Estimating Node Lifetime in Interference Environments

Alex King, James Brown, John Vidler and Utz Roedig
School of Computing and Communications
Lancaster University, Lancaster, United Kingdom
{a.king, j.brown, j.vidler, u.roedig}@lancaster.ac.uk

***Abstract*—For commercial Wireless Sensor Network (WSNs) deployments it is necessary to estimate the network lifetime. It must be possible before network deployment to determine how long a network maintains operational before maintenance is required and batteries have to be replaced. Unfortunately, node lifetime is very dependent on the radio environment in which the node is operated. As we will demonstrate in this paper the node lifetime in a very busy radio environment can be up to 11 times shorter than in a quiet environment. WSNs employ duty-cycled communication protocols where receivers periodically sample the channel to determine if it has to remain active to receive a message. Radio interference triggers the receive mechanism causing an unnecessary wake-up which leads to an increase in a node's energy consumption. In this paper we present a method for estimating node energy consumption in a target radio environment. We describe how to capture the essential characteristics of the radio environment and how to use this information to predict node lifetime. We demonstrate the usability of the proposed method using the well known WSN communication protocol ContikiMAC. Our evaluation comprising real-world scenarios shows that the proposed method is able to accurately predict node lifetime.**

I. Introduction

In commercial *Wireless Sensor Network* deployments, it is often necessary to estimate the power consumption of nodes before devices are deployed. For example, it is necessary to estimate power consumption of nodes to calculate the dimensions of power harvesting devices - such as solar panels - or to plan maintenance cycles for battery replacement. Through this, predicted energy consumption decides, in many cases, if a WSN application is commercially feasible or not. Thus, it is very important to have methods and tools which allow us to forecast energy consumption accurately.

The dominant energy consumer in most WSN nodes is the radio transceiver, so naturally it is important to reduce radio transceiver usage as much as possible in order to conserve energy. A vast number of WSN communication protocols have been developed in recent years aiming at this goal through *duty cycling*.

Duty-cycled Medium Access Control (MAC) protocols are employed and nodes periodically alter transceiver operation between an energy efficient sleep state and an energy costly listen state. The transmitter sends a message such that it coincides with the listening phase of the receiver. The frequency and duration of the transceiver's listen state dominate the energy consumption of the node.

State of the art WSN MAC protocols such as Contiki's ContikiMAC or TinyOS's LPL minimise this costly listening phase by performing short channel busy checks which are prolonged when transmission activity is recognised. Unfortunately, interference from other wireless devices in the same frequency space (such as WiFi) also cause an active channel event and triggers an extension of the listen phase, as the interference is indistinguishable from real traffic using this mechanism. Consequently, energy consumption of a listening node is dependent on the interference present in a nodes deployment location, and furthermore as we will demonstrate, a nodes energy consumption in a very busy radio environment can be up to 11 times higher than in a quiet environment.

As the feasibility of a WSN application depends on the achievable node lifetime, and the energy consumption of a node depends on the interference patterns, we were able to develop a generic method for capturing interference characteristics and predicting a node's energy consumption. Interference is captured for a period of time in the target deployment area, then, using a mathematical model it is possible to estimate the energy consumption of the node in this deployment scenario.

However, deriving a model for a MAC protocol as a closed-form solution is both difficult and inflexible when considering a variety of MAC protocols. To overcome this we describe a Monte Carol solver which can easily be programmed to incorporate various MAC protocol specifics, and allows accurate energy consumption forecasting in noisy environments.

This paper describes how to model the energy related aspects of a MAC protocol and how to predict energy consumption. The prediction method is evaluated using the well known MAC protocol ContikiMAC, and the accuracy of predictions is demonstrated by analysing energy consumption of real nodes in a variety of radio environments.

The specific contributions of this paper are:

- *Energy Model:* We describe how a closed form solution for energy consumption prediction based on interference measurements can be derived. We describe a closed form solution for Contiki's ContikiMAC protocol.
- *Energy Solver:* We describe a Monte Carlo solver for energy consumption prediction based on interference measurement. We show that the solver is able to provide accurate results while providing more flexibility than a closed form solution.

- *Evaluation:* We describe a set of experiments carried out in different interference environments. Experiments show that lifetime prediction with an accuracy of up to 2% is possible.

In the following section we discuss related work. Section III describes in detail low-power MAC protocols, and how their energy consumption is dependant on interference levels. In particular we describe ContikiMAC used in this study as an example protocol. Section IV discusses how interference patterns in a deployment area can be captured. Section V shows how a closed form solution and Monte Carlo solver can be constructed to predict energy consumption in an interference environment. Section VI describes the evaluation of the proposed estimation method. Section VII concludes the paper.

II. Related Work

Reducing energy consumption has featured frequently in wireless sensor network research, alongside methods to detect and classify local interference in deployments. To our knowledge, little work has been carried out on adjoining these two domains: estimating energy consumption with respect to interference. In the following section, we discuss related work in both areas.

Simulation of sensor network hardware has proven a viable method for estimating energy consumption before deployment [1], [2], [3]. The authors in [1] extend TOSSIM [4], a sensor network simulator, to measure energy consumption and estimate battery life. The simulator, restricted to TinyOS [5] runs code native to the host machine, estimating execution time for each block of code. Conversely, [2] and [3] emulate hardware directly, improving accuracy at the cost of execution time.

Simulation of heterogeneous interference and its impact on neighboring sensor networks is a prerequisite to accurate energy consumption in WSNs, that is missing from these works. TiQ [6] is an extension to a network simulator which supports heterogeneous networks, hence exceeding this limitation. Energy consumption however, is not measured, and the authors only briefly discuss the accuracy of their results in this context.

Simulating networks offline has advantages over testbed experiments, including repeatability, lower costs and scalability. However, estimations made off-line can quickly become invalid in a changing environment. Our on-line approach works in conjunction with WSN applications, able to provide continual predictions on future energy use based on channel conditions.

Online estimation techniques are used during deployment to monitor energy usage, for example to curtail energy use to meet certain lifetime goals [7]. Dunkels et al. [8] implement an on-line energy estimation technique in Contiki [9]. These approaches measure the state of components, such as radio, sensors and LEDs, to calculate consumption. Our technique predicts the energy use of one component: the radio, based on MAC protocol paramaters, and converges over a much shorter period.

Classifying local interference is useful for channel selection [10] and tuning interference mitigation techniques. Sampling the channel energy over time across a single channel [11], or multiple adjacent channels [10], [12], is a vaiable method for classification, by searching for known interference signatures. Periodically sampling the channel is itself costly in terms of energy. The authors in [13] instead glean information about local interferers from standard network operation, taking longer to classify interence while reducing energy consumption.

Our work does not aim to mitigate interference, rather enable accurate prediction of node energy consumption in the presence of heterogeneous interference sources.

III. Low-Power Medium Access Control

The greatest source of a node's energy consumption in typical WSN deployments is the radio, therefore it is necessary to keep the radio switched off as much as possible. However, to ensure a node can still receive messages the radio must be periodically reactivated to check for incoming transmissions.

These periodic checks are costly and dominate energy cost in a network with low traffic volume and it is therefore the aim to keep these checks brief. To shorten the radio on time for such checks, state of the art MAC protocols use the radio's Clear Channel Assessment (CCA) feature. With CCA the energy present on the channel is determined which can be used to estimate if another node is sending or not. To execute a CCA only the time required to power up the transceiver and to read the CCA value is required, and only if channel activity is detected is further time (and therefore energy) invested to receive and decode the entire transmission. This procedure is much more efficient than, for example, putting the radio periodically in listen mode and waiting for it to lock onto an incoming transmission.

In this work we consider ContikiMAC [14], a well known MAC protocol used with the Contiki OS which follows the aforementioned scheme. We discuss in detail how this protocol uses CCA and how the mechanism relates to energy consumption. In this work we use the *duty cycle*, the percentage of time a node's radio is active, as energy consumption metric. This is a valid approach as state of the art WSN radios use nearly the same energy in all active modes of operation (send, receive, listen).

A. ContikiMAC

ContikiMAC is designed for IEEE 802.15.4 radios such as the commonly used CC2420. The basic operation of ContikiMAC is illustrated in Figure 1. A sender repeats transmission of a data packet until a receiver acknowledges reception of one of the transmissions with an acknowledgement (ACK). Repeated transmission of data packets is necessary to ensure synchronisation between sender and receiver. A receiver periodically checks the channel for transmissions using two CCA operations separated apart (typically by $500\mu s$). In its default setting ContikiMAC uses a Channel Check Rate (CCR)

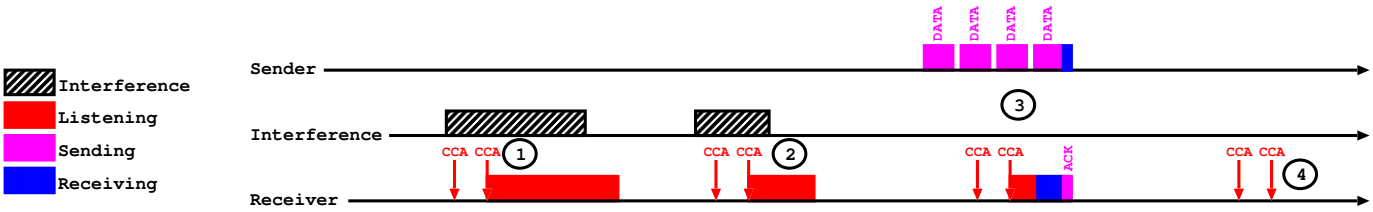

Figure 1: ContikiMAC's sending and receiving behaviour under interference. In (1) and (2) the receiver interprets interference as incoming transmission and prolongs unnecessarily the energy costly listen phase. In (3) the receiver deducts correctly from the CCA that a packet trail is transmitted and the packet is received. In (4) the receiver carries out a periodic CCA without presence of transmission or interference.

Variable	Tmote Default
$T_1 = T_2$	$294\mu S$
T_3	$122\mu S$
T_W	$500\mu S$
N_{MAX}	10
N_{SIL}	5

Table I: The default ContikiMAC variable values as used on the Tmote Sky platform.

of $8Hz$. Two CCA operations are necessary to ensure that an incoming transmission is reliably detected as one CCA may fall in between repeat transmissions. Depending on the outcome of the two CCA checks a receiver executes different steps necessary to receive the transmitted packet. We describe these steps in detail in the next paragraph.

As shown in Figure 1 the reception mechanism can either be triggered by legitimate senders (see (3) in Figure 1). However, a present interference signal will trigger as well these energy costly mechanisms (see (1) and (2) in Figure 1).

B. ContikiMAC Channel Check

ContikiMAC's channel check procedure can be described in detail using a state machine as shown in Figure 2. The receiver wakes, starts the radio and then carries out the first of the double CCA checks (CCA1) as shown in Figure 1. CCA1 is denoted in grey in Figure 2 as it contributes to the energy consumption of the node. CCA1 requires T_1 which comprises the time necessary to start the radio and the time required to take the CCA sample. If CCA1 returns an idle channel the radio is set to power down to sleep for T_w. After T_w the second CCA, CCA2 is carried out. CCA2 again contributes to the node's energy budget; the radio is powered up and a CCA sample is taken requiring T_2 radio on time ($T_2 = T_1$). If CCA2 returns idle the radio is turned off as no incoming transmission was detected and the node will wake again for the next channel check and run again through this state machine. If either CCA1 or CCA2 have detected a busy channel the radio is kept in an active state and the node attempts to receive the message. Periodically a further CCA check is carried out (CCA3) to check if the transmission is still ongoing. CCA3 requires a shorter duration T_3 than CCA1 or CCA2 as the radio is kept active and no radio startup time is required. Between subsequent CCA3 checks a delay of T_w is used. As the radio is active to receive data, this time contributes to the energy budget of the node. Two variables N_{sil} and N_{max} control for how long the listen period is extended to receive a transmission. N_{sil} is used to specify a maximum number of idle (silent) CCA3 checks can be observed before the radio is set to sleep. N_{max} is used to specify the maximum number of CCA3 checks that should be performed before the radio returns to sleep. Table I shows all parameters defining the ContikiMAC channel check procedure and lists parameter values for the Tmote Sky platform with CC2420 radio as used in our evaluation.

Obviously the time the radio is active during ContikiMAC's channel check depends strongly on the success or failure of the various CCA. These in turn depend on interference which might be present in the channel.

IV. Interference Representation

To estimate the impact of interference on energy consumption it is necessary to capture interference patterns in a deployment area in a sensible way. Capturing interference is challenging as interference is dynamic and can change over time. It is also not guaranteed that an interference pattern measured in one place is the same at other (nearby) locations. However, recent studies have shown that interference patterns are, in many scenarios, stable over long periods and are very similar in nearby locations (see [15]). In particular this is true in scenarios that lack mobility as found with most current WSN applications. Thus, it is feasible to capture an interference pattern at a deployment site and to use this pattern to estimate the energy consumption of a node deployed at this location in the future. In particular, as we aim to use interference patterns to estimate energy consumption and ultimately node lifetime (in the order of years) short term interference changes (spikes) are not of a concern. In our experiments we show that these assumptions hold and that it is indeed feasible to measure interference patterns in a deployment area and to use this information for prediction.

For the purpose of this work interference is only of a concern if it lies above the CCA threshold. As described, a node uses the CCA mechanism to determine if the channel is busy or not. The channel energy is measured and if it is above the energy threshold the CCA mechanisms returns with a negative result. Thus, we can for the purpose of this work represent interference as a binary signal, indicating only if interference is below or above the CCA threshold.

It would be possible to collect an interference trace (as binary signal) in the deployment area over a longer time period

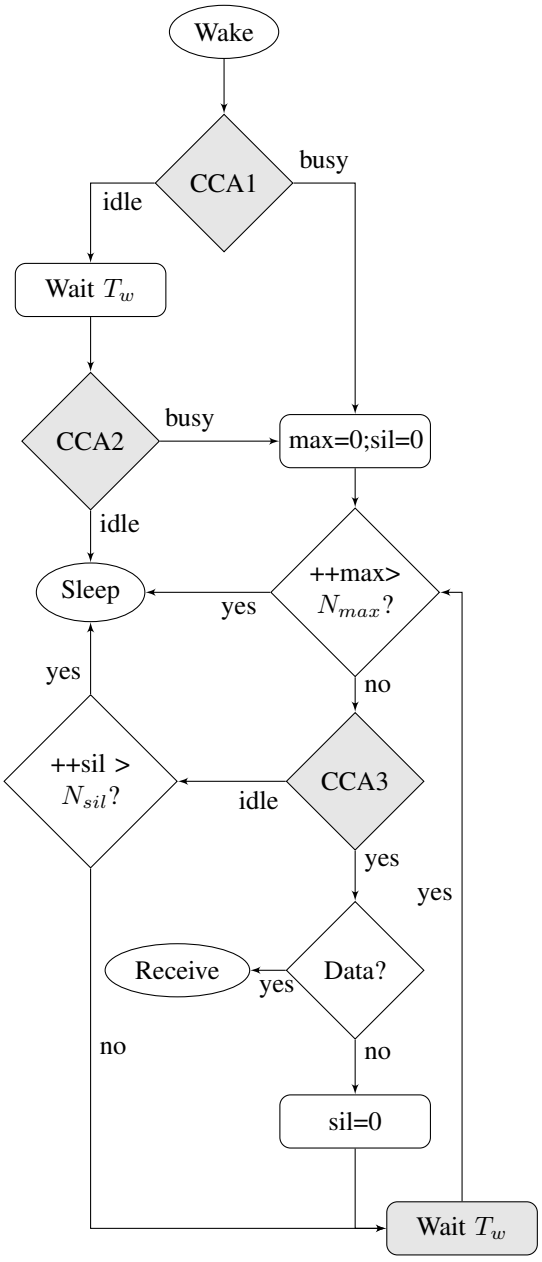

Figure 2: Flowchart of ContikiMAC's channel check procedure. Elements contributing to the idle duty cycle are shaded gray.

and to use this trace to evaluate the channel check performance of a MAC protocol in this environment. If we assume the captured interference trace is representative for the deployment area this would give provide a valid energy consumption estimation. However, capturing such an interference trace is memory intensive and thus not feasible if sensor nodes themselves are used to capture the interference characteristic. To avoid this complexity it is possible to simply record statistical properties of the interference signal and to use these properties to evaluate protocol performance. For example, it is possible to simply record the distribution of idle period lengths (durations without interference signal above the CCA threshold) and busy period lengths (see [15] for further details). A further reduction of detail is possible by aggregating this data and simply representing the interference pattern as the amount of time the channel is observed to be busy (interference signal observed above the CCA threshold). In this work we use this coarse-grained representation of an interference pattern; we use p to denote the probability of the channel being observed busy due to an interference signal above the CCA threshold. It is very easy to obtain this *channel busy probability* p via measurements with sensor node hardware and as we will show in our experimental evaluation, this interference representation is sufficiently detailed to evaluate MAC protocol performance. We show that using this representation, accurate energy consumption prediction in practical WiFi interference environments is possible.

V. Estimating Energy Consumption

Using ContikiMAC as an example we show how a closed form solution can be derived for a MAC protocol to estimate energy consumption based on a recorded interference pattern. As previously outlined, interference is described as channel busy probability p and we use the duty cycle (denoted D), the percentage of time a node's radio is active, to measure energy consumption.

As the development of a closed form solutions is very complex and has to be repeated for every new MAC protocol (and even every variation of a MAC protocol) we describe as well a generic Monte Carlo Solver that can be easily adapted to different MAC variations. As we show, a solver provides more flexibility while providing sufficiently accurate results.

A. Closed Form Solution

To give a closed form solution for energy consumption it is necessary to calculate the expected radio receiver on time $E(p)$ for a MAC protocol's channel check sequence. E is a function of the parameter p which denotes the probability of the channel being busy. p is known via measurements in the deployment area as described previously. If it is known how many channel checks a MAC protocol performs per unit of time we can calculate the duty cycle $D(p)$. Using $D(p)$ it is possible to calculate energy consumption as it is known how much energy the radio consumes when it is active. If we assume that the MAC protocol performs periodic channel checks with a frequency of f the duty cycle $D(p)$ is given as:

$$D(p) = E(p) \cdot f \tag{1}$$

$E(p)$ can be calculated by taking all possible variations of the of the channel check sequence into account. We now show this calculation for ContikiMAC as an example. $E(p)$ for ContikiMAC is composed of three main elements, E_{ii}, E_{ib} and E_b, which correspond to different branches in the ContikiMAC channel check state machine as shown in Figure 2:

$$E(p) = E_{ii} + E_b + E_{ib} \tag{2}$$

E_{ii} represents the case where ContikiMAC's first CCA (CCA1) returns clear followed by a clear result from the second CCA (CCA2). The probability for this branch of the state machine being executed is $P_{ii} = (1-p)^2$. If this path is executed the radio receiver on-time is the time required to execute the two CCA with duration T_1 and T_2. Thus, the term E_{ii} is given as:

$$E_{ii}(p) = (1-p)^2 \cdot ((T_1 + T_2)) \tag{3}$$

E_b represents the case where ContikiMAC's first CCA returns busy and ContikiMAC enters a procedure in which the channel is periodically checked via a CCA (CCA3 with duration T_3). In this procedure the node evaluates if a detected channel activity is part of an incoming transmission. A maximum number of $N_{max}+1$ CCA's are carried out, with $N_{max} = 10$ for a default ContikiMAC configuration. The procedure may terminate before $N_{max}+1$ CCA's are carried out if $N_{sil}+1$ consecutive clear CCA's are encountered, with $N_{sil} = 5$ for a default ContikiMAC configuration. Between each CCA a delay of T_w is included, which contributes to the radio on time as ContikiMAC keeps the radio active during the entire procedure. The very first CCA in this procedure returns always busy as there is no time delay between this CCA and the busy CCA leading into this procedure.

For the default ContikiMAC configuration, as given in Table I, exactly 7 possibilities exist for the procedure to terminate before the maximum number of $N_{max}+1 = 11$ CCA checks are carried out. For example, after the first CCA in the procedure – which always returns busy – we could encounter a sequence of 6 idle CCA which leads to a termination of the procedure after 7 CCA checks. The probability of this path is given by $p \cdot (1-p)^6$. Considering all possible paths through the state machine we can give E_b as:

$$\begin{aligned} E_b(p) \quad = \quad & p \cdot (1-p)^{N_{sil}} \cdot \Big((N_{sil} \cdot (T_3 + T_w) + T_1) \\ & + \sum_{m=1}^{(N_{max}-N_{sil}-2)} \sum_{n=1}^{m} \left[p^n \cdot (1-p)^{(m-n)} \right. \\ & \cdot \big((N_{sil}+m) \cdot (T_3 + T_w) + T_1 \big) \left. \right] \Big) \\ & + p \cdot \Big(1 - \big((1-p)^{N_{sil}} \\ & + \sum_{m=1}^{(N_{max}-N_{sil}-2)} \sum_{n=1}^{m} \left[p^n \cdot (1-p)^{(m-n)} \right] \big) \Big) \\ & \cdot \big((N_{max}-1) \cdot (T_3 + T_w) + T_1 \big) \end{aligned} \tag{4}$$

E_{ib} represents the case where ContikiMAC's first CCA (CCA1) returns clear but the second CCA (CCA2) returns busy which then leads to the execution of the same procedure as described for E_b. The difference here is the resulting duration of the radio on time as two CCA are executed before entering the procedure of repeated follow-up CCA checks. E_{ib} can be given as:

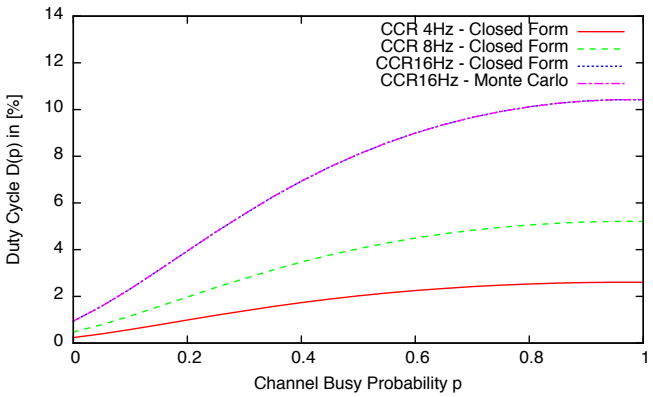

Figure 3: ContikiMAC's duty cycle in dependence of the channel busy probability p for different *CCR* (*CCR* =16 is ContikiMAC's default setting). For *CCR* =16 ContikiMAC's idle duty cycle ranges between 0.47% (no interverence) and 5.211% (interference signal is always present). Closed form solution and Monte Carlo solver provide identical results (shown for *CCR* =16).

$$\begin{aligned} E_{ib}(p) \quad = \quad & p \cdot (1-p)^{N_{sil}+1} \cdot \Big((N_{sil} \cdot (T_3 + T_w) + \\ & T_1 + T_2) \\ & + \sum_{m=1}^{(N_{max}-N_{sil}-2)} \sum_{n=1}^{m} \left[p^n \cdot (1-p)^{(m-n)} \right. \\ & \cdot \big((N_{sil}+m) \cdot (T_3 + T_w) + T_1 + T_2 \big) \left. \right] \Big) \\ & + p \cdot \Big(1 - \big((1-p)^{(N_{sil}+1)} \\ & + \sum_{m=1}^{(N_{max}-N_{sil}-2)} \sum_{n=1}^{m} \left[p^n \cdot (1-p)^{(m-n)} \right] \big) \Big) \\ & \cdot \big((N_{max}-1) \cdot (T_3 + T_w) + T_1 + T_2 \big) \end{aligned} \tag{5}$$

The expected duty cycle of ContikiMAC in the presence of varying interference is shown in Figure 3 for different channel check rates. Clearly, interference has a significant impact on energy consumption of a node. We obtain $D(0) = 0.471\%$ and $D(1) = 5.211\%$. Hence, with no interference, ContikiMAC with a channel check rate of 8 would obtain a minimum radio on time of 0.47% and with 100% interference the radio on time would be 5.211%. According to our model a node running ContikiMAC will consume 11 times more energy in a busy interference environment than in an interference free situation. Obviously, communication would not possible in an environment with continuous interference but the model shows that interference levels must be taken into account when planning a deployment's lifetime.

B. Monte Carlo Solver

Using ContikiMAC as an example, we have demonstrated that it can be quite complex to derive a closed form solution for the expected duty cycle of MAC protocols. A simpler and more flexible solution to determine the expected duty cycle is to employ a Monte Carlo solver. The functionality of the

channel check sequence employed by a MAC protocol (as shown in Figure 1 for ContikiMAC) is implemented for the solver. To determine the expected radio receiver on time $E(p)$ the implementation of the MAC's channel check sequence is executed N times and for each run the resulting radio on time $E_n(p)$ is recorded. $E(p)$ is calculated as:

$$E(p) = 1/N \cdot \sum_{n=1}^{N} E_n(p) \tag{6}$$

The implementation of the channel check sequence requires an emulation of a CCA check. Within the Monte Carlo solver a simple function is used which returns a positive or negative CCA result based on the measured interference level given by the probability p of observing a busy channel.

Our Monte Carlo solver is implemented using the LUA scripting language. New MAC protocols can be included by simply implementing one function representing the channel check sequence. Input for the solver is the measured interference in form of the channel busy probability p and the frequency f with which the channel check sequence is carried out by the MAC protocol. Output of the solver is the expected duty cycle $D(p)$.

Algorithm 1 shows the core function of the Monte Carlo solver used to evaluate the expected radio on time $E(p)$ as stated in Equation 6. The algorithm is the implementation of ContikiMAC's channel check procedure as shown in Figure 2. Clearly it is easier to adjust this implementation to accommodate changes in a MAC design than to change the closed form solution as described in the previous sections.

A result of the Monte Carlo solver is shown in Figure 3 for *CCR* =16. Comparison of the result with the result provided by the closed form solution shows little deviation. The largest deviation between both methods is 0.25% for a channel busy probability of $p = 0.3$. Hence we conclude that the Monte Carlo solver is sufficiently accurate for practical energy consumption estimation while providing flexibility to handle different MAC protocols or adjustment of MAC protocol mechanics.

Input: busy probability: p
Result: radio on time: E

```
E = 0; first = 0; silence = 0; max = 0;

E += T_1;
if cca_clear( p ) == FALSE then
    return E;
end
E += T_2;
if cca_clear( p ) == FALSE then
    return E;
end
while TRUE do
    if first != 0 then
        first++;
        period++;
        if cca_clear(busy_prob) == TRUE then
            silence++;
        else
            silence = 0;
        end
        if (silence > N_sil) OR (max > N_max) then
            return E;
        end
        E += T_3 + T_w;
    end
end
```

Algorithm 1: The core function of the Monte-Carlo solver used to evaluate the expected radio on time $E_n(p)$. This function is run N times to evaluate the expected radio on time $E(p)$ as shown in Equation 6. The algorithm is the implementation of ContikiMAC's channel check procedure as shown in Figure 2.

VI. Evaluation

To evaluate the described methods for prediction of energy consumption in an interference environment we carry out three different types of experiments. In the first experiment we use controlled WiFi interference to evaluate accuracy of the energy consumption prediction under different levels of interference in a practical setting. In the second set of experiments we analyse the efficiency of our energy consumption prediction in two deployment settings. The third experiment evaluates the impact of traffic on the prediction accuracy of our estimation method.

A. Experiment1: Controlled Interference

This experiment is carried out in a lab. A WiFi access point is placed in one corner, with a client situated in the opposite corner to generate interference for nodes deployed in the lab. The WiFi client transmits data for interference purposes using the iperf tool [16]. Different data rates are used to generate interference levels with increasing busy probability p. Table II shows the traffic rates used and the corresponding resulting busy probability p. p is measured during each experimental run of 5min duration during which we also measure the duty cycle. We then use the measured busy probability p as input for the Monte Carlo solver which allows us to compare duty cycle prediction and measurement for a given interference setting. We use traffic rates which correspond to a busy probability range from 1.74% to 23.81%. As there is always interference present (background WiFi interference from access points in the building) the lowest level of busy probability is $p = 1.74\%$. In this situation no controlled interference is created and only background interference is included. The highest busy probability is $p = 23.81\%$; we do not use higher levels as in such case a deployed sensor network would become infeasible as interference levels would be too high.

Figure 4 shows the results of our experiment. As it can be seen, duty cycle measurement and prediction are very

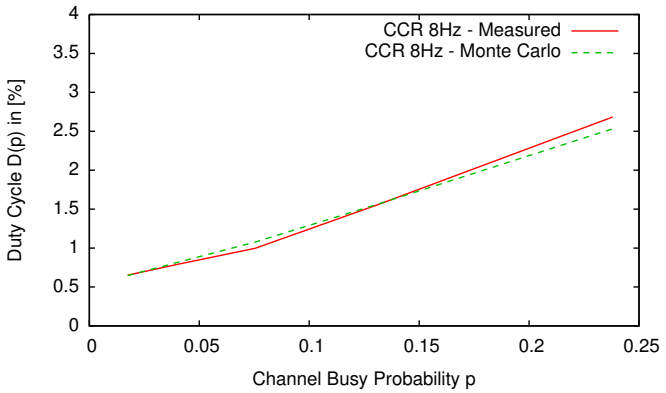

Figure 4: ContikiMAC's duty cycle in dependence of artificially induced WiFi Interference for *CCR* =8. The channel busy probability p ranges from 0% to 23% which corresponds to 8000kbit/s. The measured duty cycle and the predicted duty cycle via the Monte Carlo solver are shown.

Busy Probability p [%]	Data Rate [kbit/s]
1.74	0
3.91	500
7.55	2000
13.10	4000
23.81	8000

Table II: The mapping of WIFi data rates and measured channel busy probability p. Experiments were carried out in an environment with background activity and even with no artificially induced interference a busy probability of $p = 1.74\%$ is recorded.

close. The graph shown corresponds to the the graph shown in Figure 3. The largest deviation of the prediction from the measurement is for $p = 7.55\%$ with 7.4%. Thus we conclude that an accurate prediction (i.e. with less than 7.4% deviation) of energy consumption in a deployment area is feasible if the interference in form of channel busy probability p is known. In this experiment p is measured while measuring energy consumption and therefore an accurate value of p is known. In practice p (or a worst-case value of p) would need to be estimated before deployment.

B. Experiment 2: Uncontrolled Interference

In two separate deployments we evaluate the accuracy of the energy consumption estimation over a longer period of time. In both deployments (in a meeting room and in an office) two sensor nodes are deployed. One node is used to record the environmental interference in form of the channel busy probability p on a minutely basis while the other node is used to record the duty cycle at 5-minute intervals. From this collected data we calculate an hourly average for the measured duty cycle and an hourly average for the channel busy probability.

The results of these experiments are depicted in Figure 5 and Figure 6. In both setups it is clear that duty cycle estimation follows closely the actual observed duty cycle. The average deviation of the predicted duty cycle from the measured duty cycle is 6.23% and 2.09%. The worst case deviation is 13.1% and 12.94%.

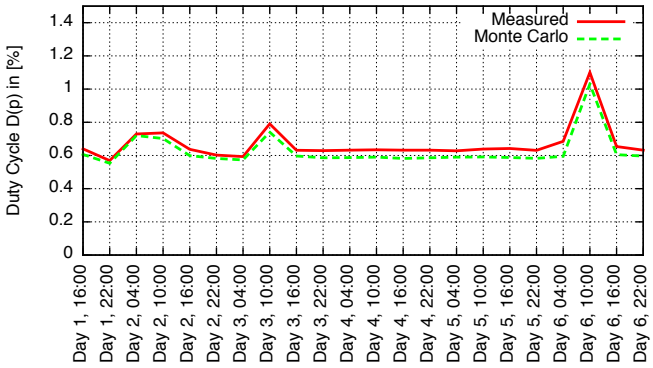

Figure 5: ContikiMAC's duty cycle estimated and measured over time (*CCR* =8). This experiment was carried out in a meeting room.

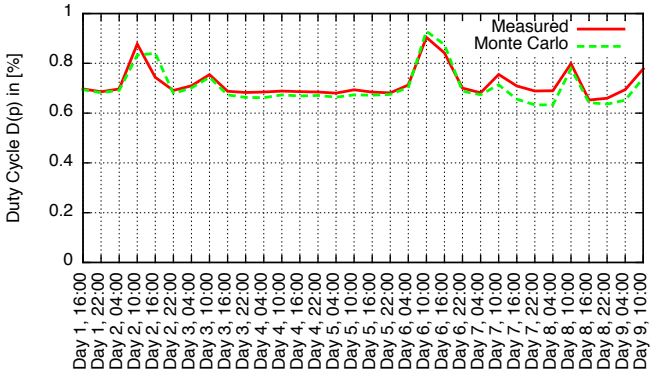

Figure 6: ContikiMAC's duty cycle estimated and measured over time (*CCR* =8). This experiment was carried out in an office.

In both scenarios the achieved duty cycle lies well above the minimal 0.47% ContikiMAC can achieve in an absolute interference free environment (see previous section). This shows that consideration of interference patterns in a deployment area are essential to estimate lifetime of a deployment.

Prediction of the duty cycle is fairly accurate if the level of interference in the deployment region in known. As can be seen from Figure 5 and Figure 6 interference in the deployment region is not constant but relatively stable over long time periods. One could use the worst case interference level observed as input for duty cycle (and hence energy) estimation to obtain an upper bound on energy consumption.

C. Experiment 3: Background Traffic

So far we have demonstrated that our energy consumption prediction method is relatively accurate if interference levels can be estimated and the assumption holds that the dominant energy cost is idle listening. In the previous two experiments nodes consume energy (or contribute to the duty cycle) only due to periodic listening for incoming packets; data is not transmitted. However, additional packet transmission will invalidate model assumptions and may make energy consumption prediction inaccurate.

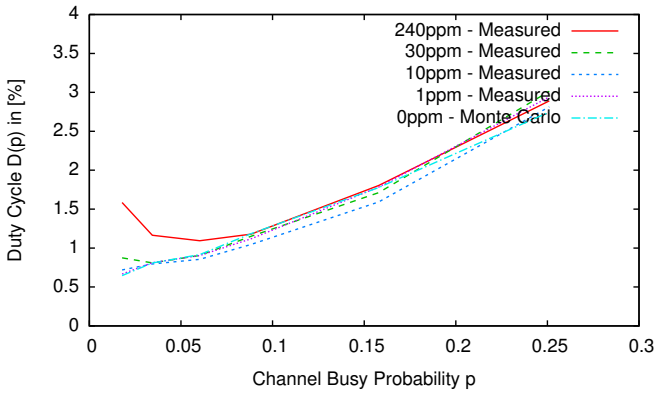

Figure 7: ContikiMAC's duty cycle in dependence of artificially induced WiFi Interference for *CCR* =8. The channel busy probability p ranges from 0% to 23% which corresponds to 8000kbit/s. The measured duty cycle with varying transmission rates and the predicted duty cycle via the Monte Carlo solver are shown.

In this experiment a sensor node transmits periodic packets to another node while being subjected to varying levels of WiFi interference. The experiment setup is similar to the first with the addition of packet transmissions.

The results of these experiments are depicted in Figure 7. For very low data rates (e.g. one packet per minute, 1ppm) measured duty cycle and predicted duty cycle are very close for all levels of interference. For high traffic rates such as 240ppm the predicted duty cycle does not match the observed duty cycle as our initial assumption does not hold: idle listening is not the dominant cost. However, this is only true for low levels of interference; when interference is high false receiver wake ups are the dominant cost. This experiment shows that duty cycle predictions are feasible with present traffic in most scenarios.

VII. Conclusion

For commercial WSN deployments it is necessary to estimate network lifetime in order to judge if an application scenario is viable. Lifetime depends in most cases on the transceiver duty cycle which, as we have shown, depends heavily on the interference environment in which the node is operated. For the popular ContikiMAC protocol the duty cycle can range from 0.47% (no interference) to 5.211% (interference signal is always present). Hence, lifetime may be up to 11 times shorter when deploying a node in heavy interference environments. We have described the methods and tools necessary to estimate duty cycle (and therefore energy consumption) before a node is deployed. The effectiveness of the proposed methods was demonstrated, using ContikiMAC as an example. However, we believe the described approach can also be used to analyse other MAC protocols such as LPL used in TinyOS. Our next step will be to use the described analysis tools to optimise ContikiMAC's procedures for operations in noisy environments.

Acknowledgements

This work was supported by the European Commission under the contract INFSO-ICT-317826 (RELYonIT).

References

[1] V. Shnayder, M. Hempstead, B.-r. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM, 2004, pp. 188–200.

[2] O. Landsiedel, K. Wehrle, and S. Götz, "Accurate prediction of power consumption in sensor networks," in *Proc. of the Second Workshop on Embedded Networked Sensors*. Citeseer, 2005.

[3] J. Eriksson, F. Österlind, N. Finne, A. Dunkels, N. Tsiftes, and T. Voigt, "Accurate network-scale power profiling for sensor network simulators," in *Wireless Sensor Networks*. Springer, 2009, pp. 312–326.

[4] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: Accurate and scalable simulation of entire tinyos applications," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 2003, pp. 126–137.

[5] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," in *ACM SIGOPS operating systems review*, vol. 34, no. 5. ACM, 2000, pp. 93–104.

[6] Y.-T. Wang and R. Bagrodia, "Scalable emulation of tinyos applications in heterogeneous network scenarios," in *Mobile Adhoc and Sensor Systems, 2009. MASS'09. IEEE 6th International Conference on*. IEEE, 2009, pp. 140–149.

[7] A. Lachenmann, P. J. Marrón, D. Minder, and K. Rothermel, "Meeting lifetime goals with energy levels," in *Proceedings of the 5th international conference on Embedded networked sensor systems*. ACM, 2007, pp. 131–144.

[8] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, "Software-based on-line energy estimation for sensor nodes," in *Proceedings of the 4th workshop on Embedded networked sensors*. ACM, 2007, pp. 28–32.

[9] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. IEEE, 2004, pp. 455–462.

[10] K. R. Chowdhury and I. F. Akyildiz, "Interferer classification, channel selection and transmission adaptation for wireless sensor networks," in *Communications, 2009. ICC'09. IEEE International Conference on*. IEEE, 2009, pp. 1–5.

[11] N. M. Boers, I. Nikolaidis, and P. Gburzynski, "Sampling and classifying interference patterns in a wireless sensor network," *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 1, p. 2, 2012.

[12] J. Ansari, T. Ang, and P. Mähönen, "Wispot: fast and reliable detection of wi-fi networks using ieee 802.15. 4 radios," in *Proceedings of the 9th ACM international symposium on Mobility management and wireless access*. ACM, 2011, pp. 35–44.

[13] F. Hermans, L.-Å. Larzon, O. Rensfelt, and P. Gunningberg, "A lightweight approach to online detection and classification of interference in 802.15. 4-based sensor networks," *ACM SIGBED Review*, vol. 9, no. 3, pp. 11–20, 2012.

[14] A. Dunkels, "The ContikiMAC Radio Duty Cycling Protocol," Swedish Institute of Computer Science, Tech. Rep. T2011:13, Dec. 2011. [Online]. Available: http://www.sics.se/~adam/dunkels11contikimac.pdf

[15] J. Brown, U. Roedig, C. Boano, and K. Roemer, "Estimating packet reception rate in noisy environments," in *Proceedings of the 39rd IEEE Conference on Local Computer Networks, 2014. LCN 2014.* IEEE, 2014, pp. 583–591.

[16] (Accessed 2015) NLANR/DAST : Iperf - the TCP/UDP bandwidth measurement tool. https://iperf.fr/. [Online]. Available: https://iperf.fr/