Does Clock Precision Influence ZigBee's Energy Consumptions?*

Christian Groß[1,**], Holger Hermanns[2], and Reza Pulungan[2]

[1] comlet Verteilte Systeme GmbH, Amerikastraße 21,
D-66482, Zweibrücken, Germany
[2] Department of Computer Science, Saarland University
Fax: +49 (681) 302-5636, D-66123, Saarbrücken, Germany
christian.gross@comlet.de, {hermanns,pulungan}@cs.uni-sb.de

Abstract. Wireless embedded sensor networks are predicted to provide attractive application possibilities in industry as well as at home. IEEE 802.15.4 and ZigBee are proposed as standards for such networks with a particular focus on pairing reliability with energy efficiency, while sacrificing high data rates.

IEEE 802.15.4 is configurable in many aspects, including the synchronicity of the communication, and the periodicity in which battery-powered sensors need to wake up to communicate. This paper develops a formal behavioral model for the energy implications of these options. The model is modularly specified using the language MODEST, which has an operational semantics mapping on stochastic timed automata. The latter are simulated using a variant of discrete-event simulation implemented in the tool MÖBIUS. We obtain estimated energy consumptions of a number of possible communication scenarios in accordance with the standards, and derive conclusions about the energy-optimal configuration of such networks. As a specific fine point, we investigate the effects of drifting clocks on the energy behavior of various application scenarios.

Keywords: Sensor networks, formal modelling, distributed coordination, power-aware design, clock drift.

1 Introduction

Quantitative analyses of ad hoc and wireless networks have in the past been concentrating on scalability and routing questions [6]. The predominantly applied techniques are based on simulation using enhanced tools such as GloMoSim or NS-2 [23,5], Omnet [21] or commercially available simulation tools, such as Opnet. Another approach is based on instruction-level simulation of the actual microcontroller codes [22,20].

The credibility of simulation results obtained using the above enhanced modelling tools seems not to be free of doubts. Such studies generally suffer from (1) unclarities of how simulation models are obtained from the modelling language, (2) the sheer number

* This work is supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS). See www.avacs.org for information.

** Part of this work was done while at Saarland University.

of parameters with non-obvious effects adjustable by the user, some of them having hidden effects on the simulation outcomes, (3) excessive simulation times needed to simulate the ensemble of many protocol stacks and states, and (4) the impossibility to validate simulation results through reproducible real-life experiments.

As a matter of fact, some recent articles have criticized the extremely poor quality and reproducibility of simulation-based experimentations with ad hoc or wireless networks [13,6]. Indeed, this questions the validity of simulation-based predictions for this area as a whole.

The work presented in this paper is among the few which attempt to attack the above mentioned principal problems. Other loosely similar approaches include [16,12]. While we still rely on discrete-event simulation as our analytic workhorse, we proceed in a drastically different way. The main difference is that (1) we use a language with a strictly formal semantics, which is equipped with well-established abstraction techniques. Consequently, the underlying stochastic model for simulation is well-defined and the obtained simulation results are trustworthy. (2) We expose all assumptions explicitly, since they are part of the formal system specification. (3) We do not model entire protocol stacks, but work with well-justified abstractions of lower layer effects. We consider worst-case scenarios, if no other information is available. But (4) we do not yet provide real-life experiments to back up our simulation results.

The model is modularly specified using the language MODEST, which has an operational semantics mapping on stochastic timed automata. The latter are simulated using a variant of discrete-event simulation implemented in the tool MÖBIUS. In this paper, that approach is applied to the IEEE 802.15.4 and ZigBee standards. This is a protocol family dedicated to low-bandwidth sensor networks operating on battery. IEEE 802.15.4 is configurable in many aspects, including the synchronicity of the communication, and the periodicity in which battery-powered sensors need to wake up to communicate. The particular configuration chosen has obvious – and non-obvious – implications on the lifetime of battery-powered devices. An obvious rule of thumb is, for instance, that battery-operated devices can survive longer timespans if they need to wake up less often. This paper investigates the non-obvious rules.

We obtain estimated energy consumptions of a number of possible communication scenarios in accordance with the standards, and derive conclusions about the energy-optimal configuration of such networks. In particular, we investigate the effects of time-slotted and unslotted medium access techniques, and their interplay with drifting clocks. Our observations allow us to establish rules of the following kind: (1) Unslotted CSMA/CA is more favourable w.r.t. energy saving than slotted CSMA/CA. (2) If devices using GTSs and CSMA/CA coexist, those operating in GTSs expend considerably less energy. (3) Small clock drifts can have far overproportional effects on energy consumption, but with only minor adverse effects on battery lifetimes. For sure, power consumption does not only depend on clock drift and synchronization policy. It also depends on many other factors, such as link quality and other environmental conditions. For our studies, these are assumed constant, since we see no way to include them in our studies without loosing focus. Further, we do not include comparison with experimental measurements. The reason is that (1) controlling physical clock drifts is virtually impossible with available hardware, and (2) real-life experiments would take several months or years to show measurably distinct effects on battery lifetimes. The main contribution of this paper is that it pioneers a model-based analysis of the interplay of clock drift and energy cost in sensor networks.

The paper is organized as follows: Section 2 provides a brief introduction to ZigBee and IEEE 802.15.4. In Section 3, the modelling formalism we employ is described together with the tool chain supporting it. Section 4 describes the modelling of ZigBee and IEEE 802.15.4 in MODEST. We also clarify the modelling assumptions we make and discuss in detail a particular model representative. In Section 5, we describe and discuss the simulation experiments and their results. Section 6 concludes the paper.

2 ZigBee and IEEE 802.15.4

This section provides a general introduction to ZigBee and IEEE 802.15.4, focussed on the characteristics and features that are important for the scope of the paper. For more detailed information, the interested reader is invited to consult the standard documents [3,2].

ZigBee is a wireless communications standard for low-cost and low-power consumption networks. It is a layered architecture. Both physical and medium access control (MAC) layers are defined in IEEE 802.15.4 standard. The ZigBee standard, on the other hand, provides the definition of the Network layer and the framework for the Application layer. ZigBee supports star, tree and mesh network topologies. In all these topologies, a device called ZigBee coordinator controls the network, but per-to-peer communication is allowed in mesh topology.

IEEE 802.15.4 is an open standard for ultra-low complexity, cost, power consumption and low data rate wireless connectivity among inexpensive devices in wireless personal area networks (WPAN) [2]. Devices participating in WPAN can be distinguished into full-function devices (FFD) and reduced-function devices (RFD). Communication between FFDs and RFDs is possible, but RFDs cannot directly communicate with each other. An FFD may become a personal area network (PAN) coordinator, a coordinator, or a device. There are two topologies in which a WPAN can operate: star and peer-to-peer.

The functional characteristic of low-rate PAN can be distinguished into beacon-enabled and nonbeacon-enabled networks. The simplest manner of operation is nonbeacon-enabled, where the network operates by using the (unslotted) Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). CSMA/CA is a multiple access protocol similar to CSMA/CD. As opposed to CSMA/CD, which operates by listening to the channel while sending in order to detect collisions, CSMA/CA tries to avoid collisions by listening to the channel for a predetermined amount of time prior to transmissions.

In beacon-enabled mode, on the other hand, the coordinator periodically emits beacon signals, which provide a frame of reference for a time-slotted access to the medium. More precisely, a so-called superframe structure, which is defined by the coordinator, is used. Inside of the superframe structure, communication can be carried out with a guaranteed time slot (GTS) mechanism or with slotted CSMA/CA mechanisms. Devices using GTS and those using CSMA/CA may coexist, as we will explain below.

The structure of a superframe is shown in Fig. 1. The coordinator broadcasts network beacons regularly. These beacons mark the beginnings and the ends of superframes. The beacons can be used for synchronization purpose, to identify the PAN, to describe the structure of the superframe, as well as to announce the GTS allocations.

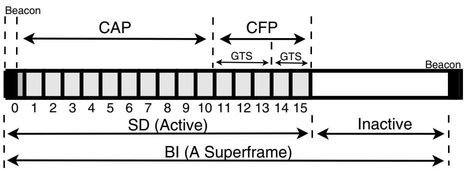

Fig. 1. The Superframe Structure

The coordinator can divide a superframe into an active and an inactive portion. The active portion is further divided into 16 equally sized slots. The coordinator may decide to allocate up to 7 of these slots as GTSs. These GTSs form the Contention-Free Period (CFP) and must appear at the end of the active portions. The rest of the slots forms the Contention-Access Period (CAP), in which devices compete using a slotted CSMA/CA mechanism.

The length of a superframe (Beacon Interval–BI) is determined by the coordinator by varying Beacon Order (BO) which influences the length exponentially:

$$\mathtt{BI} = \mathtt{aBaseSuperframeDuration} \times 2^{\mathtt{BO}}, 0 \leq \mathtt{BO} \leq 14,$$

where $\mathtt{aBaseSuperframeDuration} = 960$ symbols. The duration of active portion (Superframe Duration–SD) is set by varying Superframe Order (SO):

$$\mathtt{SD} = \mathtt{aBaseSuperframeDuration} \times 2^{\mathtt{SO}}, 0 \leq \mathtt{SO} \leq \mathtt{BO} \leq 14.$$

3 MODEST and Supporting Tools

The *Modelling and Description Language for Stochastic and Timed Systems* (MODEST) [8] is a specification formalism for stochastic real-time systems. The language is rooted in classical process algebra, *i.e.* the specification of models is compositional. Basic activities are expressed with atomic actions, more complex behavior with constructs for sequential composition, non-deterministic choice, parallel composition with CSP-style synchronization, looping and exception handling. A special construct exists to describe probabilistic choice. Clocks, variables and random variables are used to describe stochastic real-time aspects.[1]

In order to facilitate the analysis of the different models, tool support is indispensable. The MODEST Tool Environment (MOTOR) is a software tool that implements the MODEST semantics and is the central vehicle in the multi-solution analysis of MODEST models. The by now most mature backend of MOTOR is provided by a link to the MÖBIUS evaluation environment. MÖBIUS has been developed independently from MODEST and MOTOR at the University of Illinois at Urbana-Champaign [14]. From a user perspective, the MOTOR/MÖBIUS tandem enables one to perform simulation of MODEST models, and to gather performance and dependability estimates.

[1] MODEST also supports for modelling time variation, especially time non-determinism, but this feature is not used in this case study.

Simulation-based analysis covers the largest language fragment of MODEST: the only concept that cannot be supported by simulation is non-determinism, in particular of delay durations and non-deterministic choice between actions. We exclude the former by assuming maximal-progress with respect to delays. We do not restrict action non-determinism, since it is a convenient modelling instrument. However, no mechanisms, like a well-specified-check [15], is implemented yet to ensure the validity of the simulation statistics in the presence of action non-determinism.

MOTOR and its connection to MÖBIUS is mature and has been tested in a number of non-trivial case studies. In [18], it has been used for reliability analysis of the upcoming European Train Control System standard. In [9], it has been applied to the analysis of an innovative plug-and-play communication protocol, which has led to a patent application of our industrial partner. In [10], MOTOR has been used for the optimization of production schedules, in combination with timed automata-based schedule synthesis with UPPAAL.

4 Modelling

We model ZigBee/IEEE 802.15.4-compliant personal area networks in a star topology. Each network consists of a single PAN coordinator and a number of stations or devices. We assume that the PAN coordinator has continuous power supply, while the stations do not. A station can be either an FFD or an RFD, attached to a sensor. Periodically, a station communicates with the PAN coordinator, either to transmit its gathered sensor data or to receive instructions, but a station cannot communicate with other stations.

Two separate models, beacon and nonbeacon-enabled PAN models, are built. The simpler, nonbeacon-enabled model is parameterized by `BI`, but this is not used to signal beacons. Instead, `BI` is used to indicate the arrival of messages to each station from its sensor. Each station, in both models, is assumed to always have a message to send: almost one slot-length of data every beacon interval.

In the beacon-enabled PAN model, some of the stations use the CAP for CSMA/CA communications, and some use GTS in the CFP, the detailled scenarios considered are described later. The model is parameterized by `BI`, the beacon interval. We set the superframes to have the same duration as `BI`, hence there is no inactive period. Every `BI`-equivalent time units, the PAN coordinator broadcasts a beacon, and all stations must be ready to receive it. Henceforth, the coordinator is ready to receive transmissions from the stations. The stations which are not assigned any GTSs compete with other similar stations to send their messages by using slotted CSMA/CA, while the stations with assigned GTSs wait for their turns.

To save energy, a station goes to sleep mode whenever it has a chance to. For instance, a station sleeps while waiting for its GTS turn or when performing backoff or whenever it has no messages to transmit. However, a station must always wake up before its turn to transmit or before the beacon is transmitted.

We abstain from modelling the PAN initialization. We instead concentrate on the typical operations of the PAN, when the stations are transmitting messages. The physical layer of IEEE 802.15.4 is not modelled either. For instance, we assume that there is no significant propagation delay and no channel selection procedure. Nevertheless, those physical layer constants which affect the timing of the communication, such as the duration of CCAs (Clear Channel Assessments), are taken into account. Furthermore, we assume the PAN to operate in 2.4 GHz band, hence all devices transmit at 62.5 ksymbols/s. The medium is assumed uniform, in the sense that all participants have complete knowledge of it.

4.1 Models

The model of the PAN in beacon-enabled mode is described in this section, focusing on slotted CSMA/CA, which is more complex and interesting than the GTS case. Complete details are available as a tutorial-style modelling guide [4]. The model consists of two distinct process definitions: `coordinator()` and `station()`, modelling the behaviors of a PAN coordinator and a station, respectively. In all experiments, we set `macMaxCSMABackoff` (the maximum number of backoff attempts before declaring a channel access failure) to 5 and `macMinBE` (the minimum value of the backoff exponent) to 2.

The System. The model of the overall system is depicted in MODEST model 1. The system consists of 11 process instances, one coordinator and ten stations, run in parallel. This is achieved by using the parallel composition construct `par{}`. Processes inside of a parallel composition construct run concurrently and synchronize on their common actions, if existing. The operator `relabel {} by {} p()` relabels actions in the first set by the actions in the second in a particular instance of process `p()`. This allows multiple instantiations of process definition `station()`.

MODEST **model 1** : The complete system

```
par{
 ::coordinator()
 ::relabel {...} by {...} station(1)
 ...
 ::relabel {...} by {...} station(10)
}
```

The Coordinator. A simple model of the PAN coordinator is shown in MODEST model 2, especially to highlight some of the clock manipulation features of MODEST. The coordinator has two clock variables: `btimer` modelling the time progress between beacons, and clock `c` modelling the transmission time of a beacon. In MODEST, clocks increase linearly with time and can only be reset to zero. The coordinator process begins by immediately sending a beacon (action `sendb_start`).All beacons are of length

MODEST **model 2** : The coordinator

```
01 process coordinator() {
02   clock btimer, c;
03   sendb_start {= bintheair=true =} ;
04   when(c==52) sendb_end {= bintheair=false =} ;
05   do{::when(btimer==binterval) sendb_start {= c=0, btimer=0, bintheair=true =} ;
06         when(c==52) sendb_end {= bintheair=false =}
07   }}
```

52μs (lines 04 & 06), namely the duration of the smallest possible beacon. Action `sendb_end` signals the end of the beacon's transmission. From then on, the coordinator waits until clock `btimer` is equal to the value of `binterval` (line 05). At this point, a beacon interval has expired and the coordinator broadcasts a new beacon. The coordinator proceeds thus continuously, broadcasting beacons every time a beacon interval expires.

The process makes use of two global variables, *i.e.* variables accessible to all processes in the model. They are `binterval`, of type integer, which represents the time it takes for a `BI` according to the standard; and `bintheair`, a boolean variable used to indicate to the whole system that a beacon is being transmitted. Delimiters `{= =}` wrap a set of variable assignments. Such assignments are executed atomically at the time instant of the action preceeding them.

The Stations. The model of the station is shown in MODEST model 3 and 4. In the beginning, a station waits until a new beacon is transmitted (line 10). At the same time, it resets its main clock and sets `ttosend` (the duration of the remaining data to send). Once the beacon finishes, the station aligns its backoff boundary with the superframe slot boundary by waiting until a multiple of the backoff period (line 16) expires since the beginning of the previous beacon (line 14). The station then performs backoff and attempts to transmit. This is repetitively done as long as there is still something to send and enough time to do so (line 13), otherwise (line 12) the station just waits for the next beacon.

The maximum length of messages submittable to the physical layer is 133 bytes, which takes 4256 μs to transmit in 2.4 GHz band. Thus the remaining data is split accordingly (line 14). Line 16 is a placeholder for the code for the random selection of variable `r`.

MODEST **model 3** : The station (Part 1)

```
01 process station(int id) {
.. ...
10   do{::when(bintheair) beacon_received {= mainclock=0, ttosend=sendingtime =} ;
11        when(!bintheair) start_waiting ;
12        do{::when(ttosend==0 || !enoughtime) do_nothing {= enoughtime=true =} ; break
13            ::when(ttosend>0 && enoughtime) start_csmaca
14              {= NB=0, CW=2, BE=2, attosend=(ttosend>=4256)?4256:ttosend =} ;
15              when(mainclock%320==0)
16              do{::choose_random {= c=0, r=//Uniform(0,2^BE-1), backofftime=r*320 =} ;
.. ...
```

As shown in MODEST model 4, once the duration of the backoff delay is determined, the backoff is performed only if there is still enough time to complete the backoff together with a CCA before the contention-access period ends (line 19). CCA detection time is equal to 8 symbols period, namely 128 μs in 2.4 GHz band. If there is not enough time, the station stops and waits until the next beacon comes (line 18). A CCA is carried out immediately after the backoff finishes (line 20 & 21). There are three possible outcomes of the CCA: a busy channel, an idle channel with enough time to send the current portion of the message and an idle channel but not enough time. The outcome of a CCA is determined by the value of global variable `sending`. This variable indicates the number of stations currently transmitting. A station about to transmit increases this variable and decreases it again once the transmission finishes.

MODEST **model 4** : The station (Part 2)

```
.. ...
17 alt{
18  ::when(mainclock>=CAP-backofftime-ccatime) {= enoughtime=false =} ; break
19  ::when(mainclock<CAP-backofftime-ccatime)
20    when(c==backofftime) {= c=0 =} ;
21    do{::when(c==ccatime)
22          alt{::when(sending>0)
23                 alt{::when(NB<=maxbackoff) channel_busy {=CW=2,NB=NB+1,BE=(BE<6)?BE+1:6=};break
24                      ::when(NB>maxbackoff) {= restart=true =} ; break }
25               ::when(sending==0 && mainclock<CAP-attosend-960) count_down_CW {= CW=CW-1 =} ;
26                 alt{::when(CW==0) wait_for_boundary ;
27                        when (c==320) send_message_start {= sending+=1, c=0 =} ;
28                        when (c==attosend) send_message_end
29                         {= ttosend-=attosend, sending-=1, restart=true =} ; break
30                      ::when(CW>0) wait_for_boundary ;
31                        when(c==320) {= c=0 =} }
32               :when(sending==0 && mainclock>=CAP-attosend-960)
33                 {= enoughtime=false, restart=true =} ; break }
34    } ; alt{::when(restart) {= restart=false =} ; break
35             ::when(!restart)
36        } } } } } }
```

A busy channel triggers another backoff if the number of backoff so far does not exceed the maximum allowed – `maxbackoff` – (line 22 & 23), otherwise the whole CSMA/CA procedure must be restarted to transmit the current portion of the message (line 24). When the channel is idle but there is not enough time to complete the transmission (line 32) the station escapes the CSMA/CA procedure and waits for the next beacon for further attempts.

In a beacon-enabled PAN, two CCAs are required after backoffs before the transmission of the message. Hence, when the channel is idle and there is enough time to complete the transmission (line 25), the transmission is only commenced if $\mathtt{CW} = 0$ (line 26), namely when two consecutive CCAs find the channel idle. The transmission, which starts at the next backoff boundary, is announced to all other devices by increasing the global variable `sending`, and it takes the amount of time to send the portion of the message, namely `attosend`.

Once the portion of the message is transmitted, the station decreases the variable `sending` and updates the remaining portions of the message to transmit by changing `ttosend` (line 29). Henceforth, the station restarts the CSMA/CA procedure to transmit the remaining message, without waiting for a new beacon.

4.2 Energy Consumption

In modelling the energy consumption of a station, we use the technical specification of CC2420 [1]. CC2420 is a 2.4 GHz ZigBee/IEEE 802.15.4-compliant radio frequency (RF) transceiver produced by Chipcon AS. Fig. 2 summarizes the specification relevant for this paper. During its operation, the transceiver can be in four modes: $s$hutdown, $i$dle, $t$ransmitting or $r$eceiving modes. The rate of energy consumption while occupying these modes [11] are 0.144, 712, 30672 and 35280 μW, respectively. A small amount of energy is required during the shutdown mode to power clock and to witness power-ups.

Furthermore, the transitions from mode s to i, from mode i to t, and from mode i to r do not happen instantaneously, but take 970, 194 and 194 μs to complete, respectively. During these transitions [11], the transceiver is considered to be still in the original mode, while consuming energy at the level of the destination mode. This reflects that the transceiver requires time (and power) to turn on its transmitting and receiving devices. All other transitions, namely those with dotted edges in the figure, take no time to complete.

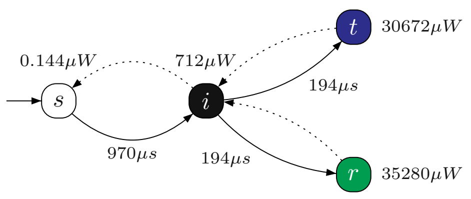

Fig. 2. Energy Modes of CC2420

When the PAN coordinator broadcasts a beacon, all stations must be in receiving mode. That means, some time beforehand, they must have woken up and proceeded to the idle and then to the receiving mode. If a station is assigned GTSs, it immediately changes to shutdown mode upon the completion of the beacon. However, the station must already be in transmitting mode, when its assigned GTSs begin. To anticipate this, the station must leave the shutdown mode 1164 (namely 970+194) μs before the GTSs. After finishing the transmission in the GTSs, the station moves to idle mode. If there is still ‘enough’ time before the next beacon, it changes further to shutdown mode.

A station without assigned GTSs must compete with similar stations by using slotted or unslotted CSMA/CA to gain access to the medium. Immediately after the end of a beacon, the station enters the CSMA/CA procedure. Depending on the duration of the backoff delay, the station may transition to idle or shutdown mode. If the backoff delay is long enough, the station may sleep and wake up 1164 μs before it must perform a CCA, and be ready in receiving mode. If two consecutive CCAs result in idle channel, the station changes to transmitting mode and sends the message portion. This is performed continuously until the whole message is sent. Afterwards, the station may move to shutdown mode if there is enough time to do so before the next beacon.

To incorporate the energy consumption to the PAN models, each instant of station changes variables `time_in_s_mode`, `time_in_i_mode`, `time_in_t_mode` and `time_in_r_mode` every time it spends some portions of time in s, i, t, and respectively r mode. The framework models are annotated accordingly and the four variables are turned into four global variable arrays, that each instant of station accesses through its `id`. The amount of energy consumed by a station during a beacon interval is the sum of the amount of time it spends in each mode weighted by the rate of the energy consumption of the mode.

Referring back to MODEST models 3 and 4, the background color of each line number of the model corresponds to the energy mode as depicted in Fig. 2. Those line numbers with two background colors indicates the case where the station is deciding whether to enter idle or shutdown mode based on the progress of its main clock so far.

4.3 Clock Precision

Clock precision is important to ensure the correct functioning of ZigBee/IEEE 802.15.4-compliant PANs. In the case of PANs which provide CFPs, it is crucial that stations with assigned GTSs transmit exactly in their allocated period of time. Similarly with PANs employing the slotted CSMA/CA mechanism, the ability to correctly determine the backoff boundaries, which requires a precise clock, is necessary to avoid collisions.

Quarz-based clocks are generally used for electronic devices. Such clocks suffer from inaccuracy due to aging and temperature variations. Usually, the manufacturers guarantee a certain upper bound inaccuracy for their clock products. Assuming this upper bound to be constant is actually not realistic. However, we do so in our models. Hence, a clock with a guaranteed accuracy may deviate from the real time within a given time interval based on the accuracy and exhibit at different times a different deviation from the real time.

Clock inaccuracy is usually expressed in 'parts per million' (ppm), referring to the maximum difference (in time units) one may witness relative to perfect time, within a million time units. In the models, we assume that a physical clock with a guaranteed accuracy of p ppm may show at time t, a clock value within the interval:

$$[t - p \cdot 10^{-6} \cdot t, t + p \cdot 10^{-6} \cdot t].$$

A station, however, does not know the exact value of the deviation. Nevertheless, it must be able to precisely observe the deadlines, for instance the arrival of a beacon. Therefore, it counts on the maximum inaccuracy and be ready for the beacon even before the actual time. It is also assumed that the stations synchronize their clocks to the PAN coordinator's, while receiving the beacons.

We model the effect of clock inaccuracy to energy consumption in the following way. Assume that a station must wait for W time units to be ready for some event. The clock of the station has inaccuracy p ppm. The actual waiting time W' for the station is:

$$W' = W - W \cdot p \cdot 10^{-6} + W \cdot p' \cdot 10^{-6}, \tag{1}$$

where $W \cdot p \cdot 10^{-6}$ is the maximum deviation of the clock given the clock inaccuracy and $W \cdot p' \cdot 10^{-6}$ is the actual deviation. The actual inaccuracy p' is a value in the interval $[-p, p]$. The models can be parametrized by the actual deviation. In full generality, a clock's inaccuracy may be time-dependent, in which case the above formula involves integration.

5 Simulations

In this section, we describe the simulations of the MODEST models presented in the previous section. The simulation was done with discrete-event simulator of MÖBIUS, and we only present an excerpt of several thousand simulation runs we performed. First, the experimental setups are described. The result of the experiments is presented afterwards, followed by its analyses and discussions.

Experiments. In all experiments, the system is a personal area network, which consists of a single PAN coordinator and 10 stations with star topology. The network uses 2.4 GHz band and all durations appearing in the standard which are defined by amount of symbols are adapted accordingly. The networks do not use any inactive periods, thus the beacon interval is always the same as the superframe duration (BI=SD).

The experiments cover BO $= 0, \cdots, 10$, which means a beacon interval ranges from 15360 μs to 15728640 μs. Similarly the length of a slot ranges from 960 μs to

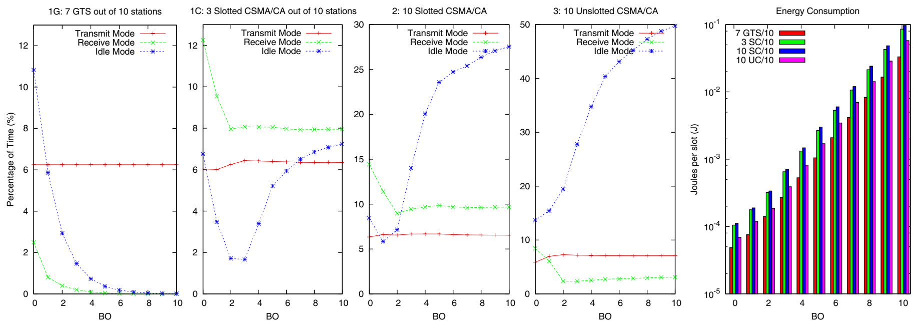

Fig. 3. Percentage of time per energy mode and energy consumption for perfect clocks

983040 μs. In all experiments, a station always has about a slot-length message to transmit during the duration between two beacons. A station using GTS mode can transmit the whole message continuously. On the other hand, a station using slotted or unslotted CSMA/CA can only transmit 133 bytes at a time, which takes 4256 μs to transmit.

The simulation time of each experiment spans 10 beacon intervals. For instance, when $\texttt{BO} = 7$, a beacon interval takes 1966080 μs. Therefore, the simulation is run for around 20 seconds (19660800 μs). Simulations are carried out to estimate the mean values of some measures of interest, such as the time a station spends in shutdown mode. The simulation is repeated until the mean values of all measures of interest converge with relative confidence interval 0.1 and confidence level 95%. MÖBIUS allows users to adjust these settings as desired.

As mentioned earlier, we consider worst-case scenarios whenever appropriate. For clock inaccuracy, the worst scenarios occur when the clock of the PAN coordinator is progressing as slow as possible within its inaccuracy bounds, while the clock of a station is in its fastest possible progress. Referring to Eqn. 1, the clock of a station is fastest when $p' = -p$. Then the station, which needs to wake up some time before the actual deadlines, actually wakes up even earlier because its clock is too fast. We proved this situation to be worst case by analytical means, which is backed up by simulation.

All simulations are conducted on a PC with a Pentium 4 3.0 GHz processor with 1 GB RAM running Linux 2.6.17-2.686. The CPU time per experiment series ranges from around 5.2 seconds to 15429 seconds (about 4.25 hours).

Results. The graphs in Fig. 3 summarize our experimental results when clocks are perfect. The leftmost graph shows the percentage of the time spent in transmit, receive and idle modes for the 7 stations assigned with GTSs in experiment series 1 (we denote this as series 1G). While the portion of time spent in transmit mode is constant for all BO's, the portions in receive and idle modes diminish as the beacon intervals get larger, and become less than 1% after $\texttt{BO} = 5$. This is because the actual time spent in receive and idle modes remains constant for all beacon intervals, thus their percentages decrease as the beacon intervals get longer. They are constant because a station only goes to receive mode while receiving beacons and to idle mode prior to the arrivals of beacons and before the transmissions of the messages.

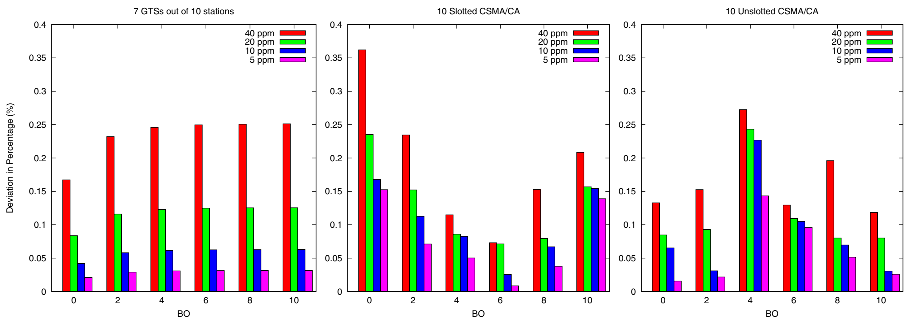

Fig. 4. Increase of energy consumptions when clocks drift, relative to perfect clocks

The second graph in the figure depicts the same percentage distribution for the 3 stations using slotted CSMA/CA in experiment series 1 (denoted as series 1C). The portion of time spent in the transmit mode remains almost constant and around 0.25% higher throughout than the previous case. The percentage of time spent in receive mode is also almost constant at around 8%, which is higher than in series 1G. The percentage increase in receive mode is due to the CCAs, and since the number of CCAs to be performed is proportional to the length of the message, the percentage is steady for most BO values. The time spent in the idle mode, on the other hand, increases with the length of beacon intervals. The longer the beacon interval, the longer a slot-length message. However, since the length of message portion transmittable is restricted, the more often slotted CSMA/CA procedure is repeated to send the whole message. Hence the station spends more percentage of time in idle mode.

The third and fourth graphs in the figure show similar percentage distribution for experiment series 2 and 3, namely 10 stations using slotted and unslotted CSMA/CA, respectively. Compared to series 1C, the percentage of time spent in receive mode is higher in series 2. This is due to the fact that there are 10 stations for 15 CAP slots in series 2 compared to 3 stations for 8 CAP slots in series 1C. Thus stations perform more CCAs and spend more percentage of time in receive mode. Similar reason applies to the percentage of idle mode. In series 3, however, the percentage of time spent in receive mode is lower than in series 1C and 2 because the stations perform only one CCA in every CSMA/CA cycle. Between themselves, the stations in experiment series 3 spend more time in idle mode compared to those in series 2. This can be explained by the absence of beacons in unslotted CSMA/CA. Once a station finishes transmitting the whole message, it has no estimation when the next message will be available and thus must stay idle. The station enters the shutdown mode when performing backoff.

The rightmost graph in Fig. 3 depicts the amount of energy consumed by a station during a beacon interval time for each value of BO and for each of the formerly described series. Note that the y-axis of the graph is in logarithmic scale. The largest energy consumption is around 0.096946940587 Joules per beacon interval when $\mathtt{BO} = 10$. We can observe that for all values of BO, a station using GTSs consumes the least amount of energy. A station using the unslotted CSMA/CA consumes more energy that that using GTSs but less than the energy consumption of a station using slotted CSMA/CA.

A question arises as to why the energy consumption of a station in series 1C is lower than that in series 2, considering the fact that the former has more CAP-slots per station. The reason is in series 1C, a station spends more time in shutdown mode, namely during the CFP period. This can also be observed from the second graph in the figure. It is also worth noting that the differences between the energy consumptions amongst the series are significant, even though they are not apparent due the logarithmic scale. For instance, a station in series 2 consumes almost thrice the energy of a station in series 1G and almost twice of a station in series 3.

Fig. 4 shows the influence of worst-case clock drifts on the energy consumptions of stations in series 1G, 2 and 3, respectively. Consistently, clock drifts increase the amount of energy consumed for all experiments. The graphs in the figure show the increase in energy expended relative to the amount of energy needed in the precise clock setting. The first graph depicts the deviations for a station in series 1G with clock inaccuracies 40, 20, 10 and 5 ppm for $\texttt{BO} = 0, 2, 4, 6, 8, 10$. We observe a linear correlation between the clock inaccuracies and the deviations in the graph. For instance, for most of BO values, a station with clock inaccuracy 40 and 20 ppm spends approximately 0.25% and respectively 0.125% more energy than a station with precise clock. The second and third graphs in Fig. 4 depicts the deviations for a station in series 2 and 3, respectively. We abstain from explaining the precise patterns exhibited by these plots for increasing BO due to space constraints. However, the effects of the level of clock inaccuracies are obvious. In both graphs, higher clock inaccuracy tends to cause higher energy consumption level.

Overall, a deviation of 0.25% or 0.125% may seem small. If extrapolated, it amounts to about half a day per year decrease in lifetime of a battery. However, a clock drift of 40 ppm is – percentwise – only 0.004%, and we thus conclude that such a clock drift may be amplified by a factor of 62.5 when looking at the excess in energy consumption caused.

6 Conclusion

This paper has advocated a rigorous approach to modelling and simulation of energy-aware networked embedded systems. In this approach, MODEST, a language with a strictly formal semantics is used to specify the system under study. Since all assumptions are exposed explicitly in the specification, the underlying stochastic model is well-defined and the obtained simulation results are trustworthy. This is different from many other simulation-based results lately published in the networking literature. The simulation is carried out via the tools MOTOR and MÖBIUS. MOTOR is available as source code from `http://www.purl.org/net/motor` under the GPL license. MÖBIUS is freely available for educational and research purposes from `http://www.mobius.uiuc.edu/`. MOTOR can be installed as an add-on package to the MÖBIUS installation.

We applied the proposed approach to the modelling of crucial medium access mechanisms in IEEE 802.15.4, with a particular emphasis on energy specific configuration relevant to ZigBee operation. When investigating the effects of time-slotted (*i.e.*, beacon-enabled) and unslotted medium access techniques, we observed that slotted

CSMA/CA is clearly inferior to unslotted CSMA/CA w.r.t. energy efficiency. In slotted mode, devices using GTSs use considerably less energy than those using CSMA/CA.

Our studies of drifting clocks reveal some interesting observations. Most of them are consistent with what one might expect – but we did not see them published elsewhere. However interesting, they only show minor quantitative effects. The IEEE standard limits the allowed clock drift to 40 ppm, and this appears to be caused by correctness concerns. (One can show that with 64 ppm the clock boundaries could drift apart to an extent that the protocol could desynchronize and thus malfunction). In the worst case we estimate that a clock drift of 40 ppm induces a shortage in battery life of about half a day per year. While this amplifies the clock drift by a factor of about 60, we think it is not particularly significant, leading us to finally negate the question in the title of this paper.

References

1. CC2420 Product Information. Chipcon AS (2005), http://www.chipcon.com/index.cfm?kat_id=2&subkat_id=12&dok_id=115
2. IEEE 802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (2003)
3. ZigBee Specification Version 1.0. ZigBee Alliance (2004), http://www.zigbee.org/en/spec_download/download_request.asp
4. MoDeST Tutorial: development of a complex model in MoDeST. Dependable Systems and Software, Saarland University, Germany (2006), http://depend.cs.uni-sb.de/modesttutorial/index.html
5. The network simulator – ns-2 website (2007), http://www.isi.edu/nsnam/ns/
6. Andel, T.R., Yasinac, A.: On the credibility of Manet simulations. IEEE Computer 39(7), 48–54 (2006)
7. Bengtsson, J., Larsen, K.G., Larsson, F., Pettersson, P., Yi, W.: Uppaal – a tool suite for automatic verification of real-time systems. In: Hybrid Systems III, pp. 232–243. Springer, Heidelberg (1995)
8. Bohnenkamp, H.C., D'Argenio, P.R., Hermanns, H., Katoen, J.-P.: MoDeST: A compositional modeling formalism for hard and softly timed systems. IEEE Trans. Soft. Eng. 32(10), 812–830 (2006)
9. Bohnenkamp, H.C., Gorter, J., Guidi, J., Katoen, J.-P.: Are you still there? - A lightweight algorithm to monitor node presence in self-configuring networks. In: DSN 2005, pp. 704–709. IEEE CS Press, Los Alamitos (2005)
10. Bohnenkamp, H.C., Hermanns, H., Klaren, R., Mader, A., Usenko, Y.S.: Synthesis and stochastic assessment of schedules for lacquer production. In: QEST 2004, pp. 28–37. IEEE CS Press, Los Alamitos (2004)
11. Bougard, B., Catthoor, F., Daly, D.C., Chandrakasan, A., Dehaene, W.: Energy efficiency of the IEEE 802.15.4 standard in dense wireless microsensor networks: Modeling and improvement perspectives. In: DATE 2005, pp. 196–201. IEEE CS Press, Los Alamitos (2005)
12. Cadilhac, M., Hérault, T., Lassaigne, R., Peyronnet, S., Tixeuil, S.: Evaluating complex MAC protocols for sensor networks with APMC. Elect. Notes Theor. Comput. Sci. 185, 33–46 (2007)
13. Cavin, D., Sasson, Y., Schiper, A.: On the accuracy of MANET simulators. In: POMC 2002, pp. 38–43. ACM Press, New York (2002)

14. Daly, D., Deavours, D.D., Doyle, J.M., Webster, P.G., Sanders, W.H.: Möbius: An extensible tool for performance and dependability modeling. In: Haverkort, B., Bohnenkamp, H.C., Smith, C.U. (eds.) TOOLS 2000. LNCS, vol. 1786, pp. 332–336. Springer, Heidelberg (2000)
15. Deavours, D.D., Sanders, W.H.: An efficient well-specified check. In: PNPM 1999, pp. 124–133. IEEE CS Press, Los Alamitos (1999)
16. Fruth, M.: Probabilistic model checking of contention resolution in the IEEE 802.15.4 low-rate wireless Personal Area Network protocol. In: ISoLA 2006 (2006)
17. Garavel, H., Lang, F., Mateescu, R.: An overview of CADP 2001. EASST Newsletter 4, 13–24 (2001)
18. Hermanns, H., Jansen, D.N., Usenko, Y.S.: From StoCharts to MoDeST: a comparative reliability analysis of train radio communications. In: WOSP 2005, pp. 13–23. ACM Press, New York (2005)
19. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM: Probabilistic symbolic model checker. In: Field, T., Harrison, P.G., Bradley, J., Harder, U. (eds.) TOOLS 2002. LNCS, vol. 2324, pp. 200–204. Springer, Heidelberg (2002)
20. Landsiedel, O., Wehrle, K., Götz, S.: Accurate prediction of power consumption in sensor networks. In: EmNetS-II, pp. 37–44 (2005)
21. Pongor, G.: OMNeT: Objective modular network testbed. In: MASCOTS 1993, pp. 323–326 (1993)
22. Titzer, B.L., Lee, D.K., Palsberg, J.: Avrora: Scalable sensor network simulation with precise timing. In: IPSN 2005, pp. 477–482 (2005)
23. Zeng, X., Bagrodia, R., Gerla, M.: GloMoSim: A library for parallel simulation of large-scale wireless networks. In: WPDS 1998, pp. 154–161 (1998)