FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 24, no.2, August 2011, 183-208

Clocks, Power and Synchronization in Duty-Cycled Wireless Sensor Nodes

Dedicated to Professor Slavoljub Aleksić on the occasion of his 60th birthday

Mile K. Stojčev, Ljubiša R. Golubović, and Tatjana R. Nikolić

Abstract: Recent advances in CMOS VLSI ICs and micro-electromechanical technology have led to development of small, low-cost, and low-power multifunctional sensors. Wireless sensor networks (WSNs) are large-scale networks of such sensors, dedicated to observing and monitoring various aspects of the physical world. Some intrinsic properties of WSNs including limited resource of energy, storage, computation, and bandwidth, make traditional synchronization methods unsuitable for WSNs. Time synchronization as an important issue consists of giving all sensor nodes (SNs) of the WSN a common time scale to operate. The common time scale is usually achieved by periodically synchronizing the clock of each SN to a reference source. In this manner the local time seen by each SN of the network is approximately the same, and time synchronization allows the entire system to cooperate. This paper gives a brief look to the time synchronization problem and the need for synchronization in WSNs. Then it points out that clock systems become a bottle-neck, after that it presents the available current clock technologies, next it examines the influence of these clock technologies, and finally provides guidelines for WSN developers who must choose among the different clock synchronization techniques.

Keywords: Wireless sensor network; clock synchronization; duty-cycled wireless sensor nodes.

Manuscript received on May 25, 2011.

M. Stojčev and T. Nikolić are with University of Niš, Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Nis, Serbia (e-mails: [mile.stojcev, tatjana.nikolic]@elfak.ni.ac.rs). Lj. Golubović is with University of Kragujevac, Technical Faculty Čačak, Svetog Save 65, 32000 Čačak, Serbia (e-mail: lj.r.golubovic@gmail.com).

1 Introduction

DURING the last two decades, tremendous technological advances have occurred in the development of low cost sensors, which are capable of wireless communication and data processing. Wireless sensor networks (WSNs) are distributed networks of such sensors, dedicated to closely observing real-word phenomena [1, 2].

One of the biggest challenges for designers of WSNs is to develop systems that will run unattended for years. The current generation of sensor nodes (SNs) is battery powered, so lifetime is a major constraint; future generations powered by ambient energy sources (sunlight, vibrations, etc), will provide very low currents, so energy consumption is heavily constrained [3].

In addition, efficient coordination of SNs requires the nodes to be synchronized, so the nodes can be turned off to save energy. The purpose of any time synchronization technique is to maintain a similar time within a certain tolerance throughout the lifetime of the network or among a specific set of nodes in the network. The problem consists of giving all the nodes of the network a common time scale to operate: 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 energy-efficient time synchronization solutions, different from the ones have been developed for wired networks. In this work we investigated the trade-offs between time synchronization accuracy and energy saving in WSNs. The common time scale is usually achieved by periodically synchronizing the clock at each node to a reference time source; therefore the local time seen by each element of the system is approximately the same [4].

An understanding of the intrinsic interactions between the diffuse local clocks and their cumulative effect on network-wide performance is considered in this paper. Over the last decade, most improvements were made in the synchronization algorithm trying to minimize message exchange and optimize radio architectures to provide accurate time-stamping mechanisms [5]. The influences of the underlying clock system and its impact on the overall synchronization accuracy has largely been unstudied. Here, we concentrate on the problem of choosing the right clock and duty cycle for a sensor node with order to achieve correct synchronization and long lifetime.

2 Clock Synchronization in WSNs

In centralized systems there is no need for synchronized time because there is no time ambiguity. Contrary to this, in distributed systems, such as WSNs, there is no global clock or common memory.

In WSNs each processor in sensor node (SN) has its own internal clock and its own notion of time. In practice, these clocks can easily drift second per day, accumulating significant errors over time, and thus potentially remaining unsynchronized.

For more applications and algorithms that run in WSN, we need to know more about time synchronization having in mind the following:

(a) The time of a day at which an event happened on a specific SN in the WSN.
(b) The time interval between two events happened on different SNs in the WSN.
(c) The relative ordering of events that happened on different SNs in the WSN.

Clock synchronization in WSN is the process of ensuring that physically distributed SNs have a common notion of time. There are three basic solutions for time synchronization in WSN [5]:

1. *One-way message dissemination*: the simplest form of synchronization deals only with ordering of events or messages. Using this approach it is possible to tell whether an event E_1 has occurred before or after another event E_2.
2. *Receiver-receiver synchronization*: SNs run their local clocks independently, but they keep information about the relative drift and offset of their clock to other clocks in the network.
3. *Two way message exchange*: the most complex form of synchronization called "always on" model, where all SNs maintain a clock that is synchronized to a reference clock in the network.

3 Clocking Terminology

Every individual SN in WSN has it own clock. Time in SN is usually kept by a specialized sub-system illustrated in Figure 1 [6]:

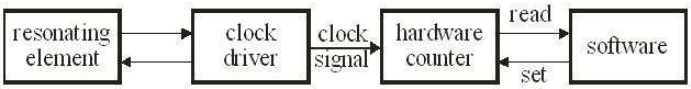

Fig. 1. Basic block diagram of a clock circuit and associated timer hardware.

The angular frequency f of the resonating element determines the rate at which the clock runs. The clock signal is periodic and the period is $T = 1/f$. It increments a hardware counter every $1/f$ seconds. At any time t, since the n-bit counter was reset, the counter reads $c(t) = \lfloor f \times t \rfloor \mod 2^n$. The $1/f$ rate at which the counter is incremented is called *the resolution*. In general, high resolution is not useful if the software cannot read the counter at that speed. Having this in mind, the

smallest increment at which an application can read a counter is called *precision*. The counter is typically set to an Universal Time Calendar, UTC. The accuracy determines how true the counter is close to that calendar [6]

For any two clocks C_a and C_b, we will point to the used terminology which is consistent with definitions given in [7].

Time: The time of a clock in a SN p is given by the function $C_p(t) = t$ for a perfect clock.

Frequency: Frequency is the rate at which a clock progresses. The frequency at time t of the clock C_a is $C_a'(t)$.

Offset: Clock offset is the difference between the time reported by a clock and the real time. The offset of the clock C_a is given by $C_a(t)$. The offset of clock C_a relative to clock C_b at time $t \geq 0$ is given by $C_a(t) - C_b(t)$.

Skew: The skew of a clock is the difference in the frequencies of the clock and the perfect clock. The skew of a clock C_a relative to clock C_b at time t is $C_a'(t) - C_b'(t)$.

If the skew is bounded by ρ, then as per eq. (1), clock values are allowed to diverge at a rate in the range of $1-\rho$ to $1+\rho$.

Drift (rate): The drift of clock C_a is the second derivative of the clock value with respect to time, namely C_a''. The drift of clock C_a relative to clock C_b at time t is $C_a''(t) - C_b''(t)$.

Due to clock inaccuracy the clock is said to be working within its specification

$$1-\rho \leq \frac{\mathrm{d}\,C}{\mathrm{d}\,t} < 1+\rho \tag{1}$$

where ρ is the maximum skew rate specified by the manufacturer. Figure 2 illustrates the behavior of fast, slow, and perfect clock with respect to UTC [7].

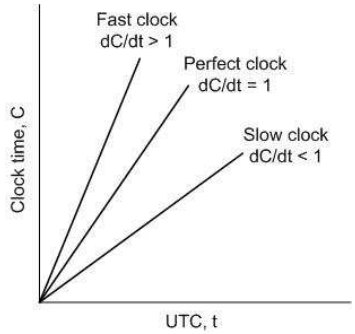

Fig. 2. Behavior of fast, slow and perfect clock with respect to UTC.

Equation (1) can be transformed into:

$$(1-\rho)\Delta t \leq \Delta C < (1+\rho)\Delta t \tag{2}$$

$$\frac{\Delta C}{1+\rho} \leq \Delta t < \frac{\Delta C}{1-\rho} \tag{3}$$

It can be inferred that the local clock difference ΔC that corresponds to the real-time difference Δt can be bounded by the following interval:

$$[(1-\rho)\Delta t,\ (1+\rho)\Delta t] \tag{4}$$

4 Sender to Receiver Synchronization

One standard technique to deal with clock uncertainty is to estimate the frequency error of the local clock. The following two factors, manufacturing inaccuracies and changes in environmental temperature, are the two major contributors to clock uncertainties. While manufacturing imprecision is static over the lifetime of a component, changes in environmental temperature require frequent recalibration. In addition, problems in clock synchronization occur because uncertainties are introduced by the wireless channel, too. Deep fading, interference, unpredictable latencies due to the broadcast nature of the wireless channel, and higher bit error rates, than in wired networks, all complicate synchronization efforts.

We use traditional sender-to-receiver synchronization which usually happens in the following three steps:

1. The sender node periodically sends a message with its local time as a time-stamp to the receiver.
2. The receiver then synchronizes with the sender using the time-stamp it receives from the sender.
3. The message delay between the sender and receiver is calculated.

Most majority methods synchronize a sender with a receiver by transmitting the current clock values as time-stamps. We call this kind of synchronization as one-way message dissemination.

5 Main Characteristics of WSNs

The basic operation of WSNs is data fusion. In spite of that data fusion requires that SNs be synchronized. The synchronization protocols for WSNs must address the following features of these networks [7]:

Limited energy: WSNs can employ thousands of battery-powered SNs. Since the amount of energy available to such sensors is quite modest, synchronization must be achieved, while data processing is active with order to utilize these sensors in an efficient fashion.

Limited bandwidth: in WSNs much less power is consumed in processing data than transmitting it. Bandwidth limitation directly affects message exchanges among SNs, and synchronization is impossible without message exchanges.

Limited hardware: the hardware of a SN is usually very restricted due to its small size. The size of SN cannot be increased because it would make it more expensive and consume more power. Table 1 summarizes some of the typical operating characteristics for four different SN classes [8].

Unstable network connections: the following issues are important: i) the communication range of SN is limited, therefore, message exchange among SNs is difficult; ii) the wireless medium is unshielded to external interference, so we have high percentage of message loss; iii) wireless connection suffers from restricted bandwidth; ivi) due to SN mobility the network topology frequently changes.

Table 1. Classification of hardware platform for sensor networks

Node Type	Typical application sensors	Radio Bandwidth [kbps]	MPS FLASH RAM	Typical active energy [mW]	Typical sleep energy [μW]	Typical duty cycle [%]
Specialized sensing platform	Spcialized low bandwiddth or advanced RF tag	< 50	< 5 < 0.1 MB < 4 kB	1.8 V (10-15) mA	1.8 V 1 μA	(0.1-0.5)%
Genering sensing platform	General purpose sensing and commun. relay	< 100	< 10 < 0.5 MB < 10 MB	3 V $(10-15)$ mA	3 V 10 μ A	(1-2)%
High bandwidth sensing	High band. sensing (video, acoustic, and vibration)	~ 500	< 500 < 10 MB 128 kB	3 V 60 mA	3 V 100 μA	(5-10)%
Gateway	High band. sensing and commun. agregation, gateway node	> 500 to 10×10^3	> 100 > 32 MB > 512 kB	3 V 200 ma	3 V 10 mA	> 50 %

6 Design Aspects of Clock Synchronization in WSNs

The following design considerations are important in WSNs [7]:

Energy efficiency: in WSNs energy conservation is very important. During the past period the energy consumption of most time synchronization services in WSNs has largely been ignored. At the best case, the implemented methods compared the

number of messages sent per synchronization exchange between SNs. While this is an important measure, other components of the hardware platform start to become much more relevant in the power budget as communication overhead gets reduced. These overheads are particularly pronounced in the extremely low duty-cycles that WSNs try to achieve. In such scenario, a high precision and high frequency clock can quickly become the most significant energy consumer during sleep times, thus invalidating any power gains made in synchronization accuracy.

Infrastructure: the SNs must cooperate to organize themselves into a network and resolve contention for available bandwidth.

End-to-end-delivery: there is a need for localization algorithms to reduce latency error as well as jitter, i.e. the unpredictable variation in transmission time. Table 2 summarizes the magnitudes and distribution of the various delays in message transmission [9].

Table 2. The sources of delays in message transmissions

Time	Magnitude	Distribution
Send and Receive	0-100 ms	nondeterministic, depends on the processor load
Access	10-500 ms	nondeterministic, depends on the channel contention
Transmission /Reception	10-20 ms	deterministic, depends on message length
Propagation	1 μs for distances up to 300 meters	deterministic, depends on the distance between sender and receiver
Interrupt Handling	< 5 μs in most cases, but can be as high as 30 μs	nondeterministic, depends on interrupts being disabled
Encoding plus Decoding	100-200 μs, < 2 μs variance	deterministic, depends on radio chipset and settings
Byte Alignment	0-400 μs	deterministic, can be calculated

Message loss and message delivery: fault tolerant algorithms handle message loss by sending extra messages.

Network dynamics: in some WSNs the nodes are mobile. Mobility directly loads to a frequent change in topology of the network, so self-configurability is used with aim to achieve synchronization.

7 Up-To-Date Methods of WSN Synchronization

There are several reasons for addressing synchronization problem in WSNs. We will point to two most important. First, SNs need to coordinate their operations and collaborate to achieve a complex sensing task. Data fusion is an example of such coordination in which data collected at different SNs are aggregated into meaningful results. Second, synchronization can be used by power saving schemes to increase network life-time. For example, SNs may sleep at appropriate time, and wake-up when necessary. When using power-saving nodes, the SNs should sleep and wake-up at coordinated times, such that the radio receiver of SN is not turned-off when there is some data directed to it. This requires a precise timing between SNs. Many different methods of time synchronization are in common use today [4].

The well known synchronization protocols for wired networks, such as Network Time Protocol (NTP), Precision Time Protocol (PTP), IEEE 1588v2 and IEEE 802.1AS are usually not suitable for WSNs due to limited hardware and energy resources available on SNs [4, 10]. To bypass these problems, in recent years, several WSN specific synchronization protocols have been proposed. However most of them are focused on achieving high synchronization accuracy for the whole network without addressing the power consumption problem. We will group WSN time synchronization methods into the following three categories [10]: a) high accuracy; b) light-weight; and c) adaptive. A short description of standard protocols that belong to each of the aforementioned groups follows:

a) **High-accuracy synchronization methods** - the following four protocols are typical for this group.

   a1) **Time-Stamp Synchronization (TSS) protocol** [11] - 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, the node estimates the time-stamp edge, and then subtracts the time-stamp edge from the message arrival time. The result of this subtraction is the time-stamp value which corresponds to the receiver time.

   a2) **Reference-Broadcast Synchronization (RBS) protocol** [12] - 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 (excluding the reference node). The nodes compute their relative clock drift rates by means of least-square linear regression.

a3) **Timing-sync Protocol for Sensor Networks** (TPSN) [13] - works in two phases. During the first, called level discovery phase, a spanning tree is created in the network, i.e. to each node a level number is assigned. At the start the root of the tree at level 0 broadcasts a level discovery packet which contains the root ID and level number. The nodes receive that packet and assign themselves a level number, greater by one than the level number of the received packet. In the second, synchronization phase, the root broadcasts a special packet to initiate synchronization.

a4) **Flooding Time Synchronization Protocol** (FTSP) [9] - floods the whole network with messages, which contain values of the global time, i.e. the time of the elected 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 and forms a reference point, which contains global and local time. When the node collects a sufficient number of referent points, it computes the drift rate of its clock with respect to the leader clock using linear regression.

b) **Light-Weight Synchronization Methods** - in general, accurate time synchronization needs more complex computations and more frequent network communications, which leads to an increased energy consumption. In order to improve power efficiency by reducing the synchronization overhead related to communication and computation several efficient protocol are proposed [10].

b1) **Tiny-Sync and Mini-Sync (TS/MS) protocol** [14] - a hierarchy of network nodes exists where each parent and child can exchange time-stamped radio messages. In this case improvement of power efficiency is achieved by reducing the synchronization complexity, and by enlarging the synchronization period.

b2) **Lightweight Tree-based Synchronization (LTS) scheme** [15] - saves power by means of fewer radio messages and less complex computations are necessary for the time synchronization. Two LTS algorithms perform periodic synchronization by exchanging a pair of time stamped messages along edges of a spanning tree. In the first, centralized algorithm, a reference node (i.e. root of the tree) synchronizes with its single hop neighbors, then they synchronize with their children, until all leaf nodes of the tree are synchronized. 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.

c) **Adaptive Synchronization Methods** - TS/MS and LTS often include separate ad hoc synchronization methods, each of which is the most suitable in certain scenario, depending on the required accuracy or number of active nodes in WSN. However these parameters can change rather quickly in WSN [10]. Therefore 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 SNs in most energy-efficient way [10]. Some of the protocols which belong in this group are the following.

   c1) **Adaptive Time Synchronization (ATS) protocol** [16] - uses the minimum number of synchronization messages to achieve a required accuracy with a certain probability. Each of the dedicated SNs 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.
   c2) **Energy Efficient Time Synchronization Protocol (ETSP)** [17] - minimizes the number of synchronizations, what depends on the number of SNs requiring synchronization. This technique is based on the observation that receiver-receiver synchronization (used in RBS) requires less transmission than sender-receiver synchronization (used in TPSN) when the number of SNs is small. On the contrary, the sender-receiver approach is more energy efficient in large and dense WSNs.
   c3) **Rate Adaptive Time Synchronization (RATS)** [18] - multiplicatively decreases and increases the synchronization interval within certain limits depending on how many times the synchronization error exceeds the user-defined bound.

More details concerning the principles of operation of all three groups of protocols can be found in [7, 10, 19].

8 How to Prolong Lifetime of SN?

Lifetime refers to the time period for which a sensor network is capable of sensing and transmitting the sensed data to the base station(s). In WSNs, thousands of

nodes are powered with very limited supply of battery power. As a result, lifetime analysis becomes an important aspect to efficiently use the available energy. In sensor networks using rechargeable energy, such as solar energy, lifetime analysis helps the node to use energy efficiently before recharging. Lifetime analysis may include an upper bound on the lifetime and factors influencing this upper bound [20].

Sensor networks should operate with the minimum possible energy to increase the life of sensor nodes. This requires power aware computation/communication component technology, low-energy signaling and networking, and power aware software communication. Design challenges encountered in the building of WSNs can be broadly classified into hardware, wireless networking, and OS/applications [3]. All three categories should minimize the power usage to increase the life of a sensor node. Hardware includes the design activities related to all hardware platforms (MEMS, digital circuit design, system integration, and RF) that make up sensor networks. The second aspect includes design of power-efficient algorithms and protocols (energy efficient protocols for MAC and routing like that discussed in the previous section). The third relates to power management in sensor nodes. Namely, additional power savings can be obtained by using Dynamic Power Management (DPM). The basic idea behind DPM is to shutdown (sleep mode) the SNs when not needed and get them back (wake up) when required. Our design solution, presented in this paper, is based on combination of DPM with timer logic as programmable hardware unit.

9 Duty Cycling

In order to minimize the energy consumption of wireless sensor node different techniques for reducing power consumption are used [3]. Among them duty cycling has become a crucial one. The idea behind this is clear. Keep hardware in a low power sleep state except on the infrequent instances when the hardware is needed. In many design solutions this allows even the processor to be put into a low power state for extended periods of time while only an external clock tracks time to trigger later wake-up [6, 7, 20, 21].

However, clock stability represents a limiting factor for the duty cycling that is possible in WSN using scheduled communications. Namely, when duty cycling is implemented in scheduled communications it is very important for SNs to wake up at the correct time so that they can communicate. During this, less stable clocks require nodes to more frequently synchronize in order to cope with clock drift. In general, more stable clocks can be used to improve duty cycling capabilities, thus indirectly saving energy and reducing communication bandwidth, since less

frequent resynchronization are then needed [**?**].

Currently available high stability clocks do not reduce power consumption due to the increased consumption of the clock. Therefore, bounds on synchronization error and constraints on power consumption are important considerations when designing WSNs. The behavior of the frequency error largely depends on the underlying technology used to generate the clock signal itself. In general, two components (see Fig. 1) are necessary to create oscillation, a resonating element and clock driver [6].

The resonating element is responsible to create an oscillation. The element itself however does not sustain the oscillation and, therefore, clock drive circuit is necessary to initiate and sustain that oscillation. More specifically, the clock driver provides both feedback (with gain ≥ 1 and total loop phase $\geq 2\pi$) and isolation to the resonator. CMOS buffers or inverters are well-suited as clock drivers since they have high input impedance, high gain, and high bandwidth. Table III summarizes the characteristics of the most common resonating elements, as candidate building block of the clock circuit in sensor node [6].

Table 3. Comparison of different resonating elements

Type	Stability	Power	Cost
LC/RC	1000's of ppm	low	cents to fee
Ring oscillator	1000's of ppm	low	cents to fee
Crystal	10's of ppm $<$ 1 ppm if controlled	low-high freq. dependent	10's of cents to $s
Crystal oscillator	$<$ 1 ppm to 10's of ppm	medium-high	$ s to 10's of $s
MEMS resonator	10's ppm to 100's ppm	high	10's of cents to $s

It comes as no surprise that most research aiming at prolonging the lifetime of WSNs focuses on limiting the radio operation of their devices. Indeed, the radio circuit of some sensor devices are measured to consume three order of magnitude more power than the rest of the hardware (CPU, memory, etc), either when the radio is in transmitting or receiving mode [**?**, 3, 20].

The main way to limit the operation of the radio is to limit the time for which the radio circuitry is switched on. This implies intermittently switching the radio on and off. The periods during which a node's radio is on or off are known as its active and inactive period, respectively. The fraction of the time that a node's radio is on, is known as the duty cycle (DC). That is

$$DC = \frac{T_{active}}{T_{active} + T_{inactive}} = \frac{T_{active}}{T} \tag{5}$$

For example, a node that is active for 10 ms every second has a duty cycle of

$$DC = \frac{10\text{ms}}{1000\text{ms}} = 1\%$$

The duty cycled-based operation of the nodes makes the synchronization of the active periods of their frames essential. Nodes whose active periods do not overlap cannot communicate with each other. In this paper we focus on the problem how for a given clock oscillator, and as small as possible duty cycle, to provide correct synchronization of the sensor node within the WSN.

10 Low Power MCU for SN

The battery-powered SNs (developed as small intelligent devices in homes, plantations, oceans, rivers, streets, and highways to monitor the real-world environment) in which the battery must lost for up to ten years (frequent battery replacement is undesirable) are applications for which power consumption is very important. As these devices become increasingly power-conscious, the need for smart power management becomes equally significant and demand for usage of ultralow power microcontroller unit (MCU), as SN's constituent, is rising. The choice of MCU is critical, too. In a attempt to prolong battery life, software engineers go to great lengths to optimize code, minimize memory accesses, etc. Hardware engineers focus on ways to shut down unused circuitry, ensure that all quiscent currents and leakage paths are minimized, and maximize power-supply efficiency [21].

In many battery-operated applications, the MCU does not run continously and peripherals may be idle most of the time. For those applications, sleep mode may represent the lion's share of power consumption and is the vital parameter to consider. In sleep mode, the MCU core, internal memory, RF transmitter, sensor electronics, and other interface logic are switched off, program execution stops, while specific clock supply and associoated peripherals continue to run. The biggest power savings are possible by frequently switching the MCU to sleep mode (see Figure 3) [21].

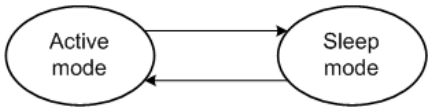

Fig. 3. Active mode vs Sleep mode.

At the basic level, MCU consumption can be defined as the sum of the following:

$$\begin{aligned} Total\_power\_consumed = {} & Active\_mode\_power \\ & + Sleep\_mode\_power \end{aligned} \tag{6}$$

However, another important metric to keep in mind is the amount of time it takes for an MCU to transition from a standbay state into an active state and vice-versa. Since the MCU cannot do any useful processing until all of the digital and analog components are fully settled and operational, it is important to add this (wasted) power when calculating total power consumption (see Figure 4).

$$\begin{aligned} Total\_power\_consumed = Active\_mode\_power + Sleep\_mode\_power \\ + Wake\_up\_power + Shut\_down\_power \end{aligned} \tag{7}$$

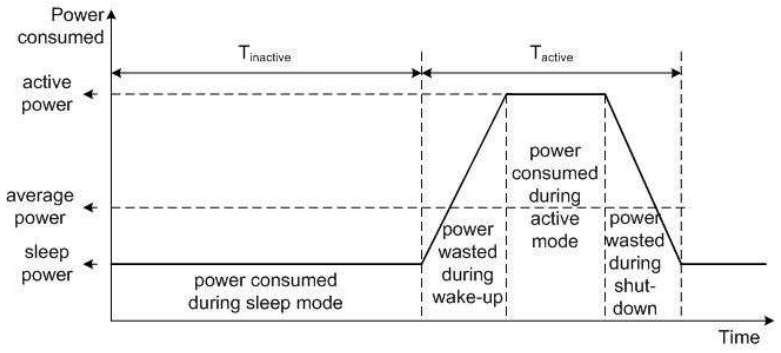

Fig. 4. Power budget.

If the sensor uses a battery rated at 750 mAh, the SN must draw less than (750 mAh / 70080 h) 10.7 μA average current to provide eight years of battery life. Since the SN can spend most of the time inactive, the designer must use a power budget, i.e., to determine the correct duty-cycle as a ratio between low power and active modes in order to calculate the average power consumption and battery requirements.

A short discussion on each of power consumption components will help highlight the types of issues systems designers need to be aware when attempting to select the best MCU solution for their SN designs.

Active mode power: For a CMOS logic gate, the dynamic power consumption is given by the following equation:

$$Active\_mode\_power = C \times V^2 \times f \tag{8}$$

where C is the load capacitance, V is the supply voltage, and f is the switching frequency.

The capacitance term is a function of the design and processing technology being used, and the frequency term is a function of the application's processing requirements. The supply voltage has a major impact on the overall power consumed by the SN.

Advanced power architecture, such as on-chip low drop-out linear voltage regulator, voltage islands, dynamic threshold voltage control, etc., can be used to maintain a constant active current over the full operating voltage range and can help systems designer to achieve a significant savings in power consumption [22].

Sleep mode power: To save energy in WSN it is a desirable to keep SNs in low-power state, if not turned off completely, for as long as possible. SN hardware is often designed with this goal in mind; processor have various "sleep" modes or is capable of powering down high-energy peripherals when not in use.

Running the MCU at full-speed all the time will never lead to a truly low-power design even if the lowest-power MCU is used. The biggest power savings are only possible by frequently switching the MCU to a low power sleep state from normal state and vice-versa. The MCU is switched into low-power mode by configuring bits in the status and control registers, i.e. by issuing a sleep command over the bus. Achieving maximum energy efficiency (and battery life) translates into ensuring that each MCU task consumes the minimum possible current at the minimum possible voltage for the shortest possible duration, so that the device spends the majority of its time in very low-power sleep mode. The sleep, or low power mode, vary in degree to which the MCU is aware of its surroundings and the different clocks that the SN must keep runing. The more common sleep modes are Idle mode, Power Save, and Power Down [23]. Idle mode is a shallow sleep mode where only parts of the SN are shut down but the main parts of the MCU are running. In Power Save mode, everything is turned off except a 32 kHz clock running from a crystal to keep track of the time. It allows only the base timer or watch prescaler to work. The operating clock for the MCU and peripheral resources is stopped in this mode to reduce the power consumption. In Power Down mode, everything is is shut down, including the clock source. All clocks are halted, but the MCU status, RAM and register contents are preserved. It has the lowest current consumption and only external interrupts can wakeup MCU from Power Down mode [23].

The advantage of having multiple sleep modes is the flexibility it provides to shut down any part of the SN that is not absolutely necessary to the function at hand. The amount of power that can be saved depends on the mode being used.

Most vendors offer different standby low-power option. Most suppliers will highlight their absolute lowest sleep mode current, which will often correspond to the current being consumed with the real-time clock and brownout detector (continuous supply voltage monitoring) disabled. Some vendors will go a step further and quote a shutdown mode current that does not retain memory and requires a reset to wake up, which in general is not a very practical mode [3, 23]. Therefore, since most applications will require full RAM and register retention, it is important for a system designer to perform a side-by-side comparisons based on the following

metrics [23]:

1. Sleep mode current with real-time clock and brownout disabled (with RAM retention),
2. Sleep mode current with real-time clock disabled and brownout enabled,
3. Sleep mode current with real-time clock and brownout enabled.

A system designer can then use the corect values when calculating the overall sleep mode power budget based on the duty cycle of their application.

Wake-up and shut-down power mode: In systems that use sleep mode a significant amount of power can be wasted waking up (shut-down) the SN and preparing it to acquire or process data. In fact, in certain applications an SN can often use just as much energy when comming out of sleep (standby) as when the SN is fully processing data. Therefore, it is important to design an SN to wake-up (shut-down) and settle in an extremelly short amount of time in order to minimize the amount of time spent in an energy-wasting state.

The SN, ie. the MCU, should be able to exit sleep mode from either an external trigger or an interval timer. The most flexible periodic wake-up (shut-down) source is a real-time clock having the capability of being run from an external crystal oscillator or from a low-frequency internal ring oscillator that eliminates the need for a crystal in lower-accuracy applications, like that used in agriculture. Avoid using a slow-starting crystal oscillator for high-speed clock; an accurate, quick-starting, on-chip oscillator is a better alternative.

Something as conclusion concernig selection of low-power MCU: Having in mind that every SN application will be affected by the combination of sleep mode power, active mode power and wake-up and shut-down mode power, it may be helpful for systems designers to simply start any analysis by sistematically breaking down the power consumption numbers into the parts afore mentioned. Once these numbers have been derived, a system designer can then factor in the application's duty cycles - the amont of time the application expects to spend in sleep, active, wake-up and shut-down modes - to calculate an overall average consumption number. The resulting value should provide the system designer a close approximation that can be used to objectively evaluate and compare SN alternatives to achieve the lowest possible system-level power consumption.

The primary factor of low-power modes is that recovering (wake-up) into normal operating modes and returning to low power mode can impose a significant delay. Therefore, a key feature to look in any MCU is the shortest wake-up and shut-down time. MCU wake-up time from some low-power modes should be fast enough to meet the response times of interrupts. Total $T_{restore}$ is approximatyely 3.5 ms with $T_{wake-up}$ =1.4 ms, $T_{shut-down} = 1$ ms with PLL stabilization of 1 ms, and

gear-up time of about 75 μs. This illustrates the ability to optimize the design for low-power while having a flexible architecture in the MCU that helps to achieve high-performance and low-power consumption.

11 Battery Issues

From the system's perspective, a good micro-battery should have the following features [21]: 1) high energy density; 2) large active volume to packaging volume ratio; 3) small cell potential (0.5 - 1.0 V) so digital circuits can take advantages of the quadratic reduction in power consumption with supply voltage; 4) efficiently configured into series batteries to provide a variety of cell potentials for various components of the system without requiring the overhead of voltage converters; 5) rechargeable in case the system has an energy harvester.

A number of small batteries are being developed until now for wireless communications. It seems that three cell chemistries currently dominate the growing wireless sensor network application market: Nickel-Metal Hydride (NiMH), Lithium Ion (Li-Ion), and Lithium Polymer (Li-polymer).

Each battery type has unique characteristics that make it appropriate, or inappropriate, for a SN. Knowing the specific characteristics of each cell chemistry in terms of voltage, cycles, load current, energy density, charge time, and discharge rates is the first step in selecting a cell for a SN. The following discussion gives a short overview of the characteristics, strengths, and weaknesses of each of the three cell chemistries.

The crucial battery parameters are given in Table 4 [3, 24–26].

Table 4. Battery types

Type	Voltage	Energy density	Specific energy	Self discharge
Lead-Acid	2.0 V	60-75 Wh/dm^3	30-40 Wh/kg	3-20 %/month
Nickel Cadmium	1.2 V	50-150 Wh/dm^3	40-60 Wh/kg	10 %/month
Nickel Metal Hybrid	1.2 V	140-300 Wh/dm^3	30-80 Wh/kg	30 %/month
Lithium-Ion	3.6 V	270 Wh/dm^3	160 Wh/kg	5%/month
Lithium-Polymer	3.7 V	300 Wh/dm^3	130-200 Wh/kg	1-2 %/month

The volumetric energy density of Lithium-ion batteries is within the interval

$$w_e = \frac{W_e}{V_1} = (0.90 \div 2.23)\text{MJ/L} \quad \text{or}$$
$$= (250 \div 620)W\frac{h}{L}$$

where W_e is accumulated electrical energy and V_1 is active volume in liter [24, 25].

If we assume that $w_e = 2$ MJ/L, the useful volume $V_1 = 10^3$ L and the nominal battery operating voltage (average potential difference) is $E_B = 3.6$ V then we have

1. The accumulated electrical energy into a form of chemical energy is

$$W_{ch} = w_e V_1 = 2 \times 10^6 \times 10^3 = 2 \times 10^{-3} \text{ J}$$

2. The electrical energy which, from battery with nominal operating voltage $E_B = 3.6$ V during the battery life-time t_B, is delivered in a form of work, for an average current of $I_a = 8\ \mu$A, is equal to

$$E_e = E_b I_a t_B = 3.6 \times 8 \times 10^{-6} \times t_B$$

For $W_{Ch} = W_e$ it is possible to determine the usage battery time

$$t_B = \frac{2 \times 10^3}{3.6 \times 8 \times 10^{-6}} = 69\,444\,444 \text{ s}$$

Since in one year we have $365 \times 24 \times 3\,600 = 31\,536\,000$ s, by dividing with this value we obtain

$$t_B = \frac{69\,444\,444}{31\,536\,000} = 2.202 \text{ years}$$

In Table 5, for different values of average circuit current I_{av}, the battery life-times t_B, are calculated.

Table 5. Battery operating characteristic

$I_a\ \mu$A	t_B [years]	k_s	k_{rs}	N_m
2.74	6.429	4380	0.99	14 741 077
5	3.523	2 400	0.99	8 080 096
8	2.202	1 500	0.99	5 050 347
10	1.726	1 200	0.99	4 041 195

The current saving factor, k_s, as a metric, corresponds to the ratio between the maximum, I_{max}, and average current, I_a, of the sensor node. For $I_a = 8\ \mu$A and $I_{max} = 12$ mA [3] we have

$$k_s = \frac{I_{max}}{I_a} = 1\,500 \tag{9}$$

The relative current saving factor is defined as the following ratio

$$k_{rs} = \frac{I_{max} - Ia}{I_{max}} \tag{10}$$

For $I_a = 8\ \mu$A and $I_{max} = 12$ mA we obtain $k_{rs} = 0.99$.

The total number of measuring cycles for $DC = 10^{-3}$, N_m, is given by

$$N_m = \frac{t_B}{t_{cycle}} \tag{11}$$

For $I_a = 8\ \mu$A, $t_{cycle} = 13.75$ s, number of transmitted bytes per packet 64, and data transfer rate 50 kbps, we have $N_m = 14\,741\,077$.

In Table 5 calculated values for k_s, k_{rs} and N_m, for different average current I_a, and fixed DC and t_{cycle} are given.

12 Workload Profile of Sensor Node

As is shown in Fig. 5 a typical workload profile for SN consists of two distinct phases [27]:

1. Low workload - corresponds to the state of a wireless SN in the absence of intruders. SNs periodically wake-up, sample their sensors in order to detect any intruders, and, in their absence, go back to sleep. To cope with high energy efficiency in this phase a SN should provide: a) ultra low power sleep mode; and b) rapid wake-up capability.
2. High workload - represents the state when intruder activity is detected. During this phase the SN performs significant amount of computation and communication with other SNs.

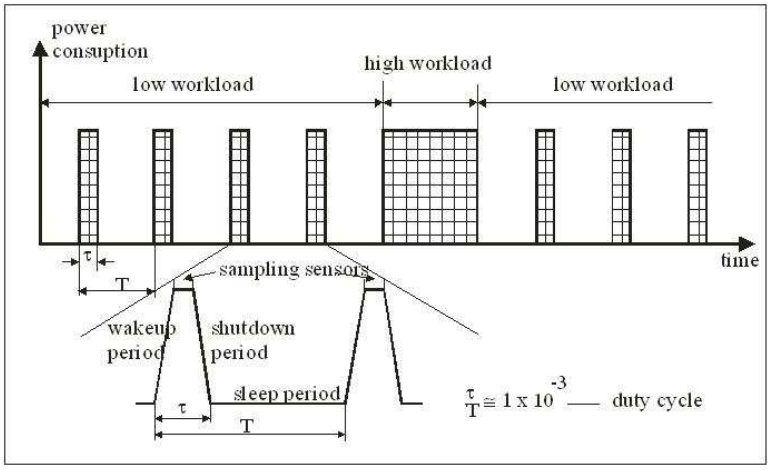

Fig. 5. Two phases of SN's operation.

The following two approaches are used for reducing energy consumed by a SN [28]:

1. duty cycling - consists of waking-up the SN only for the time needed to acquire a new set of samples and then powering it off immediately afterwards;
2. adaptive-sensing strategy - is able to dynamically change the SN activity to the real dynamics of the process.

Power consumption is the product of operating voltage, $E_B = V_{CC}$, multiplied by the current consumption, I_{CC}. Usually I_{CC} is the only measure while describing power characteristics of a SN or the chip. This is a mistake because decreasing V_{CC} directly reduces the current consumption and the overall power gain. In low power designs, the average current consumption, I_a, determines battery life.

13 Implementation of Duty Cycle Technique

As we have already mentioned radio duty-cycling has received significant attention in sensor networking, particularly in the form of MAC protocols and topology management.

The dominant factor that prevents the optimal usage of the radio in real deployment settings is time uncertainty between SNs. Existing duty-cycling techniques use a variety of approaches to deal with time uncertainty. BMAC [29] uses an asynchronous technique that involves no time synchronization or clock estimation. Instead, each packet is transmitted with a long preamble which is chosen such that the receiver would wake-up some time during the preamble (see for example Figure 6). This incurs significant transmission overhead. For example, with 11.5 %

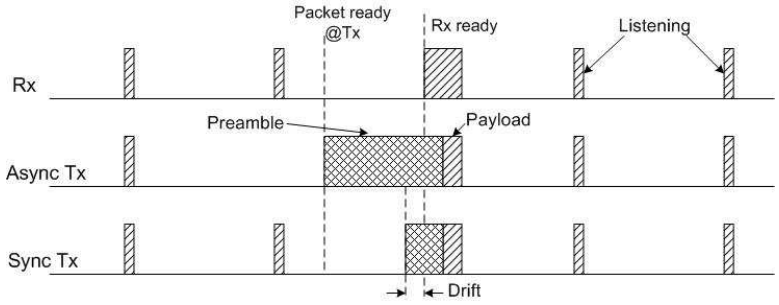

Fig. 6. Transmission mechanisms.

duty cycle a preamble of 250 bytes is used to transmit a 29 byte payload. Other techniques such as SMAC and TMAC [29] use synchronized techniques where explicit time synchronization beacons are transmitted periodically between neighboring nodes. This enables the transmitter to turn on the radio at the right moment (see Figure 6) but the inability to deal effectively with time varying changes in clock drift force these techniques to resynchronize frequently. Therefore, existing

radio duty-cycling approaches expend a lot of energy in handling time uncertainty between SNs.

Typical frequency skew of ±30-50 ppm due to manufacturing variations, and additional ±10-20 ppm due to temperature variation, and an exponentially decreasing ±3 ppm per year are common [30]. These frequencies skews result in relative clock disproportion between SNs, and, as a result, SNs must include a guard time, equal to the maximum drift, which grows linearly with the interval between communications [6, 30]. Let s_{skew} ($\Delta f/f$ in Hz/Hz) be the frequency skew and T_{tot} be the total packet period, than the minimum guard time becomes

$$t_{guard} = 2s_{skew}T_{tot} \tag{12}$$

If it takes t_{sp} time to send a packet, then the total transmission time will be $t_{guard} + t_{sp}$ and the DC is

$$DC = \frac{2s_{skew}T_{tot} + t_{sp}}{T_{tot}} \tag{13}$$

which can be simplified to

$$DC = 2s_{skew} + \frac{t_{sp}}{T_{tot}} \tag{14}$$

Equation (14) establishes a lower bound of $2s_{skew}$ on the DC and points to three solutions how to reduce radio on-time [30]. The first solution is based on reducing the frequency skew with higher tolerance crystal oscillators, power-hungry temperature-compensated crystal oscillators, or improved calibration. A second solution is to reduce the packet transmission time by increasing the radio speed. Reducing radio wake-up and shut-down times is also preferable. The third solution deals with decreasing the data rate by increasing the communications period.

Most of the proposed techniques for time synchronization ignore the power overhead that a time synchronization protocol introduces. In general if a system needs less than 10 ms accuracy, then a time synchronization of up to 1000 s is sufficient to guarantee such approaches. In most systems, this interval is much larger than the communication interval necessary to transmit sensor data from the nodes to a fusion center and we can assume that the overhead of time synchronization in these scenarios is minimal. However, if high accuracy is needed the synchronization intervals have to be of the order of tens of seconds. In these cases, time synchronization could be piggy-backed on regular messages, only introducing a small message overhead of the order of a timestamp.

As we have already mentioned each within WSN has its own clock oscillator. From metrological point of view the relative measuring inaccuracies, C, of the oscillator is in order of

$$\delta_r = \frac{\Delta t}{t_w} \sim 10^{-5} \tag{15}$$

where Δt corresponds to absolute measuring inaccuracy during the observed time interval of duration t_w.

According to eq (15) it is possible now to form the following two equations:

$$t_w = k_1 t \tag{16}$$

and

$$t_w = \frac{k_2}{\delta_r(t)} \tag{17}$$

By adding equations (16) and (17) and dividing by two we obtain

$$t_w = \frac{1}{2}\left[k_1 \Delta t + \frac{k_2}{\delta_r(t)}\right] \tag{18}$$

From the condition $\mathrm{d}t_w/\mathrm{d}(\Delta t) = 0$ we determine the constant k_2

$$k_2 = \Delta t \tag{19}$$

Having in mind that the left sides of eqs (13) and (14) are identical, and by substituting the result derived in eq. (16) we obtain

$$k_1 = \frac{1}{\delta_r(t)} \tag{20}$$

By substituting the results obtained in eqs (19) and (20) into (16) we obtain

$$t_w = k_1 k_2 = \frac{\Delta t}{\delta_r(t)} \tag{21}$$

According to eq. (16) we can conclude that the minimal width of the time window t increases as the error of clock reading Δt increases (refer to Figure 7).

In a similar way, equation (17) points out to the relative clock error decreasing, as the t_w width increases (refer to Figure 8).

As a conclusion we can say that: the minimal width of a time window t_w depends of the relative oscillator error $\delta_r(t)$ for a given value Δt.

14 Conclusion

Clock synchronization is a critical component in the operation of WSNs as it provide common frame to different nodes. WSNs make heavy use of synchronized time, but often have unique requirements in respect to lifetime of SN, precision of synchronization achieved, as well as the time and energy required to achieve it. Existing time synchronization methods need to be extended to meet all these

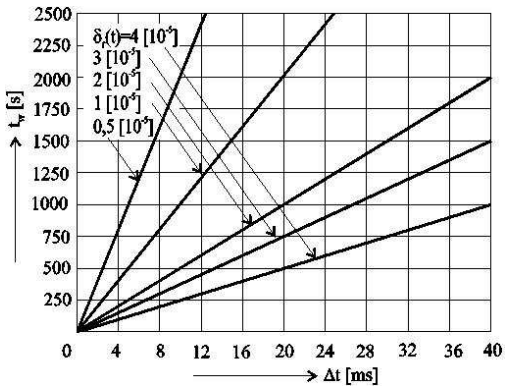

Fig. 7. Dependency of the time window t_w in term of Δt for a given relative measuring inaccuracy $\delta_r(t)$.

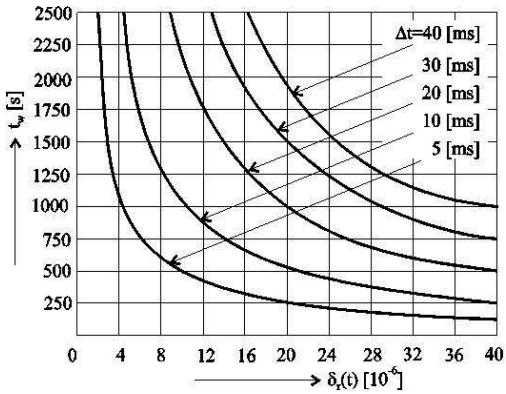

Fig. 8. Dependency of t_w in term of relative clock inaccuracy $\delta_r(t)$ for a given value of Δt.

needs. During this, choosing the right clock for a SN is an important factor for overall performance of SN, and WSN, too. In general radio operation dominates the SN power budget. Even at low duty cycles of approximately 1 to 2% radio operation is about an order of magnitude more expensive than all other operations combined (data manipulation by the MCU, AD and DA conversion, etc). Hence, lifetime improvements will require substantial reductions in radio on-time. Our analysis shows that the power aspects of clocking subsystems are not trivial, and that sometimes a simple assumption, like more stable clocks will save power in smaller guard band, have to be investigated carefully before taken for true. This work is concentrated on the better understanding of the link between clocks and power. In focus of our interest were radio duty-cycles for low-rate data collection applications, including polled radio operation (periodically samples the channel for

radio activity and power-down the radio between successive samples) and scheduled radio-operation (establishes well-known and periodic time intervals when it is legal (or illegal) to transmit). Our proposal is to use piggy-backing as the process of combining the data fusion messages with messages that carry synchronization data among nodes. Instead of sending independent clock synchronization messages, these messages are piggy-backed on the data fusion messages that have to be send to the SN. Piggy-backing is clearly advantageous because WSNs are often subject to limited communication bandwidth and power-consumption constraints. Further research has to be done on the link between clock stability and its effect on time synchronization. Currently, 1 μs seems to be the lower and of what is possible for time synchronization in WSN with current technology [31]. Future research will show why this is the case, and where the bottlenecks are, in order to break this barrier. The challenge will definitely lay in providing such accuracies while still maintaining the low-power requirements given by the power constraints of WSNs [31].

Acknowledgement

This work was supported by the Serbian Ministry of Science and Technological Development, Project No. TR-32009 - "Low-Power Reconfigurable Fault-Tolerant Platforms".

References

[1] S. Soloman, *Sensors and Control Systems in Manufacturing*. New York: McGraw Hill, 2010.

[2] A. Hac, *Wireless Sensor Network Designs*. Chichester: John Wiley & Sons, 2003.

[3] M. Stojčev, M. Kosanović, and L. Goluibović, "Power management and energy harvesting techniques for wireless sensor nodes," in *Proc. of IX International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services*, vol. 1, Nis, Serbia, Oct. 7–9, 2009, pp. 65–72.

[4] E. Serpedin and Q. M. Chaudhari, *Synchronization in WSN: Parameter Estimation, Performance Benchmarks and Protocols*. Cambridge University Press, 2009.

[5] Q. Yik-Chung Wu Chaudhari and E. Serpedin, "Clock synchronization of wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 124–138, Jan. 2011.

[6] T. Schmid, R. Shea, Z. Charbiwala, J. Freidman, and M. B. Srivastava, "On the interaction of clocks, power, and synchronization in duty-cycled embedded sensor nodes," *ACM Transactions on Sensor Networks*, vol. 7, no. 3, pp. 24.1–24.19, Sept. 2010.

[7] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: A survey," *Ad Hoc Networks*, vol. 3, no. 3, pp. 281–323, 2006.

[8] J. Syh and M. Horton, "Powering sensor networks," *IEEE Potentials*, vol. 23, no. 3, pp. 35–38, Nov. 2004.

[9] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in *Proc. of the 2nd International Conference on Embedded Networked Sensor Systems, SynSys'04*, Baltimor, Maryland, USA, Nov. 3–5, 2004, pp. 39–49.

[10] A. Ageev, "Time synchronization and energy efficiency in wireless sensor networks," Ph.D. dissertation, DISI - University of Trento, Mar. 2010.

[11] K. Romer, "Time synchronization in ad hoc networks," in *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 01)*, Long Beach, CA, USA, Oct. 2001, pp. 173–182.

[12] 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)*, vol. 36, Boston, MA, USA, Dec. 2002, pp. 147–163.

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

[14] M. L. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," in *Proc. Wireless Communi-cations and Networking Conference (WCNC 2003)*, vol. 2, 2003, pp. 1266–1273.

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

[16] 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)*, Apr. 2004, pp. 340–348.

[17] 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)*, Mar. 2008, pp. 971–976.

[18] 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)*, San Diego, CA, USA, Nov. 2005, pp. 130–141.

[19] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: a survey," *IEEE Network*, vol. 18, no. 4, pp. 45–50, 2004.

[20] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastan, "Energy aware wireless microsensor networks, IEEE Signal Processing Magazine," vol. 19, no. 3, pp. 40–50, Mar. 2002.

[21] J. E. Elson, "Time synchronization in WSN," Ph.D. dissertation, University of California, Los Angeles, 2003.

[22] M. Salas. (2011, May) Low power design basics: How to choose the optimal low power MCU for your embedded system. [Online]. Available: http://low-powerdesign.com/PDF/Low-Power-Design-Basics.pdf

[23] R. Dasgupta. (2011, May) Low power MCU selection criteria and sleep mode implementation using hardware/software codesign technique. [Online]. Available: http://www.eetimes.com /design/industrial-control/4014277/Low-power-MCU-selection-criteria-and-sleep-mode-implementation

[24] F. Pistoia, *Battery operated devices and systems: From portable electronics to industrial products*. Amsterdam: Elsevier, 2008.

[25] (2011, May) Lithium-ion battery. [Online]. Available: http://en.wikipedia.org/wiki/Lithium-ion_battery#cite_note-greencarcongress-1

[26] E. Jens, "Low-power design methodologies for embedded internet systems," Ph.D. dissertation, EISLAB, Lulea University of Technology, Sweden, 2008.

[27] V. Raghunathan, S. Ganeriwal, and M. Srivastava, "Emerging techniques for long lived wireless sensor networks," *IEEE Communication Magazine*, vol. 44, no. 4, pp. 108–114, Apr. 2006.

[28] C. Alippi, G. Anastasi, M. Di Francesco, and M. Roveri, "Energy management in wireless sensor networks with energy-hugry sensors," *IEEE Instrumentation & Measurement Magazine*, vol. 12, no. 2, pp. 16–23, Apr. 2009.

[29] S. Ganeriwal, D. Ganesan, H. Sim, V. Tsiatsis, and M. B. Srivastava. (2011, May) Estimating clock uncertainty for efficient duty-cycling in sensor networks. [Online]. Available: http://nesl.ee.ucla.edu/fw/documents/journal/2008/Ganeriwal08_TNET.pdf

[30] P. Dutta, D. Culler, and S. Shenker. (2011, May 20,) Procrastination might lead to a longer and more useful life. [Online]. Available: http://iteseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.93

[31] T. Schmid, R. Shea, J. Friedman, Z. Charbiwala, M. B. Srivastava, and Y. H. Cho. (2011, May) On the interaction of clocks and power in embedded sensor nodes. [Online]. Available: http://nesl.ee.ucla.edu/fw/thomas/schmid2009techreport.pdf