is estimated erroneously). At the same time, such sensors have to serve as long as possible under resource constraints and without maintenance. Although this problem has to be addressed considering many more factors, in order to solve it someone can start from the ideas proposed in this thesis. Finally, the experiments performed during this work provide interesting quantitative data, which can be used to develop and compare other synchronization techniques.

6.3 Future Work

We plan to investigate further the relationship of time synchronization and various performance characteristics of wireless sensor networks. Specifically, it is interesting to study the impact of the accurate clock drift rate estimation on the energy consumption of WSN. Also, we intend to explore temperature-aware time synchronization algorithms, which use temperature readings either to recompute the clock drift rate or to initiate a new inter-node synchronization procedure. Besides, the time synchronization methods could be studied for the cases when sensor networks are deployed in harsh environments and withstand unpredictable faults of nodes. Another planned research direction is the development of new wireless sensor nodes with specific hardware features supporting the time synchronization in view of minimizing the power consumption according to the proposed

methods.

Bibliography

[1] *ISO/IEC Guide 98:1995, Guide to the Expression of Uncertainty in Measurement.* Geneva, Switzerland, 1995.

[2] *Fundamentals of Quartz Oscillators, Application Note 200-2.* Hewlett Packard Co., May 1997.

[3] *IEC 61588:2004 - IEEE 1588:2002,Precision clock synchronization protocol for networked measurement and control systems.* Geneva, Switzerland, Sep. 2004.

[4] *IEEE 802.15.4:2006, Standard for Information Technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - specific requirement. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs).* New York, USA, Sep. 2006.

[5] Agilent Technologies, Inc. *Agilent 33220A 20 MHz Function/Arbitrary Waveform Generator, Datasheet.*

[6] R. Akl and Y. Saravanos. Hybrid energy-aware synchronization algorithm in wireless sensor networks. In *Proceedings of the 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'07)*, Sep. 2007.

[7] C. Alippi, G. Anastasi, M. Di Franceso, and M. Roveri. Energy management in wireless sensor networks with energy hungry sensors. *IEEE Instrumentation and Measurement Magazine*, 12(2):16–23, Apr. 2009.

[8] D. W. Allan. Time and frequency (time-domain) characterization, estimation, and prediction of precision clocks and oscillators. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, UFFC-34(6):647–654, Nov. 1987.

[9] V.G. Androsova, E.G.Bronnikova, A.M. Vasilyev, et al. *Piezoelectric resonators: The handbook*. Radio and communication, Moscow, Russia, 1992. (in Russian).

[10] B. Bougard, F. Katthoor, D. C. Daly, A. Chandrakasan, and W. Dehaene. Energy efficiency of the IEEE 802.15.4 standard in dense wireless microsensor networks: Modeling and improvement perspectives. In *Proc. Design, Automation, and Test in Europe (DATE)*, pages 196–201, Munich, Germany, Mar. 2005.

[11] M. Bradonjić, E. Kohler, and R. Ostrovsky. Near-optimal radio use for wireless network synchronization. `http://arxiv.org/abs/0810.1756`http://arxiv.org/abs/0810.1756, 2008.

[12] S. Bregni. Clock stability characterization and measurement in telecommunications. *IEEE Transactions on Instrumentation and Measurement*, 46(6):1284–1294, Dec. 1997.

[13] C. M. Chao and Y. C. Chang. A power-efficient timing synchronization protocol for wireless sensor networks. *Proc. Journal of Information Science and Engineering*, pages 985–997.

[14] C.-Y. Chong and S. P. Kumar. Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, August 2003.

[15] H. Dai and R. Han. Tsync: A lightweight bidirectional time synchronization service for wireless sensor networks. *ACM SIGMOBILE Mobile Computing and Communication Review*, pages 125–139.

[16] J. Elson and D. Estrin. Time synchronization for wireless sensor networks. In *Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS-01)*, Apr 23-27 2001.

[17] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proc. Symposium on Operating Systems Design and Implementation (OSDI)*, volume 36, pages 147–163, Boston, MA, USA, Dec. 2002.

[18] P. Ferrari, A. Flammini, D. Marioli, E. Sisinni, and A. Taroni. Non invasive time synchronization for ZigBee wireless sensor networks. In *Proc. International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS)*, pages 121–126, Ann Arbor, MI, USA.

[19] A. Flammini, D. Marioli, E. Sisinni, and A. Taroni. A real-time wireless sensor network for temperature monitoring. In *Proc. International Symposium on Industrial Electronics (ISIE)*, pages 1916–1920, Vigo, Spain, June 2007.

[20] Freescale Semiconductor. *MC13192 2.4 GHz Low Power Transceiver for the IEEE 802.15.4 Standard*, Apr. 2008.

[21] S. Ganeriwal, D. Ganesan, H. Shim, V. Tsiatsis, and M. B. Srivastava. Estimating clock uncertainty for efficient duty–cycling in sensor

networks. In *Proc. International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 130–141, San Diego, CA, USA, Nov. 2005.

[22] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. In *Proc. International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 138–149, Los Angeles, CA, USA, Nov. 2003.

[23] G.M. Garner, F. Feng, K. den Hollander, J. Hongkyu, K. Byungsuk, L. Byoung-Joon, J. Tae-Chul, and J.J. Joung. IEEE 802.1 AVB and its application in carrier-grade ethernet. *IEEE Communications Magazine*, 45(12):126–134, Dec. 2007.

[24] L. Girod and D. Estrin. Robust range estimation using acoustic and multimodal sensing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001)*, Mar. 2001.

[25] A. Giusti, A. L. Murphy, and G. P. Picco. Decentralized scattering of wake-up times in wireless sensor networks. In *Proceedings of EWSN 2007*, pages 245–260, Berlin Heidelberg, 2007. Springer-Verlag.

[26] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. In *Proc. Int. Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 93–104, Cambridge, MA, USA, Nov. 2000.

[27] J. Hsu, S. Zahedi, A. Kansal, M. Srivastava, and V. Raghunathan. Adaptive duty cycling for energy harvesting systems. In *Proc. International Symposium on Low Power Electronics and Design (ISPLED)*, pages 180–185, Tegernsee, Germany, Oct. 2006.

[28] IEEE Standards Coordinating Committee 27. *IEEE Standard Definitions of Physical Quantities for Fundamental Frequency and Time MetrologyRandom Instabilities, IEEE Std 1139-2008.*

[29] Isotemp Research Inc., Charlottesville, VA 22903, USA. *Crystal Oscillator Aging, Document 146-000.*

[30] V. R. Jain, R. B., A. Kumar, and P. Ranjan. wildCENSE: GPS based animal tracking system. In *ISSNIP 2008*, pages 617–622, 2008.

[31] H. W. Johnson and M. Graham. *High-Speed Digital Design. A Handbook of Black Magic.* Prentice Hall PTR, Englewood Cliffs, New Jersey 07632, USA.

[32] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *Proceedings of IPSN'07*, pages 254–263, Cambridge, MA, USA, Apr. 2007.

[33] L. Kleinrock and F. A. Tobagi. Packet switching in radio channels: Part I-carrier sense multiple-access modes and their throughput-delay characteristics. *IEEE Trans. on Communications*, pages 1400–1416.

[34] H. Kopetz and W. Ochsenreiter. Clock synchronization in distributed real-time systems. *IEEE Trans. on Computers*, 36(8):933–939, August 1987.

[35] B. Kusý, P. Dutta, P. Levis, M. Maróti, A. Lédeczi, and D. Culer. Elapsed time on arrival: a simple and versatile primitive for canonical time synchronization services. *International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC)*, 2(1), 2006.

[36] E.-Y. A. Lin, J. Rabaey, and A. Wolisz. Power–efficient rendez–vous schemes for dense wireless sensor networks. In *Proc. IEEE Interna-*

tional Conference on Communications (ICC), volume 7, pages 3769–3776.

[37] M. Lukac, P. Davis, R. Clayton, and D. Estrin. Recovering temporal integrity with data driven time synchronization. In *Proceedings of the 8th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN09)*, Apr. 2009.

[38] L. Ma, H. Zhu, G. Nallamothu, and Z. Zhang B Ryu. Impact of linear regression on time synchronization accuracy and energy consumption for wireless sensor networks. In *Proc. IEEE MILCOM 2008*, San Diego, CA, USA, 2008.

[39] D. Macii and D. Petri. An effective power consumption measurement procedure for bluetooth wireless modules. *IEEE Transactions on Instrumentation and Measurement*, 56(4):1355–1364, Aug. 2007.

[40] D. Macii, P. Tavella, E. Perone, P. Carbone, and D. Petri. Accuracy comparison between techniques for the establishment of calibration intervals: Application to atomic clocks. *IEEE Transactions on Instrumentation and Measurement*, 53(4):1167–1172, Aug. 2004.

[41] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications (WSNA02)*, Atlanta, Georgia, USA, Sep. 2002.

[42] M. Maróti, B. Kusý, G. Simon, and A. Lédeczi. The flooding time synchronization protocol. In *Proc. International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 39–49, Baltimore, MD, USA, Nov. 2004.

[43] A. Milenkovic, C. Otto, and E. Jovanov. Wireless sensor networks for personal health monitoring: Issues and an implementation. *Computer Communications*, 29:2521–2533, 2006.

[44] D. L. Mills. Precision synchronization of computer network clocks. *ACM Computer Communication Review*, 24(2):28–43, 1994.

[45] Moteiv. *Telos, Rev. B (Low Power Wireless Sensor Module)*, Dec. 2004.

[46] NCSL. *Establishment and Adjustment of Calibration Intervals, Recommended Practice RP-1*, Jan. 1996.

[47] B. Neubig. Comparison of passive and active aging of SC-cut and AT-cut crystals. In *Proc. European Frequency Time Forum*, pages 37–43, Mar. 1996.

[48] B. Neubig and W. Briese. *The Crystal Cookbook*. Franzis-Verlag, 1997. (in German).

[49] S. Palchaudhuri, A. K. Saha, and D. B. Johnson. Adaptive clock synchronization in sensor networks. In *Proc. International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 340–348, Apr. 2004.

[50] D. Raskovic, O. Lewis, and D. Giessel. Time synchronization for wireless sensor networks operating in extreme temperature conditions. In *Proc. 41st Southeastern Symposium on System Theory*, pages 24–28, Tullahoma, TN, USA, Mar. 2009. University of Tennessee Space Institute.

[51] K. Römer. Time synchronization in ad hoc networks. In *Proceedings of the International Symposium on Mobile Ad Hoc Networking and*

Computing (MobiHoc 01), pages 173–182, Long Beach, CA, USA, Oct. 2001.

[52] K. Römer, P. Blum, and L. Meier. Time synchronization and calibration in wireless sensor networks. In *Handbook of Sensor Networks: Algorithms and Architectures*, chapter 7. Wiley and Sons, Oct.

[53] T. Schmid, J. Friedman, and Z. Charbiwala. XCXO: An ultra-low cost ultra-high accuracy clock system for wireless sensor networks in harsh remote outdoor environments.

[54] C. Schurgers. *Wireless Sensor Networks and Applications*, chapter Wakeup Strategies in Wireless Sensor Networks, pages 195–217. Springer US, 2008.

[55] Sensirion AG, Staefa, ZH, Switzerland. *Datasheet SHT1x (SHT10, SHT11, SHT15). Humidity and Temperature Sensor*, Apr. 2009.

[56] K. Shahzad, A. Ali, and N. D. Gohar. ETSP: An energy–efficient time synchronization protocol for wireless sensor networks. In *Proc. International Conference on Advanced Information Networking and Applications (AINAW)*, pages 971–976, Mar. 2008.

[57] J.-P. Sheu, W.-K. Hu, and J.-C. Lin. Ratio-based time synchronization protocol in wireless sensor networks. *Telecommunication Sysems*, pages 25–35, Mar. 2008.

[58] M. L. Sichitiu and C. Veerarittiphan. Simple, accurate time synchronization for wireless sensor networks. In *Proc. Wireless Communications and Networking Conference (WCNC)*, volume 2, pages 1266–1273.

[59] G. Simon, M. Maróti, A. Lédeczi, G. Balogh, B. Kusý, A. Nádas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based counter-

sniper system. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 1–12, New York, NY, USA, 2004. ACM Press.

[60] K. Sohraby, D. Minoli, and T. Znati. *Wireless Sensor Networks: Technology, Protocols, and Applications*. John Wiley and Sons, May 2007.

[61] B. Sundararaman, U. Buy, and A. D. Kshemkalyani. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks, Elsevier*, 3(3):281–323, Mar. 2005.

[62] Tektronix. *TDS3000 Series Digital Phosphor Oscilloscopes 071-0382-01, Service Manual*.

[63] Texas Instruments. *2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*.

[64] Texas Instruments. *MSP430x15x, MSP430x16x, MSP430x161x Mixed Signal Microcontroller, SLAS386E*.

[65] Texas Instruments. *MSP430x1xx Family. Users Guide, SLAU094F*.

[66] Y. Uchimura, T. Nasu, and M. Takahashi. Time synchronized wireless sensor network for vibration measurement. In *Proc. SICE Annual Conference 2007*, pages 2940–2945. Kagawa University, Japan, Sept. 2007.

[67] S. Ullah, H. Higgins, M. A. Siddiqui, and K. S. Kwak. A study of implanted and wearable body sensor networks. In *Proceedings of KES-AMSTA 2008*, pages 464–473, Berlin Heidelberg, 2008. Springer-Verlag.

[68] US Army Communications-Electronics Command. *Performance Specification for Crystal Controlled Oscillators, MIL-PRF-55310E*, Mar. 2006.

[69] J. van Greunen and J. Rabaey. Lightweight time synchronization for sensor networks. In *Proc. ACM International Conference on Wireless Sensor Networks and Applications (WSNA)*, pages 11–19, San Diego, CA, USA, Sep. 2003.

[70] J. R. Vig. Quartz crystal resonators and oscillators for frequency control and timing applications - a tutorial. Technical Report SLCET-TR-88-1(Rev.8.0), U.S. Army Communications-Electronics Command (CECOM), Fort Monmouth, New Jersey 07703, USA, Sep. 1997.

[71] J. R. Vig and T. R. Meeker. The aging of bulk acoustic wave resonators, filters and oscillators. In *Proc. Forty-fifth Annual Symposium on Frequency Control*, pages 77–101, 1991.

[72] A. Wang, S-H. Cho, C.G. Sodini, and A.P. Chandrakasan. Energy-efficient modulation and MAC for asymmetric microsensor systems. In *Proc. International Symposium on Low-Power Electronics and Design (ISPLED)*, pages 106–111, Huntington Beach, CA, USA, Aug. 2001.

[73] G. Werner-Allen, K. Lorincz, M. Welsh, Omar M., J. Johnson, M. Ruiz, and J. Lees. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2):18–25, Mar. 2006.

[74] M. Xu, M. Zhao, and S. Li. Lightweight and energy efficient time synchronization for sensor network. In *International Conference on Wireless Communications, Networking and Mobile Computing*, pages 947–950, Sep. 2005.

[75] J. Yang, C. Zhang, X. Li, S. Fu Y. Huang, and M. Acevedo. An environmental monitoring system with integrated wired and wireless sensors. In *WASA 2008*, pages 224–236, Berlin Heidelberg, 2008. Springer-Verlag.

[76] Y. Yang and T. P. Yum. Delay distributions of slotted ALOHA and CSMA. *IEEE Trans. on Communications*, pages 1846–1857.

[77] Q. Ye, Y. Zhang, and L. Cheng. A study on the optimal time synchronization accuracy in wireless sensor networks. *Computer Networks*, pages 549–556, Jan. 2005.