Research by Experimentation
for Dependability on the
Internet of Things

D-2.2 & D-2.3 – Report on Protocol Models & Validation and Verification

Grant Agreement no: 317826
www.relyonit.eu

Date: December 17, 2014

Author(s) and affiliation: Marco Zúñiga, Ioannis Protonotoarios, Si Li, Faisal Aslam, Koen Langendoen (TUD); Carlo Alberto Boano, Kay Römer (TUG); James Brown, John Vidler, Ibrahim Ethem Bagci, Utz Roedig (ULANC); Luca Mottola, Thiemo Voigt and Ioannis Glaropoulos (SICS)

Work package/task: WP2
Document status: Final
Dissemination level: Public
Keywords: Wireless sensor and actuator networks, temperature, interference, protocol models.

Abstract This is the final report for Tasks 2.3 (Protocol Models) and 2.4 (Validation and Verification). Initially, these tasks were going to be presented in two different deliverables, D2.2 and D2.3. However, due to the strong relation between mathematical models and their validation, the program officer and the consortium decided to present a joint deliverable. Out of the five protocols described in D-2.1, we present results for four of them: Estimation of Packet Reception Rate, JAG, MiCMAC, and TempMAC. We also introduce two new models and their validation: Radio Energy Prediction, to estimate the radio energy consumption under interference, and TempLife, a general framework to capture the lifetime of the network. This document contains information that will be under submission in academic conferences, and hence, should be kept private.

Disclaimer

The information in this document is proprietary to the following RELYonIT consortium members: Graz University of Technology, Swedish ICT, Technische Universiteit Delft, University of Lancaster, Worldsensing, Acciona Infraestructuras S.A.

The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The user uses the information at his sole risk and liability. The above referenced consortium members shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials subject to any liability which is mandatory due to applicable law.

Contents

List of Figures

List of Tables

1 Executive Summary

To have a deep understanding of networking protocols, it is necessary to construct analytical models for them. Most protocols are designed and evaluated under a limited set of environmental conditions. This approach does not allow the user to estimate the performance of the network when exposed to different settings. In order to predict how a protocol works in a given (previously untested) circumstance, we need models.

We follow a bottom-up approach to build analytical models for the various protocols developed by our consortium. Mitigating the impact of temperature and interference requires a deep understanding at various levels. First, we had to model the *phenomenon* itself (temperature or interference). Then, we had to model the impact of the phenomenon on our hardware *platforms*. The required environmental and platform models were developed in WP1. In this report we use those 'low-level models' to derive the protocols' models.

Our models allow us to predict and adjust the performance of sensor networks according to the current characteristics of the scenario. The models developed in our consortium play a crucial role not only in estimating the performance of the network under various settings, but also in exposing parameters that can be adjusted before and at runtime to optimize the performance of the network. This multi-level approach allows us to connect the *'low level'* characteristics of the environment and platform (WP1) with the *'high level'* requirements of the application during runtime (WP3).

This report presents models for six protocols: four of them focus on overcoming interference effects and the other two on temperature. Our models (i) describe how nodes can adjust their packet sizes to avoid interference (estimation of packet reception rate); (ii) optimize the number of channel checks performed under interference (radio energy prediction); (iii) provide bounds for the performance of agreement protocols operating under various interference patterns (JAG); (iv) capture the reliability of a multichannel protocol that avoids interference (MicMAC); (v) quantify the delivery rate in networks affected by high temperatures, and (vi) quantify the lifetime of networks.

Considering that the derivation of a model is intrinsically related to its validation, this report merges the content of two planned deliverables in the Description of Work (D2.2 and D2.3). The Validation and Verification of each protocol, i.e., what would have been D2.3 in the original plan, is clearly marked in each subsection of Chapters 3 and 4 as: "Validation and Verification (D-2.3)".

2 Introduction

During the last decade, the deployments of wireless networks have been experiencing some dramatic changes. On one hand, the explosive growth of wireless devices is crowding the radio frequency spectrum and is leading to high levels of interference. This high interference reduces the delivery rate of networks, increases their latency, and makes network performance very hard to predict. On the other hand, more and more wireless networks are being deployed outdoors without much infrastructure to protect them from their surrounding environment. We have shown that temperature is an important environmental parameter that plays a major role — affecting communication, timing, and battery lifetime.

Overcoming interference and temperature effects is hard because every scenario has a unique setting that can not be predicted ahead of time: a big city close to the equatorial line may have a high temperature and high levels of WiFi interference, while an oil exploration platform in the North Sea may have low temperatures with most of the interference coming from industrial wireless networks. Furthermore, the settings are not static, temperature and interference change significantly throughout the day. For example, during the day, an outdoor network in a sunny city is exposed to the compound effects of interference and temperature, while at night the impact of both phenomena decreases significantly.

When several unknowns affect the performance of a system, it becomes increasingly complex to evaluate all the possible scenarios in testbeds. Arguably, the best way to estimate and adjust the performance of such a system is to model it mathematically. Given a particular environment, mathematical models can not only predict the performance of the system, but they can also be used to calibrate parameters before and at runtime (to adjust to continuously changing environments).

To allow a robust operation of future wireless networks under interference and temperature effects, we developed mathematical models for the protocols proposed in previous deliverables. These protocols will allow us to (i) provide quality-of-service guarantees and (ii) perform optimisations at runtime. Our modeling efforts are divided in two fronts: models for protocols overcoming interference and models for protocols overcoming temperature.

For interference effects, we model and validate three of the protocols presented in D-2.1: JAG, MiCMAC, and Estimation of Packet Reception Rate; and add a new model named Radio Energy Prediction. The main problem of interference is that it causes packets loss, which in turn affects three important network metrics: delivery rate, energy, and latency. Arguably, the best way to overcome interference is to avoid it; with this goal in mind, we model JAG and MiCMAC. JAG uses the interference models developed in work package 1 to provide probabilistic guarantees during agreement events, such as changing the communication channel between

two nodes, from a crowded channel to a less crowded one. MiCMAC aims at hopping among various channels until encountering one that is (relatively) free. For MiCMAC, we first model a single-channel MAC protocol, and we then enhance this model for multi-channel operation. In the event that interference can not be avoided (because all channels are heavily used), we need protocols that make the best out of the limited idle periods observed by nodes. For this latter scenario, we propose Estimation of Packet Reception Rate (EPRR) and Radio Energy Prediction (REP). Based on the interference models developed in WP1, VPS adjusts the packet size to increase the probability of successfully delivering information under various interference patterns; and REP optimizes the number of channel checks performed under interference to reduce the energy consumption of nodes.

For temperature effects, we model TempMAC, which is a protocol presented in D-2.1, and TempLife, which is a new theoretical framework aimed at analyzing the lifetime of the network. For TempMAC, the goal is to quantify the impact of temperature on the delivery rate (leveraging the platform models proposed in WP1). For TempLife, the aim is to estimate the lifetime of networks running data collection protocols.

The four models for interference are complete and have been validated in testbeds, except for MicMAC which was evaluated through simulations. Regarding the models for temperature, TempMAC is complete and evaluated in testbeds. Temperature is known to affect the discharge of batteries (higher temperatures lead to lower battery lifetimes), and data collection protocols are notoriously known for being unbalanced. The combination of *variable battery capacities* with *heavily loaded nodes* can reduce significantly the lifetime of the network. To have an approximate idea about how temperature can affect the lifetime of collection protocols, we evaluate TempLife with different initial values of battery capacities.

Instead of presenting the validation of the models in a separate deliverable (D-3.3), we present them in this deliverable because combining the models' derivation with their verification provides a better understanding of our work. The protocols presented in Chapters 3 and 4 contain a subsection named "Validation and Verification (D-3.3)" which has the content of what would have been otherwise D-3.3. Figure 2.1 shows an overview of the models, their connection with the key network metrics and the previously developed environmental and platform models.

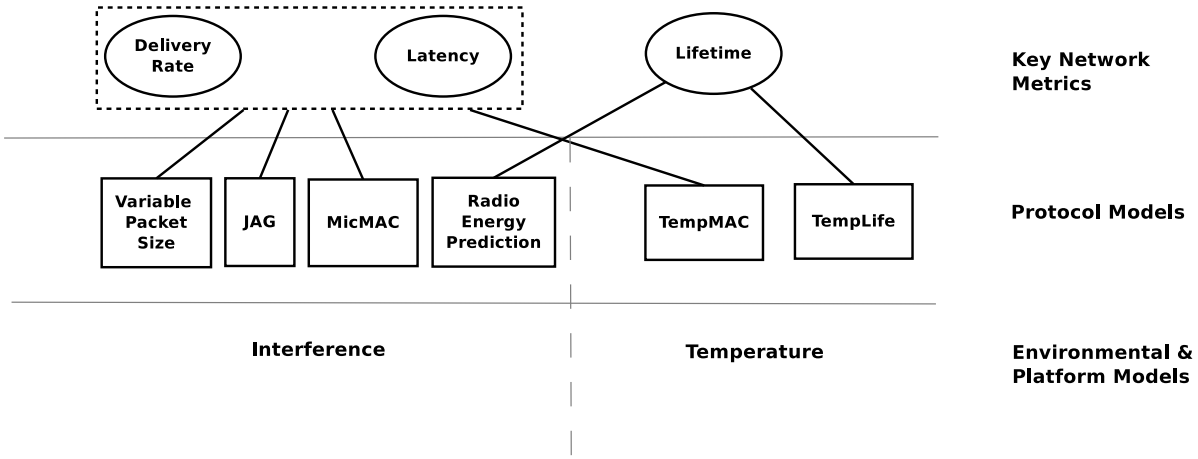


Figure 2.1: Overview of protocol models.

3 Models for Protocols Tackling Interference

3.1 Jamming-based AGreement (JAG)

In this chapter, we illustrate a model of JAG, an enhanced protocol for reliable agreement in congested environments that consists in a three-way handshake in which the last acknowledgement is sent in the form of a jamming signal. Figure 3.1 shows the basic principle of the protocol: JAG uses a jamming signal instead of a message transmission to increase the probability that the reception of the acknowledgement message by node S is correctly identified by node R.

As we have presented in [47], JAG is particularly suitable for congested environments and can make sure that two neighbouring nodes agree, outperforming message-based approaches in terms of agreement probability, energy consumption, and time-to-completion.

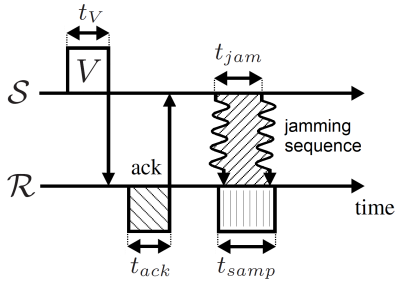


Figure 3.1: Illustration of JAG: the last acknowledgement of the 3-way handshake between nodes $\mathcal{S}$ and $\mathcal{R}$ is sent in the form of a jamming signal.

Variable	Description
t_{pkt}	Transmission delay of PKT containing V
t_{ack}	Transmission delay of ACK
t_{jam}	Duration of jamming signal in JAG
X	Random variable denoting the length of the idle period
$p(x)$	Probability density function (*pdf*) of X

Table 3.1: Notation used in our probabilistic model.

3.1.1 Predictable Performance under Interference

The length of the jamming sequence t_{jam} can be tuned in order to provide probabilistic guarantees on the fraction of disagreements. Denoting t_{busy}^{max} as the maximum busy period that can be encountered in the presence of interference, we can guarantee that $\mathcal{S}$ and $\mathcal{R}$ will agree on V by setting $t_{jam} > t_{busy}^{max}$. In such a case, an idle period will surely be encountered during t_{samp}, and the absence of a jamming sequence unequivocally detected, as discussed in Sect. [11]. Hence, the most pernicious outcomes (disagreements) are eliminated, and only positive or negative agreements can occur.

In some scenarios, however, one may need to know the outcome of the agreement process before t_{busy}^{max}. In these cases, where $t_{jam} \leq t_{busy}^{max}$, disagreements may occur. For these type of scenarios, given t_{jam}, we derive an upper bound for the probability of obtaining disagreements. In this way, a user with stringent real-time constraints can assess if the fraction of disagreements is within the limits permitted by the QoS requirements of the application. We now present a probabilistic model bounding the fraction of disagreements is presented in Sect. 3.1.1.

When setting $t_{jam} > t_{busy}^{max}$ is not possible, it is important to precisely calibrate t_{jam} so that a user with stringent real-time constraints can know in advance the fraction of disagreements to be expected. Hence, we now derive a probabilistic model that bounds the probabilities of positive agreements and disagreements for JAG, given a certain value of t_{jam}.

The parametrization of the probabilistic model requires the user to run a wireless sniffer in order to capture the characteristics of the surrounding interference. We use continuous RF noise measurements to measure the duration of idle and busy periods and compute their probability density function (*pdf*): a channel is defined as busy if the RSSI is higher or equal than a configurable threshold R_{thr} and idle otherwise.

Preferably, this operation should be carried out before the actual deployment, but it would also be possible to characterize interference at runtime, for example in case the RF environment has changed significantly from the prior observation.

The user can then follow three simple steps: (i) compute the *pdf* of the idle periods $p(i)$, where i represents the length of the idle period, (ii) compute the conditional *pdf* of the busy periods following the idle periods $p(b > x|i)$, and (iii) use the model to obtain the value of t_{jam} that provides the desired QoS.

Table 3.1 shows the notation used in our analysis. Our goal is to derive the probabilities of positive agreements and disagreements for JAG given a certain value of t_{jam}. First, we obtain the probability of selecting an idle period of length i, then, we derive the probabilities of obtaining positive agreements and disagreements over all possible idle periods.

Denoting $p(i)$ as the probability density function of the idle periods formed by the interference pattern, the probability of selecting an idle period of length i is given by:

$$s(i) = \frac{ip(i)}{\sum_{i=1}^{\infty} ip(i)} \tag{3.1}$$

i.e., the more frequent and the longer the idle period, the higher the likelihood of selecting it.

In order to derive the required probabilities, we need to understand the interplay between the length of an idle period i and the 3-way handshake method used by JAG (i.e., the transmission of the PKT embedding V, the ACK, and the JAM signal). In principle, losing an ACK should

lead to negative agreements (see complete discussion in [11]). The practical implementation of JAG, however, takes an optimistic approach that increases the likelihood of positive agreements at the cost of turning some negative agreements into disagreements. In JAG, if $\mathcal{R}$ sends the ACK, four outcomes can occur: (i) a positive agreement, if the ACK is successfully delivered to $\mathcal{S}$ and the JAM signal is correctly decoded by $\mathcal{R}$; (ii) a negative agreement, if the ACK is lost and $\mathcal{R}$ detects the *lack of* JAM; (iii) *another positive agreement*, independently of the fact that the ACK is received or not if, after sending the ACK, $\mathcal{R}$ detects an interference signal with a strength higher than the expected JAM signal and hence assumes a successful transaction (this is the optimistic approach, which assumes the JAM was buried within the stronger signal); and (iv) a disagreement, if the ACK is lost, but, by chance, a high interference signal lasts longer than t_{samp}. In this case, $\mathcal{R}$ assumes, mistakenly, a successful exchange, i.e., a negative agreement turns into a disagreement.

Based on the above description, in JAG, positive agreements are given by the following equation:

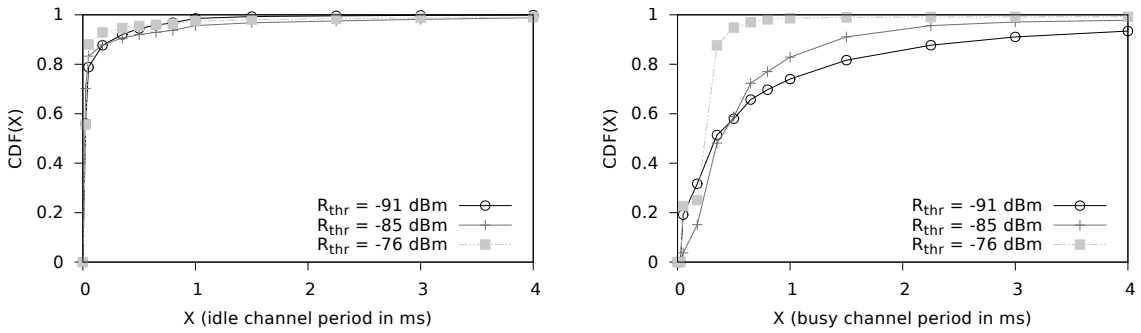
$$P_{\text{jam}}\{\text{Pos. Agr.}\} = \sum_{i>t_{\text{pkt}}+t_{\text{ack}}}^{\infty} s(i)(1 - \frac{t_{\text{pkt}} + t_{\text{ack}}}{i}) \tag{3.2}$$

whereby the first term of the product states the probability of obtaining an idle slot of length i, and the second term states the probability that the selected idle slot can "contain" the transmission of the packet followed by the ACK ($t_{\text{pkt}} + t_{\text{ack}}$).

In order to obtain the fraction of disagreements, we use a bounding probability. There are three necessary but not sufficient conditions to obtain disagreements: (i) PKT is transmitted successfully, (ii) the ACK is corrupted and (iii) the interference signal after the ACK is longer than t_{jam} (to shadow the JAM signal). Hence, we define the probability of obtaining disagreements with JAG as follows:

$$\begin{aligned} P_{\text{jam}}\{\text{Disagreement}\} \leq & \sum_{i=1}^{t_{ack}} s(i)p(b > t_{jam}|i) + \\ & \sum_{i>t_{ack}}^{t_{pkt}+t_{ack}} s(i)p(b > t_{jam}|i)(1 - \frac{\min(t_{pkt}, i)}{i}) + \\ & \sum_{i>t_{pkt}+t_{ack}}^{\infty} s(i)p(b > t_{jam}|i)(\frac{t_{ack}}{i}) \end{aligned} \tag{3.3}$$

Each of the sums on the right side of the equation has three terms. The first term $s(i)$ denotes the probability of obtaining an idle slot of length i. The second term $p(b > t_{jam}|i)$ denotes the probability of obtaining a busy period b longer than t_{jam} after an idle period of length i (the minimum requirement to shadow the jamming signal). The third term differs for each sum, and denotes the probability that the ACK will be corrupted: in the first summation this probability is 1, because the idle time is less than t_{ack}, i.e., the ACK will always be corrupted; in the second and third summations, this probability describes the chances that the agreement starts early enough to allow a successful delivery of PKT, but late enough to corrupt the ACK. Please note that, in Eq. 3.3, the term $p(b > t_{jam}|i)$ assumes that the corrupted ACK ends exactly before the next busy period starts. In practice, the ACK will likely have a Δ overlap with the beginning of the busy period b, and hence, b will need to be longer than $(t_{jam} + \Delta)$



(a) Idle periods

(b) Busy periods

Figure 3.2: Cumulative distribution function (CDF) of idle and busy periods measured by a Maxfor MTM-CM5000MSP node in the presence of a laptop continuously downloading a file from a nearby access point.

to lead to a disagreement. Given that $p(b > t_{jam}|i) > p(b > (t_{jam} + \Delta)|i)$, in practice, we can expect a lower fraction of disagreements.

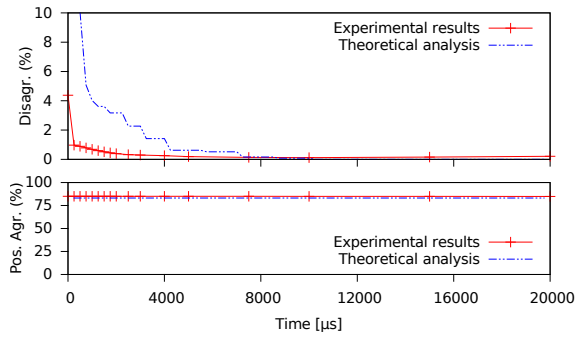
For the case of disagreements, JAG allows the user to fine-tune the duration of t_{jam} according to the requirements of the application (Eq. 3.3). In Sect. 3.1.2, we will observe that this fine-tuning capability is central to provide QoS guarantees.

3.1.2 Validation and Verification (D-2.3)

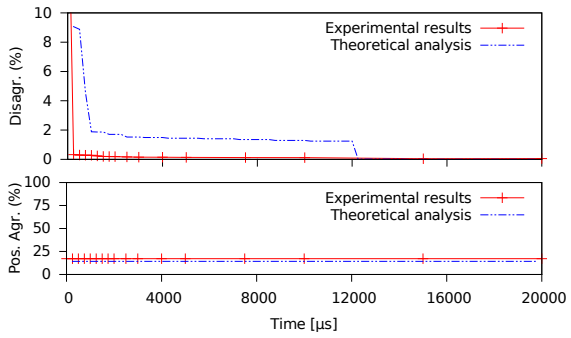
We now evaluate the goodness of the probabilistic model presented in Sect. 3.1.1 with respect to the predictability of the performance of JAG. In order to do this, we firstly obtain the *pdf* of idle and busy periods using sensor nodes in wireless sniffer mode in the scenarios described in the previous sections, i.e., in the presence of JamLab's emulated interference and real Wi-Fi interference generated by a laptop (the *pdf*s in the presence of real Wi-Fi interference are shown in Fig. 3.2). Then, based on equation (2) and (3), we obtain an upper bound for the probability of obtaining disagreement and a lower bound for the probability of obtaining positive agreements as a function of t_{jam} using t_{pkt} = 1 ms, t_{ack} = 750 μs, and t = -90 dBm.

By running JAG on real wireless sensor nodes, we verify experimentally whether the probabilistic model is able to predict the performance of JAG. The results illustrated in Fig. 3.3 show that our probabilistic model parametrizes correctly t_{jam} by giving an upper bound on the amount of disagreements and a lower bound on the amount of positive agreements, hence predicting the performance of the protocol correctly. Note that the probabilistic model was computed for every possible t_{jam}, whereas due to memory limitations of real nodes only a finite amount of t_{jam} were computed experimentally. Please note that Fig. 3.3 shows a different performance between emulated and real interference: whilst JamLab is designed to attain repeatability and test algorithms under the same conditions, real-world settings have several variables affecting their dynamics.

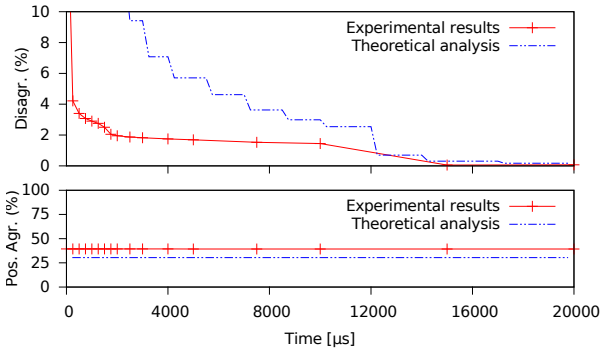
Based on our results, we can conclude that our theoretical model is indeed able to parametrize



(a) Bluetooth



(b) Emulated Wi-Fi



(c) Real Wi-Fi

Figure 3.3: Comparison of the rate of positive agreement and disagreement obtained running JAG on real wireless sensor nodes, and deriving the probabilities using the analytical model shown in Sect. 3.1.1. The model actually returns a lower bound for positive agreements and an upper bound for disagreements.

JAG and predict correctly the maximum amount of disagreements occurring for a given t_{jam}. This can be useful when the latter is shorter than the longest busy period created by interference (t_{busy}^{max}).

3.2 MiCMAC

In this chapter, we model MiCMAC [29], which is an extension of ContikiMAC [19] for multiple channels. When configuring MiCMAC to use a single channel, it operates identically to ContikiMAC. Hence, we start by modelling ContikiMAC. ContikiMAC has similarities to X-MAC, which we have modeled earlier in the context of pTunes [45]. We thus base our model of ContikiMAC on pTunes' X-MAC model. After discussing how X-MAC and its most important parameters are affected by interference, we extend our model of ContikiMAC to multiple channels.

3.2.1 pTunes: ContikiMAC's Modeling Framework

pTunes breaks up the modeling of low-power MAC protocols into three distinct layers [45]. The upper layer defines application-level metrics (Reliability R, Latency L, Lifetime T) as functions of link and node-specific metrics (R_l, L_l, T_n). The middle layer expresses these metrics in a protocol-independent manner, and provides the entry point for the modeling of a concrete MAC protocol by exposing six terms to the lower protocol-dependent layer. Binding these terms to concrete protocol-specific expressions is sufficient to adapt the network-wide performance model in pTunes to a given MAC protocol.

Model inputs are the MAC parameters and the network state, comprising information about routing topology, traffic volumes, and link qualities. As a measure of the latter, we take the probability of successful transmission p_l over the link to the parent in the routing tree. To keep the models simple and practical, pTunes assumes the delivery of individual packets to be independent of their size, of the delivery of any other packet, and of the link direction they travel along.

Application-level Metrics: In a typical data collection scenario with static nodes, a tree-shaped routing topology provides a unique path from every sensor node to a sink node. These paths are generally time-varying, as the routing protocol adapts them according to link quality estimates among other things [35]. pTunes uses $\mathcal{N}$ to denote the set of *all nodes* in the network excluding the sink, and $\mathcal{M} \subseteq \mathcal{N}$ to denote the set of *source nodes* generating packets. pTunes also indicates with $\mathcal{L}$ the set of *communication links* that form the current routing tree. The *path* $\mathcal{P}_n \subseteq \mathcal{L}$ originating at node $n \in \mathcal{M}$ includes all intermediate links that connect node n to the sink.

End-to-end reliability and latency. The reliability $R_{\mathcal{P}_n}$ of path $\mathcal{P}_n$ is the expected fraction of packets delivered from node $n \in \mathcal{M}$ to the sink along $\mathcal{P}_n$. Thus, $R_{\mathcal{P}_n}$ is the product of per-hop reliabilities R_l, $l \in \mathcal{P}_n$. We define the *end-to-end reliability* R as the average reliability of all paths $\mathcal{P}_n$.

$$R = \frac{1}{|\mathcal{M}|} \sum_{n \in \mathcal{M}} R_{\mathcal{P}_n} = \frac{1}{|\mathcal{M}|} \sum_{n \in \mathcal{M}} \left(\prod_{l \in \mathcal{P}_n} R_l \right) \tag{3.4}$$

Likewise, the latency $L_{\mathcal{P}_n}$ of path $\mathcal{P}_n$ is the expected time between the first transmission of a packet at node $n \in \mathcal{M}$ and its reception at the sink. Thus, $L_{\mathcal{P}_n}$ is the sum of per-hop latencies L_l, $l \in \mathcal{P}_n$. Similar to (3.4), we define the *end-to-end latency* L for *successfully delivered* packets as the average latency of all paths $\mathcal{P}_n$, and omit the formula.

pTunes defines R and L as averages of all source-sink paths since the global, long-term performance is of ultimate interest for most data collection systems [40–42]. Local, short-term deviations from the requirements are usually tolerated, provided they are compensated in the long run. In other scenarios (*e.g.*, industrial settings), it might be more appropriate to define R and L as the minimum reliability and the maximum latency among all source-sink paths, which would only require modifying the two definitions above.

Protocol-independent Modeling: The section above expressed the application-level metrics R, L, and T as functions of per-hop reliability R_l and per-hop latency L_l. pTunes defines

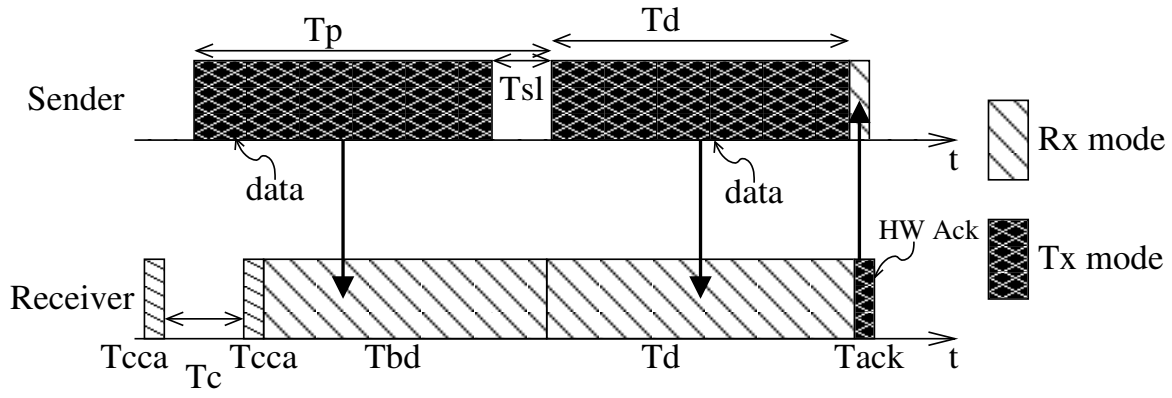


Figure 3.4: Unicast transmission in ContikiMAC.

the latter three in a protocol-independent manner, which increases flexibility and generality by isolating protocol-dependent quantities.

Per-hop reliability and latency. Several factors influence these metrics: (*i*) the MAC operation when transmitting packets, (*ii*) packet queuing throughout the network stack due to insufficient bandwidth, and (*iii*) application-level buffering (*e.g.*, to perform in-network processing). The MAC parameters control (*i*) and may avoid the occurrence of (*ii*), provided a MAC configuration exists that provides sufficient bandwidth. Application-specific in-network functionality akin to (*iii*) is not modeled in pTunes and not needed here either.

We present next expressions for per-hop reliability and latency due to the MAC operation, corresponding to (*i*). Additionally, pTunes includes models to detect situations akin to (*ii*). which are not needed in our context. Since pTunes automatically adjusts the MAC parameters to provide higher bandwidth against increased traffic, it avoids the occurrence of local packet queuing until the network capacity is fully exhausted.

pTunes defines the *per-hop reliability* R_l of link $l \in \mathcal{L}$, which connects node $n \in \mathcal{N}$ to its parent m in the routing tree, as the probability that n successfully transmits a packet to m.

$$R_l = 1 - (1 - p_{s,l})^{N+1} \tag{3.5}$$

Here, $p_{s,l}$ represents the MAC-dependent probability that a single unicast transmission over link l succeeds, and N is the maximum number of retransmissions per packet, modeling automatic repeat request (ARQ) mechanisms used by many low-power MAC protocols to improve reliability.

Furthermore, pTunes defines the *per-hop latency* L_l of link l as the time for node n to deliver a message to its parent m.

$$L_l = N_{ftx,l} \cdot T_{ftx,l} + T_{stx,l} \tag{3.6}$$

$T_{ftx,l}$ and $T_{stx,l}$ are the MAC-dependent times spent for each failed and the final successful transmission. The expected number of failed transmissions $N_{ftx,l}$ depends on $p_{s,l}$ and N, and the retransmission policy of the MAC protocol. Next we model ContikiMAC.

ContikiMAC-specific Modeling: Fig. 3.4 shows a successful unicast transmission in ContikiMAC [19]. Nodes that expect to receive a packet wake up periodically. The time interval between wake-

ups is denoted by T_{off}[1]. Each wake-up consists of two fast CCA checks of duration T_{cca}, with a short sleeping time (T_c) in between these checks unless the first check is already successful. Contrary to X-MAC, which transmits short strobe packets, ContikiMAC transmitters repeatedly transmit the whole packet. Once one of the receiver's CCA checks has detected a packet transmission, the receiver stays awake until it receives the next transmitted packet. We define T_{bd}, as the time from the CCA-based detection of a packet until the time the receiver starts receiving the packet.

To send a packet, the transmitter repeatedly transmits the data packet. ContikiMAC uses a phase-lock mechanism [19], where nodes keep track of their neighbors' wake-up time and start transmitting data packets briefly before the expected wake-up time of the intended receiver. We call the time between two transmissions of the same data packet T_{sl}. T_{sl} is short (0.4 ms in the current ContikiMAC implementation for the Tmote Sky). Note that it must be shorter than T_c, the time between the two consecutive CCA checks of the receiver, to prevent that both CCA checks fall into T_{sl}.

T_{bd} is a random variable due to the random clock drift at the receiver, which results in loss of synchronization between the transmitting and the receiving node. T_{bd}, as well as several other time variables, depend on the synchronization loss modeling. For simplicity we assume here that the receiving node wakes up at an arbitrary point in time within the transmitter's **transmission period**, T_p, **uniformly at random**, even if phase-lock is considered. The duration of the transmission period is the sum of the frame-in-the-air duration, T_d, and the inter-frame transmission time, T_{sl}: $T_p = T_d + T_{sl}$.

The exact point of receiver wake-up affects the probabilities of certain events, discussed later in this Section. We give here the probabilities of receiver waking up in the different sub-periods of T_p:

$$\Pr\{\text{Rx wake-up in } T_{sl}\} \triangleq P_{b2} = \frac{T_{sl}}{T_{sl}+T_d}$$

$$\Pr\{\text{Second CCA on idle}\} \triangleq P_{b1} = \frac{T_c}{T_{sl}+T_d}$$

$$\Pr\{\text{Both CCA on busy}\} \triangleq P_{b12} = \frac{T_d - T_c}{T_{sl}+T_d}$$

We determine the average value of T_{bd} under the above distinct cases:
$\overline{T}_{b2} = 0.5 \cdot T_{sl}$, $\overline{T}_{b1} = 0.5T_c + T_{sl}$, and $\overline{T}_{b12} = 0.5T_d + T_{sl}$, respectively.

Once the receiver has successfully received the packet it sends an acknowledgment. In ContikiMAC this is an IEEE 802.15.4 hardware acknowledgment that requires T_{ack} time units.

In the X-MAC implementation, pTunes considers several variables that are adjustable, including T_{on}, T_{off}, and N. T_{on} is the time the receiver is on to listen to data strobes but ContikiMAC uses CCA checks instead. In ContikiMAC only N and T_{off} have a direct implication on the energy consumption for low data rates. Note that our model considers the steady case where the wake-up phase of the receiver has been learnt. The current ContikiMAC implementation re-learns the wake-up phase only if no ACK has been received for 16 tries or for 30 seconds. Experiments on real testbeds have shown that once ContikiMAC has acquired the phase-lock it looses it very rarely.

[1] In other words, T_{off} denotes the duty-cycle period.

Per-hop reliability. In this paragraph we determine $p_{s,l}$, (defined in (3.5)), that is, the probability that a single unicast transmission from node n to its parent m succeeds. For this, it is required that node m can correctly detect the presence of a frame transmission, and successfully receive, **and acknowledge it**, afterwards. We define p_{CCA}, as the probability that *one* of the performed CCA checks succeeds. The data (or d-ack) packet transmission succeeds with probability p_l, which depends on several factors such as packet size and environment, for example, interference. Based on the above consideration, and assuming that the events of data and HW d-ack packet transmission success are uncorrelated, we have:

$$p_{s,l} = p_{CCA} \cdot P_l, \tag{3.7}$$

where P_l denotes the probability of successful frame transmission (p_l) and acknowledgment (p_a),

$$P_l = p_l \cdot p_a \tag{3.8}$$

In the following we derive the probability of correct CCA detection at the receiving node. As discussed above p_{cca} depends on the exact time of receiver's wake-up within the transmission period, T_p. In addition, we make the simplifying assumption that CCA trials do not result in false *positives*.

In the event of receiver wake-up sometime within T_{sl}, the node will perform a single CCA trial[2]. Consequently, $p_{CCA|b2} = p_{cca}$. Similarly, $p_{CCA|b1} = p_{cca}$, while $p_{CCA|b12} = 1-(1-p_{cca})^2$, assuming that consecutive clear channel assessment results are uncorrelated. The probability of successful frame detection is, therefore, averaged over all possible cases:

$$p_{CCA} = p_{cca} \cdot \frac{T_{sl} + T_c}{T_p} + \left(1 - (1 - p_{cca})^2\right) \cdot \frac{T_d - T_c}{T_p}. \tag{3.9}$$

Replacing (3.9) in (3.7) we can derive the link reliability under ContikiMAC. Under the generic model of ContikiMAC, the transmitter keeps sending the data frame until it is successfully acknowledged. The number of additional frame probes is limited by a time threshold T_m. This corresponds to an additional number of frame strobes, $N_m = \lfloor \frac{T_m}{T_d} - 1 \rfloor$. Based on this model, the probability of successful link transmission during a single attempt will be:

$$P_l = \left[(p_l \cdot p_a) + \sum_{k=1}^{N_m} (1 - p_l)^k \cdot (p_l \cdot p_a) \right]. \tag{3.10}$$

Per-hop latency. In this paragraph we determine $T_{ftx,l}$, and $T_{stx,l}$, defined in (3.6), that is, the expected times spent for failed and successful one-hop transmissions under ContikiMAC. A link transmission attempt may fail on three cases: *i*) the receiver CCA trials fail to wake-up the node, or *ii*) the frame transmission fails, or *iii*) the HW acknowledgment fail.

The latency for each transmission attempt has a fixed part, which includes the data and HW d-ack packet durations, T_d, and T_{ack}, respectively.

For a successfully received packet, the latency includes, additionally, a delay term, T_w, that reflects the time between the packet reception from the upper layer and first the transmission

[2]We silently assume that the frame-in-the-air duration of a packet is much shorter than the WSN duty cycle, so the receiver has a single chance to detect a transmitting probe under phase-lock consideration. This assumption is vital, and is used in later derivations as well.

attempt. This delay appears due to the considered phase-lock mechanism of ContikiMAC. This time delay appears, additionally, before each re-transmission attempt[3]. Assuming that the packet can arrive anytime within the WSN cycle – uniformly at random, the expected delay becomes

$$\overline{T}_w = 1/2\,(T_{off} + 2T_{cca} + T_c) + T_x \tag{3.11}$$

In the current ContikiMAC implementation, there is a delay in form of a number of additional CCA checks a sender performs before transmitting the first strobe. We call this delay T_x. The one-hop latency, however, includes T_{on}, as discussed above, whose expected duration is correlated with the (event of) link transmission success. Applying Bayesian inference, we get:

$$\overline{T}_{bd|ftx} = \overline{T}_{bd|b1} \cdot P_{b1|ftx,l} + \overline{T}_{bd|b2} \cdot P_{b2|ftx,l} + \overline{T}_{bd|b12} \cdot P_{b12|ftx,l}, \tag{3.12}$$

where

$$P_{bi|ftx,l} = \frac{P_{ftx,l|bi} P_{bi}}{P_{ftx,l}} = \frac{\left(1 - P_{CCA|bi} \cdot P_l\right) P_{bi}}{1 - P_{CCA} \cdot P_l}, \quad \forall i \in \{1, 2, 12\}. \tag{3.13}$$

Similarly, it holds

$$\overline{T}_{bd|stx} = \overline{T}_{bd|b1} \cdot P_{b1|stx,l} + \overline{T}_{bd|b2} \cdot P_{b2|stx,l} + \overline{T}_{bd|b12} \cdot P_{b12|stx,l}, \tag{3.14}$$

where it we have

$$P_{bi|stx,l} = \frac{P_{stx,l|bi} P_{bi}}{P_{stx,l}} = \frac{P_{CCA|bi} P_{bi}}{P_{CCA}}, \quad \forall i \in \{1, 2, 12\}. \tag{3.15}$$

In case of link transmission failure, the transmitter backs-off for T_b before retransmitting the frame. As a result, the final expressions for the expected per-hop latency become:

$$T_{ftx,l} = \overline{T}_w + T_d + T_{ack} + T_b + \overline{T}_{bd|ftx}, \tag{3.16}$$

$$T_{stx,l} = \overline{T}_w + T_d + T_{ack} + \overline{T}_{bd|stx}, \tag{3.17}$$

For the generic model that considers a maximum of $N_m + 1$ frame strobes, we replace $T_d + T_{ack}$ in (3.16) with $(T_d + T_{sl}) \cdot (N_m + 1)$. Similarly, in (3.17) we replace $T_d + T_{ack}$ with

$$\sum_{k=0}^{N_m+1} (T_d + T_{ack})(1 - (p_l \cdot p_a))^k (p_l \cdot p_a). \tag{3.18}$$

The actual value of T_b, i.e. the back-off latency, depends on the actual retransmission scheme of the link-layer protocol module, which, however, falls outside the scope of ContikiMAC. Here, we give the expressions for T_b, when considering a *csma*-like retransmission scheme – as the one in Contiki OS – where the actual value of the back-off delay grows exponentially with the number of packet re-tries. In general, it holds,

$$T_b = \sum_{i=1}^{N} T_{b|i} \cdot Pr\{\text{retry occurs after the i-th trial}\}, \tag{3.19}$$

[3]However, if back-off time is a multiplicative of the channel check interval, subsequent transmission attempts have correlated success rates.

where N denotes the maximum number of packet retransmissions. The expected back-off latencies, $T_{b|i}$ are given by the MAC retry scheme configuration. We compute the probabilities in (3.19) based on the relative frequencies of the retry occurrences at each retransmission *index*. We consider only packets requiring retransmissions, otherwise T_b is zero. The rate of packets requiring exactly k retries is

$$Pr\{\text{exactly } k \text{ retries}\} \triangleq P^{(k)} = (1-P_l)^{k-1} \cdot P_l, \forall k = 1, \ldots, N-1, \tag{3.20}$$

while $P^{(N)} = (1-P_l)^{N-1}$, as we consider both eventually transmitted and dropped packets. Consequently, the total rate of retry occurrences after the i-th try will be:

$$Pr\{\text{retry occurs after the i-th trial}\} = \frac{\sum_{j=i}^{N} P^{(j)}}{\sum_{k=1}^{N} k \cdot P^{(k)}}. \tag{3.21}$$

3.2.2 ContikiMAC under Interference

External interference has the highest impact on two parameters of the ContikiMAC model: the probability of a positive CCA check p_{cca} and the probability p_l. The latter describes the probability that a receiver having its radio turned on will successfully receive a transmitted packet.

During interference, p_{cca} actually increases while p_l decreases as the sender's packet might be corrupted during transmission. Note that the CCA check may also be a false positive, meaning that the positive CCA check was caused by interference, which makes p_{fpcca} increase. An increase of the latter only has an impact on lifetime, but not on per-hop reliability and per-hop latency, which are the two metrics we consider in this chapter.

Derivation of p_l We can derive p_l from the environmental model described in Deliverable D-1.1. As discussed in Chapter 4.2, our environmental models do not describe the individual sources of interference. Instead, we use models that can be easily used on constrained sensor nodes that are able to carry out energy detection, i.e., nodes that can measure the received signal strength in absence of packet transmissions. Denoting x_i as the RSSI noise floor sampled by a node at a given time instant, we express the occupancy of the channel as (see Equation 4.3 in D-1.1):

$$X_i = \begin{cases} Busy \ \ (1) & if \ \ x_i > R_{THR} \\ Idle \ \ (0) & if \ \ x_i \leq R_{THR} \end{cases} \tag{3.22}$$

with X_i being a binary number specifying a busy channel (1) or an idle channel (0), and R_{THR} being a user-specified threshold.

Here we describe a simple algorithm to derive p_l. Given a sequence of n noise floor samples and a packet of a certain size, we can compute the number of samples m needed to transmit the packet. We then slide the packet over the sequence of the first $n-m$ samples (where $n >> m$). For each sample we check if this sample and the next $m-1$ samples are Idle. If they are, we denote this as a success. We call the number of successes s. Then $p_l = \frac{s}{n-m}$.

The algorithm works if we have non-periodic interference. However, the receiver wakes up with a certain period (125 ms by default). If we head for the worst case, then we could in

principle (since we know that the receiver wakes up periodically) slide our window over the RSSI samples with that periodicity. For example assume the periodicity is 10 samples and the packet size is 2 samples. Then we would see how many of the samples are above the threshold for the time slots 1,2,11,12,21,22, then 2,3,12,13,22,23, then 3,4,13,14,23,24 etc., and so find the worst case when interference has the same periodicity as the wake-up of the ContikiMAC receiver.

Independence of Data packet and ACK transmissions When computing the probability of a successful frame transmission and acknowledgment using Equation 3.8, we assume independence of the probabilities of successful transmissions of data packets and acknowledgments, as in pTunes, i.e., $P_l = p_l \cdot p_a$. When packet loss is caused by interference, this assumption might not hold and we would need to change this equation to $P_l = p_l \cdot (p_a|p_l)$. The purpose of our modeling activity is to provide a lower bound. Hence, if $p_l \cdot p_a \leq p_l \cdot (p_a|p_l)$, the assumption of independence always consitutes a lower bound. From $p_l \cdot p_a \leq p_l \cdot (p_a|p_l)$ follows $p_a \leq (p_a|p_l)$. Therefore, in order to verify that we are modeling a lower bound by assuming independence, we have taken existing traces that are used in the COOJA simulator to simulate realistic radio interference [9], and shown that in all of them $p_a \leq (p_a|p_l)$. There are four such traces: one with Bluetooth interference, which leads to low packet loss around 10%, and three with WiFi interference, which have higher packet loss. The WiFi activity is caused by radio streaming, a file transfer, and video streaming. For all these traces $p_a \leq (p_a|p_l)$ is valid, which verifies that the assumption of independence does indeed present the lower bound.

3.2.3 MiCMAC: ContikiMAC on Multiple Channels

As discussed above, MiCMAC [29] is, basically, ContikiMAC, operating on multiple channels. Like ContikiMAC, MiCMAC extends the phase-lock mechanism, from keeping track of just the next wake-up time of each receiver, to, actually, include the channel, on which the receiver wakes up. In this deliverable, we derive the per-hop reliability and per-hop latency for multiple channels.

Per-hop reliability on multiple channels. In this paragraph we determine $p_{s,l}$ in (3.5), the success rate of single-hop, unicast transmission from node n to its parent m under MiCMAC. Considering the properties of the pseudo-random channel hopping sequence of MiCMAC, we **assume** that once a packet arrives from the upper layer, the wireless channel, on which the transmission will be attempted, is picked uniformly and in i.i.d. fashion among all possible channels[4], N_C. As a result of this, the probability of successful link transmission is derived as:

$$p_{s,l} = \frac{1}{N_C} \sum_{k=1}^{N_C} p_{s,l,c_k} \tag{3.23}$$

where $p_{s,l,c_k} = p_{CCA,c_k} \cdot p_{l,c_k}$, p_{CCA,c_k} has been calculated for ContikiMAC in (3.9), and p_{l,c_k} is derived in (3.8) or (3.10) for the generic frame strobe model.

[4]For the pTUNES model to work, consecutive packet retries shall have the same stochastic link transmission properties. This is guaranteed if we assume that each retry sees a uniformly selected channel, independent of previous retries.

Per-hop latency on multiple channels. As in the case for one channel (pure ContikiMAC), we need to determine $T_{ftx,l}$ and $T_{stx,l}$, the latencies corresponding to failed and successful single-hop frame transmissions. In the following we focus on the required changes with respect to ContikiMAC model presented in the previous Section.

We consider that a packet may arrive from the upper layers at a random point in time. Under a phase & channel-lock assumption, the transmitter will tune to the channel, to which the intended packet receiver will wake up in the next duty-cycle. Consequently, the initial waiting latency, $\overline{T}_w$, can still be given by (3.11), while the wireless channel for the first transmission attempt is uniformly distributed among the available channels.

Next we consider the expected time between CCA check and packet reception. $\overline{T}_{on|ftx}, \overline{T}_{on|stx}$ can still be given by Eq. (3.12) and (3.14), respectively, however, they, now, need to be averaged over all possible channels, that may exhibit different CCA trial and frame transmission success probabilities.

The remaining terms in (3.16), (3.17) do not depend on the selected wireless channel. For the generic model with multiple frame strobing, (3.18) should, now, be averaged over all available channels.

Although in the above derivations we silently assumed that successive frame retransmission attempts will observe the same (uniform) distribution of the available channels, the derivations of $T_{ftx,l}$, $T_{stx,l}$, in (3.16), (3.17) will hold, even if at each retransmission the same wireless channel is selected.

3.2.4 Validation and Verification (D-2.3)

In order to validate the model, we implement the model in Octave and verify it by comparing it against simulations in COOJA [31]. We verify to the per-hop reliability and per-hop latency of ContikiMAC since these are the basis also for MiCMAC. As shown in the previous sections, the extension from ContikiMAC to MiCMAC is not complex.

Verification in the COOJA simulator with Packet Loss

COOJA is the Contiki network simulator. COOJA runs deployable code, i.e., the Contiki code that is running inside COOJA can be used on real hardware. COOJA's focus is to assist developers in developing Contiki applications and system software such as networking stacks and operating system functionality. Because COOJA executes the real protocol stacks, it is a good tool to verify protocol implementations.

During the verification it has turned out that COOJA's functionality is not sufficient to verify our models. For example, packets that are lost in COOJA are not transferred into COOJA's radio model, and therefore cannot cause a positive CCA check at the receiver. In order to verify ContikiMAC's per-hop reliability, we hence modify the ContikiMAC layer itself to simulate failed CCA checks and broken packets (both data packets and acknowledgments). Using COOJA's timeline [32], we visually verify the correct implementation of failed CCA checks and broken packets.

For the verification we use a small Contiki application program that transmits packets with a packet transmission interval that is uniformly distributed between one and two seconds. The receiving end prints the contents of the received packets together with some statistics.

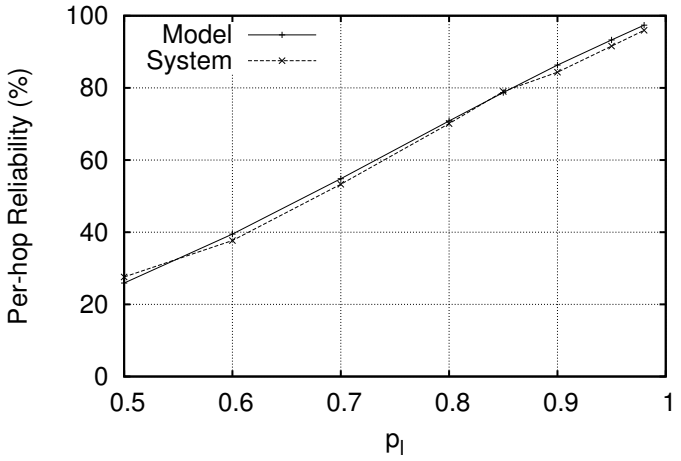


Figure 3.5: ContikiMAC per-hop reliability with 90 bytes packet size.

Table 3.2: Parameters for ContikiMAC/MiCMAC verification.

Name	Description	values
T_m	Strobing time	$\frac{1}{60}$ s
T_{off}	Wake-up interval	$\frac{1}{8}$ s
d	packet length	67 and 90 bytes
dr	data rate	250 kbit/s
T_d	Time to send data packet	$\frac{8d}{dr}$
T_{cca}	receiver's CCA check duration	$\frac{1}{8192}$ s
T_c	receiver's time between the two CCA checks	$\frac{1}{2000}$ s
T_{sl}	sender's time between two packet strobes	$\frac{1}{2500}$ s
N_m	additional frame probes (see Eq. 3.10)	1

Parameter Settings For the verification we use the parameter settings in Table 3.2. Most of the parameters are taken from the ContikiMAC implementation in Contiki. The data rate is defined by the IEEE 802.15.4 standard, which is implemented by the radios used in RELYonIT. For the experiments, the input parameters p_l and p_a are set as $p_l = p_a$. In the simulations in this section, they are independent. We also set the success rate of a single CCA check to the same value as the packet loss rate.

Reliability Results ContikiMAC without Retransmissions Given the importance of the packet size, we use two different packet sizes, namely 67 and 90 bytes. In the first experiments, we look at ContikiMAC without retransmissions. The results are depicted in the Figures 3.5 and 3.6. The results show that the model and simulation results are indeed very similar with only small discrepancies between simulation and model.

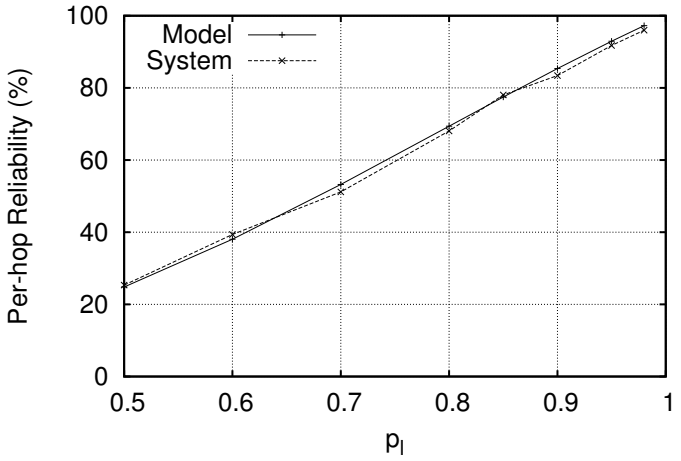


Figure 3.6: ContikiMAC per-hop reliability with 67 bytes packet size.

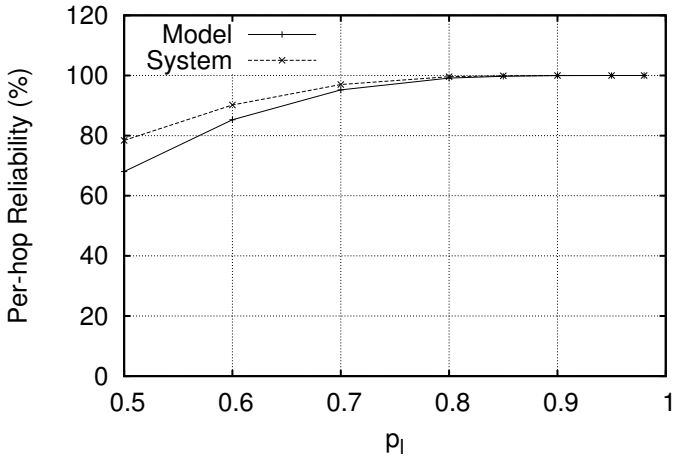


Figure 3.7: ContikiMAC per-hop reliability with 67 bytes packet size.

Reliability Results ContikiMAC with Retransmissions In the next experiment, we evaluate the results for ContikiMAC with retransmissions on the CSMA layer implemented in Contiki. The results are shown in Figure 3.7. In comparison to the results above, the deviation between model and simulation is much larger, in particular when the packet loss rate is high. For a packet loss rate of 50%, the difference between model and simulation is around 10%.

The main source of deviation is that the analytic model is more strict in the way it counts success. In the simulation, a success is the reception of the data packet and the loss of the last acknowledgment does not matter, whereas in the analytic model also the acknowledgment needs to be received.

Table 3.3: Results when no acknowledgments are lost.

p_l	CMAC model	CMAC sim	CSMA model	CSMA sim
70	76.90	78.31	99.4	99.76
60	64.33	65.81	97.2	98.63
50	51.85	51.84	93.8	94.61

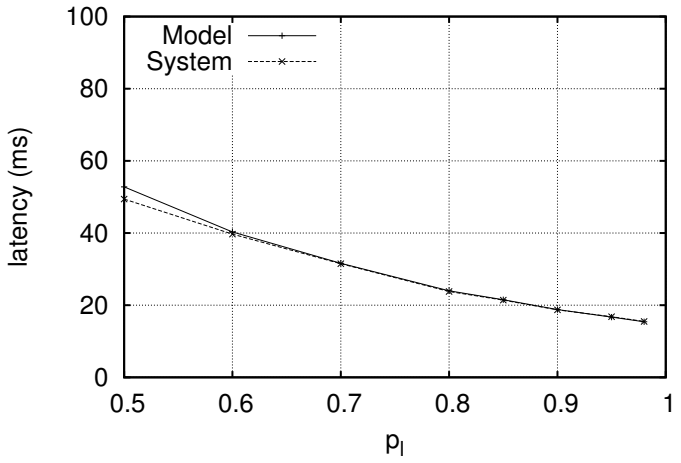


Figure 3.8: ContikiMAC per-hop latency with 67 bytes packet size.

In order to demonstrate that this is indeed the reason for this discrepancy, we perform an experiment where we ensure that only data packets are lost, and hence the last ACK cannot be lost. For this we update Equation 3.8. We depict the results for the experiments with data packets of size 90 bytes in Table 3.3. The results show that model and simulation almost perfectly match when acknowledgments are not lost. Similar results are achieved when the data packets have a size of 67 bytes. These results show that the main source of deviation is indeed that the analytic model is more strict in the way it counts success.

Latency Results ContikiMAC with Retransmissions In the next experiment we verify the latency for single-hop ContikiMAC similar to reliability. Figure 3.8 depicts the latency results from both simulation and the model. The results show very little discrepancy between simulation and model results.

Verification in the COOJA simulator with Interference

In this section, we validate the model under interference. Since it turned out to be very difficult to control the interference in a real environment, we decided to use the COOJA simulator again. COOJA contains so-called interferer nodes, written in Java, that create continuous interference on certain channels. We modified the interferer to create semi-periodic interference as Boano

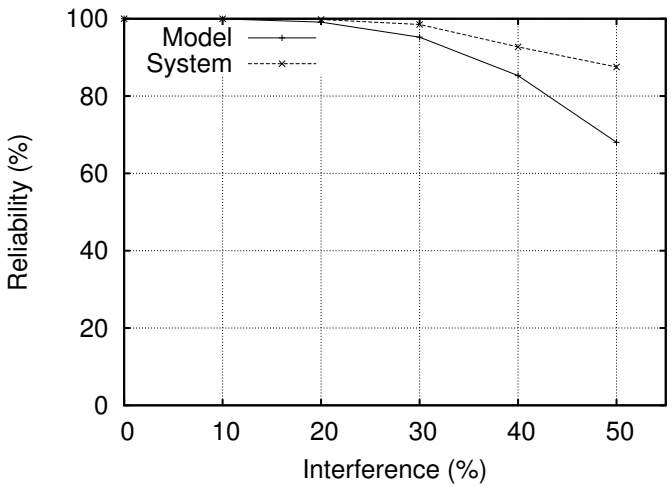


Figure 3.9: ContikiMAC per-hop reliability with 67 bytes packet size under interference.

et al. have done earlier [8]. We used the same application as in the experiments above where one sender transmit packets with a packet transmission interval that is uniformly distributed between one and two seconds. In the experiments in this section, we use packets of size 67 bytes. In order to obtain the input parameters p_l and p_a, we make measurements with Contiki's NULLMAC MAC layer instead of deriving these values from fast RSSI sampling which is not possible in COOJA.

We depict the results in Figure 3.9. The results of the simulation are very similar to the ones shown in Figure 3.7. Also in this experiment, the main source of deviation is that the analytic model is more strict in the way it counts success as discussed above. The major reason for the similarity with Figure 3.7 is that the overall packet loss rate p_l is similar even though the distribution is different: in this experiment we have semi-periodic interference while we had independent packet losses in Figure 3.7. ContikiMAC offers several possibilities to recover from packet loss. Within one wake-up cycle there are usually two chances to transmit and the CSMA mechanism on top of ContikiMAC offers retransmissions in more wake-up cycles. Therefore, ContikiMAC should be robust against most disturbances—unless interference always occurs when the receiver wakes up, which happens periodically.

3.3 Estimation of Packet Reception Rate

For the design of dependable and efficient Wireless Sensor Networks (WSNs) it is essential to estimate achievable Packet Reception Rates (PRR) in the deployment environment. Their knowledge can be used both to fine-tune communication protocols and to evaluate if network performance is sufficient to support a given application. For example, one can employ PRR estimations to customise a TDMA protocol such that sufficient spare capacity is provided for potentially needed retransmissions. Similarly, these estimations can be used to calculate end-to-end data delivery rates, an essential step to judge if an application can meet its performance

requirements.

Besides the used packet size, achievable *PRR* depends to a large extent on the presence of radio interference in the deployment area as WSNs operate in license-free ISM bands and share the radio spectrum with other wireless technologies. This problem is especially relevant in the $2.4GHz$ frequency space as wireless sensor nodes need to coexist with IEEE 802.11 (Wi-Fi) devices which transmit at higher power levels [5]. Estimating the achievable *PRR* is not trivial as it depends on the specific interference patterns at the network deployment site. Some previous work estimated the *PRR* by making general assumptions regarding the nature of interference expected at the target area [25]. Such methods, however, are often inaccurate as the actual encountered interference deviates from the chosen general interference model. Other previous work made use of test measurements on links to estimate achievable *PRR* in a later deployment [34]. However, such techniques are tailored to specific protocols and results cannot be generalised.

In this section we present a novel measurement-based method to estimate achievable *PRR* for specific deployment areas based on the characteristics of interference. We use sensor nodes to capture the interference patterns at a given location by measuring how long the channel remains idle and by computing the corresponding idle distribution. We define an *idle period* as a time interval in which the received signal strength (RSS) of a radio transceiver remains below a given threshold R_{Thr}, indicating that no harmful interfering source is active. We then measure the Probability Distribution Function (PDF) of idle period lengths, referred to as *IDLE-PDF*. As we will show, the *IDLE-PDF* can be captured efficiently using resource-constrained motes, and can be used to estimate *PRR* with very high accuracy. We further present an extensive evaluation of the proposed method focusing on two different aspects. First, we analyse the cost of obtaining the *IDLE-PDF* using off-the-shelf sensor motes and show that, with a surprisingly short measurement duration, we can obtain a sufficiently detailed interference model. Second, we analyse the *PRR* prediction accuracy of the proposed prediction method using different interference scenarios. We show that the proposed *PRR* prediction method has a high accuracy; estimated *PRR* and actual measured *PRR* differ on average by only 3.2%. Specifically our contributions are :

- ***PRR* estimation using the *IDLE-PDF***: We introduce the *IDLE-PDF* as efficient metric for capturing detailed interference patterns. We also describe methods for *PRR* estimation based on the *IDLE-PDF*.
- **Measurement tool**: We present a tool for *IDLE-PDF* measurement based on off-the-shelf sensor node hardware. We show that the tool is able to capture detailed interference data with little storage requirement.
- **Evaluation of *PRR* estimation**: We provide an evaluation of the proposed approach in environments with different interference patterns.

The next subsection describes related work for PRR prediction. In Section 3.3.2 we give the theoretical background of our work: we first give a formal definition of the *IDLE-PDF* and we then describe how this distribution can be used to estimate *PRR*. In Section 3.3.3 we discuss how the *IDLE-PDF* is captured in a deployment area and we describe the dependency between measurement effort and measurement accuracy. In Section 3.3.4 we analyse the efficiency and accuracy of the proposed method using several interference scenarios.

The work documented within this section has been accepted for publication in IEEE International Workshop on Practical Issues in Building Sensor Network Applications (IEEE SenseApp'14) in conjunction with IEEE LCN 2014 [16].

3.3.1 Related Work

The aim of our work is to measure interference *before* deployment and to use the measurement for *PRR* prediction.

There has been a vast body of work on interference measurement. However, the existing work generally does not aim to measure pre-deployment interference to predict achievable *PRR*. For example, existing work has addressed interference measurement for the purpose of interference classification [44] [14], to generate realistic interference for testbeds [10] and to select transmission channels in interference scenarios [23] [30].

A small body of work has similar aims to our work which we discuss in detail in the next paragraphs.

Huang et al. [25] have carried out a statistical analysis of interference traces and presented a generic model that characterizes the white spaces in Wi-Fi traffic. They use this model to estimate *PRR* in dependence of packet size and use this information to schedule transmissions such that delivery ratio is maximised, following the idea by Chowdhury and Akyildiz [17]. Differently from our approach, these two works rely on a generic interference model, whereas we use interference information collected at the deployment area to construct a specific interference model that is not bound to a specific technology (e.g., Wi-Fi).

Shariatmadari et al. [39] have proposed a method for *PRR* estimation based on interference measurements. It is assumed that data on a link will be transmitted with a fixed rate. A receiver measures periodically (using the expected data transmission frequency) the observed interference. The measurement is used to estimate the achievable *PRR*. The method is used to rank channels using *PRR* as link quality metric. This work differs from ours in many aspects. First, the interference measurement is not based on probability distributions. Second, the estimation technique requires assumptions regarding traffic patterns used in the network. Third, there are no guidelines about how long before deployment the interference should be characterised.

Pöttner et al. [34] characterise an interference environment by values B_{min} and B_{max}, and send a trail of test packet transmissions on a link. B_{max} describes the largest number of subsequent transmission failures while B_{min} describes the maximum number of subsequent successful transmissions. The two values can be used to configure a communication protocol such that one can give transmission guarantees for the measured interference. Similar to our work, interference patterns are determined by measurement carried out before deployment. However, the recorded interference patterns are only useful to configure protocols using transmission spacing and packet lengths as used in the measurements (i.e. specific TDMA protocols). The method presented in this section, instead, follows a more general approach.

In this work we also describe a method for updating at runtime the interference measurement collected before the deployment. This is necessary as the interference environment may change over time. The related work outlined previously does not provide this feature and relies on the assumption that the interference does not vary significantly over time. A notable exception is the work by Brown et al. [15] which describes a method to update initially measured B_{min}/B_{max}

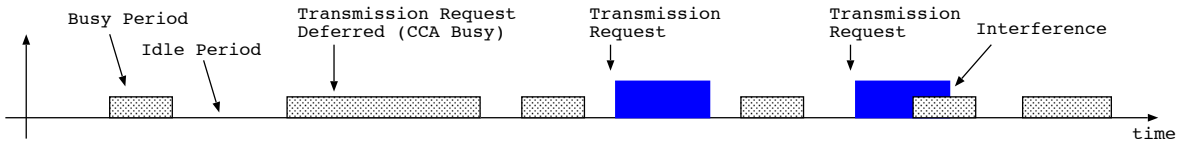


Figure 3.10: Example sequence of idle and busy periods. The first transmission is successful while the second one is subject to interference. A fixed packet length L is used.

measurements while the sensor network application is running.

3.3.2 Packet Reception Rate and Interference

In this section we give a definition of the *IDLE-PDF* and we describe how this distribution can be used to estimate *PRR*. First we describe a closed form solution to compute the *PRR* from the *IDLE-PDF*. As the closed form solution has its limitation with arbitrary shaped *IDLE-PDF* distributions we then describe a solver based on Monte Carlo simulation.

The *IDLE-PDF*: Interference levels can be measured by sampling the energy level in a transmission channel over time. If the sampled energy level is above a given threshold R_{Thr}, a packet transmission would be destroyed by concurrent activities in the frequency channel. This is an approximation of the interference process but is a reasonable assumption for the work presented in this deliverable (as shown by our Evaluation). Thus, interference can be represented as a sequence of idle (free channel) and busy (ongoing activity in the medium) periods of different length. Using such a recorded interference trace it is then possible to analyse success rates of packet transmissions. For example, to estimate *PRR*, a number of transmissions can be randomly placed in the recorded interference trace and the success or failure of these transmissions can be evaluated. Unfortunately, due to the volume of data it is not possible to store reasonably long interference traces of this type on practical measurement systems.

To reduce the required storage space for interference traces we decide not to store the actual sequence of idle and busy periods and their respective length. Instead, we record the distributions of observed busy and idle period lengths. This measurement does not allow us to reproduce the exact measured interference trace but it allows us to produce an interference trace which exhibits the same statistical distributions of idle and busy periods and period lengths. Thus, the recorded distribution of idle and busy period lengths (*IDLE-PDF* and *BUSY-PDF*) can be used for analysis of *PRR* instead of using an actual recorded interference trace. The *IDLE-PDF* and *BUSY-PDF* can be measured over arbitrary time period with a fixed storage volume requirement. Thus, measuring *IDLE-PDF* and *BUSY-PDF* is a practically feasible method for collection of detailed interference patterns.

A transmitter usually performs a Clear Channel Assessment (CCA) before attempting a transmission. If a transmitter aims to send a packet during a busy period (i.e., when there are other ongoing activities in the channel stronger than R_{Thr}), the CCA would return false, and the transmission would be deferred. If the transmitter aims to send a packet during an idle period (i.e., while the channel is free), the CCA will return true and the transmission is

started. The latter can only complete successfully if the remaining idle period is longer than the duration necessary for packet transmission. The CCA test assumes that communicating nodes are within the same collision domain, such that the channel state for the transmitter is the same as the receiever. This outlined transmission behaviour is illustrated in Figure 3.10. As transmissions are not attempted in a busy environment only the *IDLE-PDF* is necessary for analysis of the *PRR*. Hence, in the remainder of this section we focus on measurement and analysis of the *IDLE-PDF*. We use the mathematical notation $p_i(x)$ to refer to the probability distribution function *IDLE-PDF*.

In this work we only consider the raw packet transmission process and do not assume a particular Medium Access Control protocol (MAC) behaviour which may have an impact on the aforementioned assumptions. We do not assume particular transmission scheduling policies and assume random placement of transmissions within an idle period. For example, if a 1-persistent Carrier Sense Multiple Access (CSMA) strategy would be used idle and busy periods would need to be considered. However, we believe the presented assumptions are compliant within many used WSN MAC protocols.

In Section 3.3.3 we will describe in detail how a suitable *IDLE-PDF* can be obtained from measurements taken using off-the-shelf sensor nodes, and discuss the effects of different durations and resolutions of the measurement. For the remainder of this section, we assume that we obtained an *IDLE-PDF* that gives an accurate representation of interference in the target area.

Closed Form Solution: The start of a packet transmission falls within an idle period as we assume CCA before a transmission attempt. The packet transmission has a probability P of completing successfully. This probability depends on the length of the idle period and as well on the packet length L. Figure 3.10 illustrates two transmissions, one successful while the other transmission is unsuccessful due to interference. A transmission is equally likely to start at any point within an encountered idle period of length y and a transmission will complete successfully if it starts at any point before $y - L$ in the idle period. The probability $P_a(y)$ of a successful transmission in an idle period of length y can therefore be computed as:

$$P_a(y) = \frac{y - L}{y} \qquad \forall y > L \tag{3.24}$$

The probability $P_b(y)$ of attempting a transmission in an idle period of length y is dependent on the measured *IDLE-PDF* $p_i(y)$ and is given as:

$$P_b(y) = \frac{y}{E[y]} \cdot p_i(y) \tag{3.25}$$

$E[y]$ is the expected value of random variable y. The overall probability P of successfully transmitting a packet in an interference environment characterised via the *IDLE-PDF* can be calculated by summing the products of the probability of a given idle size with the probability of the transmission being successful for this size:

$$P = \int_L^\infty P_a(y) \cdot P_b(y)\, \mathrm{d}y \tag{3.26}$$

In many observed interference scenarios, the *IDLE-PDF* follows an exponential distribution. In this case the *IDLE-PDF* $p_i(x)$ is given as:

$$p_i(x) = \begin{cases} \lambda \cdot e^{-\lambda \cdot x}, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{3.27}$$

With Equation 3.27, Equation 3.26 becomes:

$$P = \int_L^\infty \lambda^2 \cdot e^{-\lambda \cdot y} \cdot (y - L)\, \mathrm{d}y = e^{-\lambda \cdot L} \tag{3.28}$$

This closed form solution using an exponential distribution is useful for quick *PRR* estimation. An exponential function can be fitted to the measured *IDLE-PDF* to determine λ. Using Equation 3.28 and a packet length L provides an estimation of the achievable *PRR*. The computational effort of this method is dominated by the used curve fitting algorithm. This method is computationally cheap compared to the Monte Carlo Method we describe next.

Monte Carlo Solver: In some environments, a measured *IDLE-PDF* $p_i(x)$ may not be approximated by a well known distribution function. In this case the aforementioned closed form solution as described by Equation 3.26 is not straightforward to solve for P. To solve the equation in these cases we use a Monte Carlo approach.

The *IDLE-PDF* distribution is used to create a trace of duration T seconds consisting of idle periods only. As transmissions do not occur in busy periods these do not have to be included in this trace. Then, for N packets with length L a random (with uniform distribution) transmission start point $t < T$ is selected within the created trace. Transmission success or failure of each packet is recorded. This process is repeated for R runs and then the average transmission success rate across all runs is calculated and, thus, the packet delivery probability for packets of length L in presence of interference with *IDLE-PDF* $p_i(x)$ is determined.

The accuracy of this approach depends on the computational effort invested which is given by the number of tries specified via N and R. To estimate the effort necessary for obtaining sufficiently accurate results we compare the closed form solution with results obtained using the Monte Carlo method. A distribution is created using Equation 3.27 with a $\lambda = 100$. This distribution is used in the Monte Carlo Solver and its output is compared with the results given by Equation 3.26 (Using packet sizes of $L = 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100$ bytes). The solver is configured with $T = 100$, $N = 1000$ and $R = 100$. The results produced by the simulator are within 0.44% on average to those of the closed form solution and have a maximum *PRR* difference of 1.42%. When $R = 50$ and $R = 10$ is used the average difference between predicted *PRR* by the model and the solver increases by 0.04% and 0.05% respectively. This suggests that reasonable accuracy can be obtained with limited numbers of simulation runs.

3.3.3 Capturing the *IDLE-PDF*

In this section we discuss *IDLE-PDF* measurement considerations and describe a measurement tool based on standard sensor node hardware.

The *IDLE-PDF* Capture Tool: The *IDLE-PDF* is captured in the deployment area. Obviously it would be possible to deploy dedicated equipment to carry out this measurement which would result in undesirable additional deployment costs. Thus, we aim to carry out interference measurements with the same sensor node hardware used for the final application. As a result, accuracy and detail of interference measurements are limited by the available hardware. However, a benefit is that interference can be measured at the exact position where it occurs and with the same hardware impacted by the interference signal.

Modern IEEE 802.15.4-compliant radio transceivers provide the capability of reading the received signal strength (RSSI) in absence of packet transmissions. When sampled at a high rate, these measurements can be used to quantify the level of interference at a given node. Following the approach used in [10, 11] (i.e., by boosting the CPU speed, optimizing the SPI operations that are used to interface the radio), one can indeed perform a high-speed sampling of the RSSI register on Maxfor MTM-CM5000MSP nodes up to $50kHz$.

To capture the distribution of idle and busy periods, we build a Contiki application that carries out RSSI sampling as described previously and computes statistics on the idle and busy periods on a specified channel until a set amount of RSSI samples R is collected (in our application we use $R = 12.5$ million samples for a $5min$ sampling window). We make sure every interrupt is disabled and that no other process can interfere with our operations since we need to sample at the highest possible rate. This way, we achieve a sampling rate of one RSSI value approximately every $24\mu s$.

We introduce an RSSI threshold R_{Thr} defining whether a channel is idle or busy (RSSI values above R_{Thr} identify a busy channel, RSSI values below R_{Thr} identify an idle channel). We count the number of consecutive RSSI readings in which a channel remained idle or busy and as soon as the current channel state (idle, busy) differs from the previous one, we increment a field in one of two arrays $A_{idle}[i]$ and $A_{busy}[i]$. Each array holds 16 fields which correspond to different ranges in length of an idle or busy periods. On recording a period, the length of the period is compared to 16 variable length running ranges and the corresponding field within either the idle or busy array is incremented. Because of the limited memory of the nodes, we truncate the maximum duration of an idle or busy period to $100ms$.

Storage requirements. In our implementation we chose to quantise an idle or busy sample (1 bit), counting the length of a period in sample units and store this recorded value using 16 variable length ranges. An alternative to this would be to store a trace of recorded samples as either RSSI values or a count of consecutive samples of a given state (idle, busy). Whilst some compression of a trace may be possible (e.g. run-length encoding) storing a distribution, even with no compression, requires significantly less storage. For comparison, during a typical $5min$ sample with no artificial interference 25722 idle and busy periods are present. Storing each period as a 16-bit integer would require approximately $101Kbyte$ of memory for both distributions. This compares to just $128byte$ for both distributions storing 16 ranges of 32-bit values.

Limitations. The achieved sampling rate is sufficiently high to identify the short instants in which the radio medium is idle due to the Inter-Frame Spaces (IFS) between 802.11 b/g packets as shown by Hauer et al. [23, 24]. Although the achievable $50kHz$ sampling rate is sufficient to detect IEEE 802.11b frames, it may not be enough to capture all 802.11g/n frames (the minimum size of a Wi-Fi packet is 38 bytes, and the maximum speed of Wi-Fi transmissions is 11, 54, and $150Mbit/s$ for 802.11b/g/n standards, respectively).

Updating the *IDLE-PDF* at Runtime: The environment in which a sensor network is deployed is typically dynamic and may change over time. In this case the *IDLE-PDF* captured before deployment may become invalid once the network becomes operational. It is therefore useful to measure the *IDLE-PDF* periodically at runtime to either verify that the *IDLE-PDF* used for network and application configuration is still valid or to produce an entirely new *IDLE-PDF* in case the surrounding environment has radically changed. We refer to this process as *runtime assurance*.

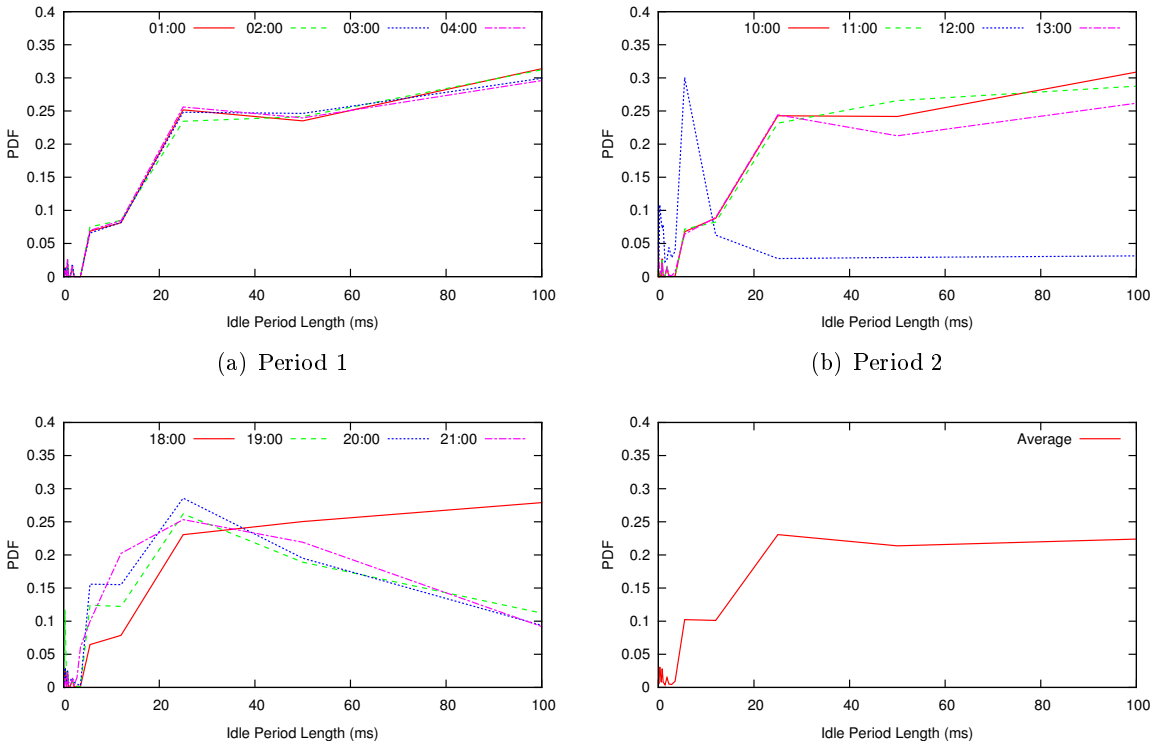
Runtime assurance can run alongside normal WSN software on nodes within a deployment. Periodically, a sensor node may hand control to runtime assurance which then carries out an interference measurement. The same limitations regarding measurement time, duration and location applies as discussed in the previous paragraphs. Runtime assurance may execute during times a node would normally enter a sleep state in order to ensure interference measurements do not impact on normal node operation. The software used for measurement is the same as the one used for capturing the *IDLE-PDF* before deployment (see previous paragraph).

Measurement Considerations: It is our aim to capture interference patterns in a deployment area such that the measurement allows us to estimate packet delivery probabilities during later network operation. Independent of the applied measurement technique this approach can only be successful if the interference impacting on the deployed network is present during measurement.

When to measure. Obviously, measurements must be carried out when a representative interference is present in the deployment. In most deployment areas interference can vary much over time. For example, in an office building Wi-Fi usage during the day will create much interference while during night little activity will be observed. Similarly, interference measured during weekends is typically lower than during work days.

Figure 3.11 illustrates the impact of measurement time on the shape of the recorded *IDLE-PDF*. Figure 3.11(a) shows 4 *IDLE-PDFs* captured at night in our university building. Each *IDLE-PDF* is captured over a period of $5min$ and all sampling phases are one hour apart. All 4 *IDLE-PDFs* have a very similar shape and any of the 4 captured *IDLE-PDF* would be a good representation of the interference environment present at night. The 4 *IDLE-PDFs* shown in Figure 3.11(b) are captured by the same device during daytime. For most times the interference is similar to the night time interference; however, the *IDLE-PDF* recorded at 12:00 clearly differs. During this *IDLE-PDF* sampling, an increase in interference (most likely a very active Wi-Fi client) is present, and the *IDLE-PDF* distribution is shifted towards short idle periods. This increases the chances that a transmission is not completed before interference occurs, likely leading to a loss of the transmitted packet. The 4 *IDLE-PDFs* shown in Figure 3.11(c) are captured in the evening. The *IDLE-PDF* at 18:00 is comparable to the PDF at night while the other 3 *IDLE-PDF* show higher levels of interference.

Therefore, the point in time in which the measurement is carried out has a strong impact on the obtained *IDLE-PDF*. However, it depends on the users intention on how to use the *IDLE-PDF* which informs the approach to take to deal with the observed temporal changes. If the user is interested in long-term average packet delivery reliability figures all measured *IDLE-PDF* at different points of time can be aggregated into one single average-case *IDLE-PDF*. The average-case *IDLE-PDF* for the 12 example PDFs shown in Figure 3.11 is given



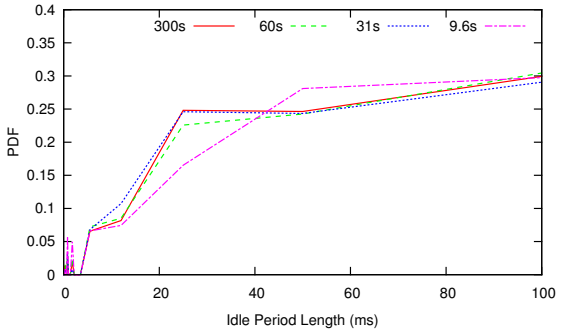
(a) Period 1

(b) Period 2

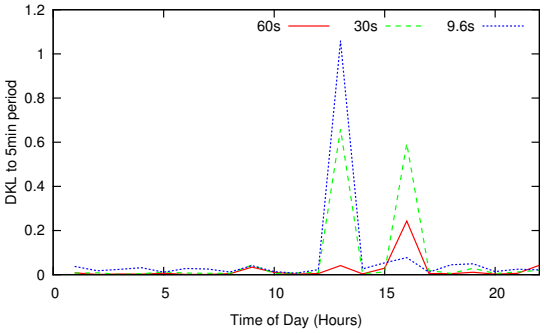
(c) Period 3

(d) Aggregated *IDLE-PDF*

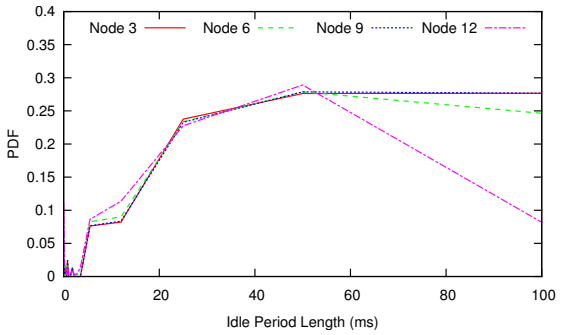
Figure 3.11: *IDLE-PDF* captured by four nodes on Channel 19 with 5*min* sampling periods at different times of the day. The first shows those captured at night; the second during working hours; the third during the evening. The last shows the aggregation of all three periods.



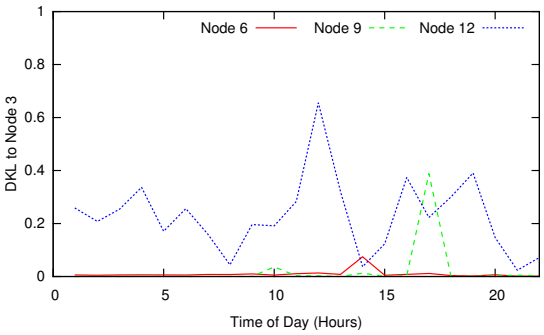
(a) Different Sampling Periods



(b) D_{KL}, Sampling Periods



(c) Different Sampling Locations



(d) D_{KL}, Sampling Locations

Figure 3.12: *IDLE-PDF* on Channel 19. The first two figures describe the impact of using different sample durations. The second two figures describe the impact of using different sample locations.

in Figure 3.11(d). If the user is interested in worst-case packet delivery reliability that might be encountered in the deployment it would be better to select the worst-case *IDLE-PDF* (the *IDLE-PDF* leaving to the worst *PRR* prediction) as representation of the interference situation. For the example given in Figure 3.11 one would select the *IDLE-PDF* from 12:00 in period 2.

How long to measure. As previously shown it is important to consider the point in time when to measure the *IDLE-PDF*. The next important aspect to consider is the necessary sampling duration at each sampling time. Shorter sampling periods would be beneficial as shorter sampling durations would require less energy. We intend to use sensor nodes as sampling devices which often rely on battery power.

Figure 3.12(a) shows an *IDLE-PDF* captured at one point in time where different sampling durations are used ($300s, 60s, 31s, 9.6s$). As it can be seen, the shape of the *IDLE-PDF* is not very dependent on the sampling duration. Formally, the Kullback-Leibler (KL) divergence can be used as a measure of the information lost when approximating the probability distribution with high sampling duration by a probability distribution with lower sampling duration. Small values of the KL measure are good as little information is lost due to lower sampling duration. Figure 3.12(b) shows the KL divergence with $300s$ sampling duration as base over an entire day. As it can be seen the KL divergence is very low except at two measurement points. At these points, as expected, the KL divergence is higher for low sampling durations. This analysis shows that high sampling durations provide little benefit in terms of probability distribution accuracy of the *IDLE-PDF*. Combining this insight with the previous aspect on "when to measure" it seems more beneficial to distribute a large number of small sampling periods over a long time period instead of using one long sampling period at one point in time.

Where to measure. An idle period is defined as a time duration in which the energy detected in a transmission channel stays below a given threshold R_{Thr}. If interference is measured in one place it is obviously not guaranteed that interference measured at a different place in the same area is similar. Thus, most accurate results can be achieved when measuring the *IDLE-PDF* with the node that is later used as receiver for the packets for which the delivery success rate is computed. However, it is possible to record interference at one location and then use this measurement to predict the outcome of transmissions at different locations in the vicinity of the measurement spot.

Figure 3.12(c) shows the *IDLE-PDF* captured at 4 different locations at the same time (the used capture devices are spaced $3m$ apart). As can be seen, the measured interference patterns are similar. Figure 3.12(d) shows the KL divergence with the distribution captured at one node as base over an entire day. As it can be seen, the interference patterns at two nodes are almost identical (very small KL value) to the reference node while one node (node 12) experiences a slight variation in terms of measured *IDLE-PDF*. These experiments show that it is not necessary to capture the *IDLE-PDF* at every single node in the deployment. Instead, it is sufficient to capture the distribution once for an interference area (However, in practice it might be difficult to decide what these areas are).

3.3.4 Validation and Verification (D-2.3)

For evaluation we use Maxfor MTM-CM5000MSP sensor nodes running the Contiki operating system [20]. The nodes are used for both the transmission of data packets and to capture the *IDLE-PDF* using the software described in Section 3.3.3. All experiments are carried out in a

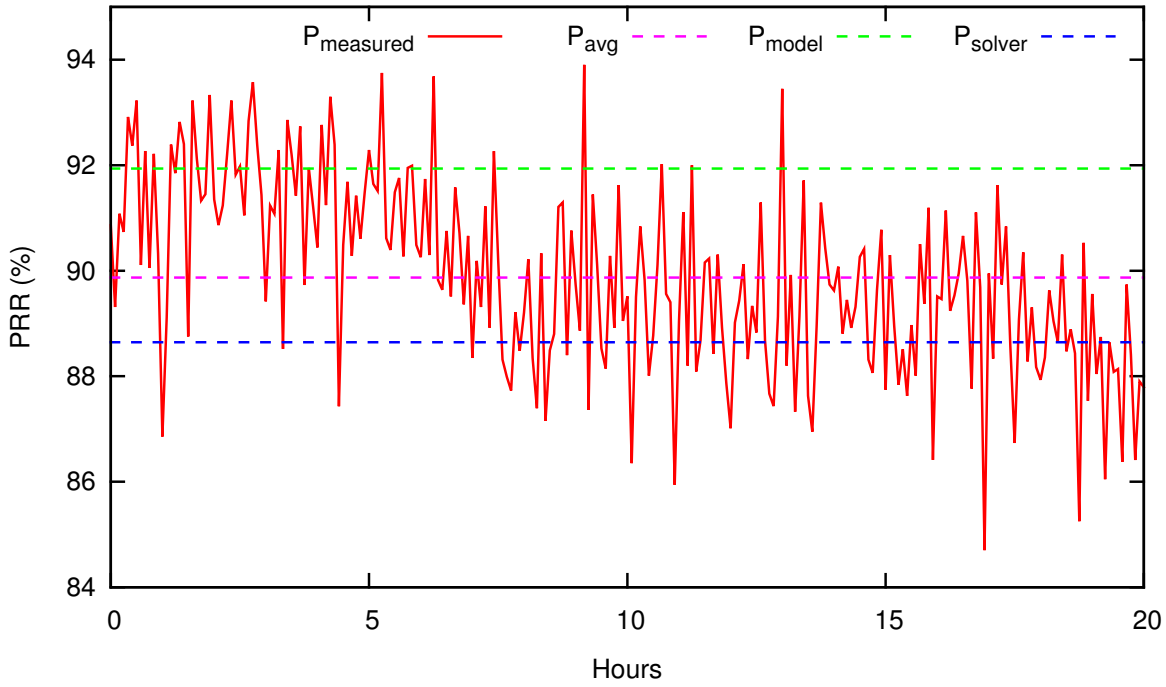


Figure 3.13: Predicted *PRR* using the model and solver and actual *PRR* for a packet size of 5 under background interference.

university office environment in a room of approximately $5m^2$, vacated for the duration of each experiment. During packet transmission tests, the sending sensor node transmits 16 packets per second with size L randomly chosen from 12 fixed packet sizes to a paired receiver on a specific channel. Inter-packet spacing is approximately 1/16 of a second with a small amount of jitter introduced to avoid any potential synchronisation effects. Each sender and receiver pair are placed approximately $3m$ apart around the centre of the room. In the experiments we use two types of interference. Firstly, Wi-Fi networks operating on channel 1 in the building carrying university network traffic are used as a form of uncontrolled background interference, the access point is located approximately $4m$ from the room perimeter. Secondly, we use the network traffic generator *Iperf* between two desktop PCs interconnected via Wi-Fi 802.11g on channel 3 to produce controlled Wi-Fi interference. The Wi-Fi network used to generate controlled interference operates on an isolated network using a channel that is not occupied by other access points. The access point is placed in one corner of the room with connection to the first PC whilst the client adapter is placed in the opposite corner connected to the second PC.

Model Based *PRR* Prediction: In our first experiment we use interference introduced by the university Wi-Fi network. We record the *IDLE-PDF* over a period of 24 hours where in each hour the *IDLE-PDF* is recorded with a sample period of $5min$. The average *IDLE-PDF* distribution is then calculated from the 24 individual recorded distributions. Thus, an

interference profile is created which describes the average interference level over $24h$. We use the exponential model introduced in Section 3.3.2 to model this measured average *IDLE-PDF*. We then transmit packets of size $L = 5bytes$ over a duration of $20h$ and record the achieved *PRR* which is then compared to the model predicted *PRR*. To determine parameter λ which characterises the exponential model, we fit the model to the measured data using a total least squares fit. The model is expected to give a *PRR* prediction close to the observed *PRR* where the observed *PRR* may oscillate around the predicted value.

The results of the experiment are shown in Figure 3.13. The measured *PRR* (sliding window of $5min$) is shown over the experiment duration of $20h$. The model predicted *PRR* matches very well the observed values. The *PRR* records oscillate as expected around the predicted value. The overall measured *PRR* over the entire duration of the experiment is $P_{measured} = 89.9\%$ while the model prediction is $P_{model} = 91.9\%$.

In this case, the exponential model produced from the captured distributions can be used to estimate the achievable *PRR* reasonably well. However, in many cases the observed distributions do not follow exactly an exponential shape and therefore the model will produce less accurate approximations. More accurate results can even be obtained in the case presented here by using the Monte Carlo Solver described in Section 3.3.2. The solver will always produce more accurate results as it uses the actual measured *IDLE-PDF* instead of an approximation as used by the exponential model. Figure 3.13 includes the *PRR* prediction obtained using the solver which provides a *PRR* prediction of $P_{solver} = 88.6\%$. The prediction of the solver is $1.3pp$ below the actual achieved *PRR*; in case of the model the prediction is $2pp$ above the achieved *PRR*.

It has to be noted that the interference distribution used for *PRR* prediction was recorded during $24h$ before the experiment was run for $20h$. The interference is introduced by packet transmissions on the university campus network. This shows that interference characteristics can be very stable over long periods of time. In situations where this would not be the case, the *IDLE-PDF* might need to be updated during network operation as we have described in Section 3.3.3.

Monte Carlo Simulation Based *PRR* Prediction: We now use our network traffic generator to produce six different levels of Wi-Fi interference. We vary the rate of the generated interference between $500kbit/s$ and $10Mbit/s$, and record the *IDLE-PDF* over a $1h$ period measuring two distributions with a sample duration of $5min$ and using the average of these two distributions. We then calculate the expected *PRR* for different packet lengths L in presence of the different interference intensity levels using the Monte Carlo Solver. Thereafter we transmit packets of the different sizes L in the environment exposed to the different interference intensity levels. For each intensity we record the achieved *PRR* for each packet size L over a $2h$ window. The aim of this experiment is to evaluate achievable *PRR* prediction accuracy.

The results of the experiment are shown in Figure 3.14. The figure shows that generally for lower interference intensities the predicted *PRR* closely matches that of the recorded values across all packet sizes. *PRR* predictions under $500kbit/s$ of interference have an average error (average distance between predicted *PRR* and actual *PRR* over all packet sizes) of 2.32% and a worst-case error (maximum distance between predicted *PRR* and actual *PRR* for all packet sizes) of 4.97% as described in Table 3.4. As the interference intensity increases, both the

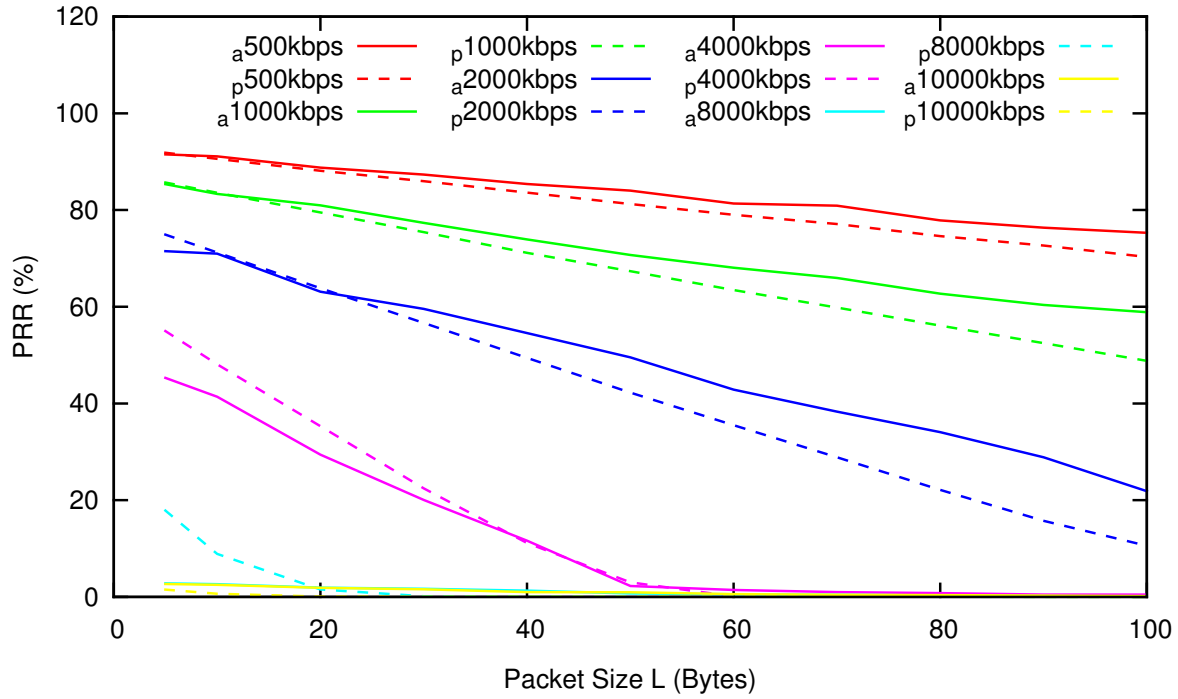


Figure 3.14: Predicted and actual *PRR* for different packet sizes under increasing interference (predicted prefixed with p and actual prefixed with a).

Table 3.4: Average and Worst-Case *PRR* Prediction Error. The overall average of all average errors is 3.2%.

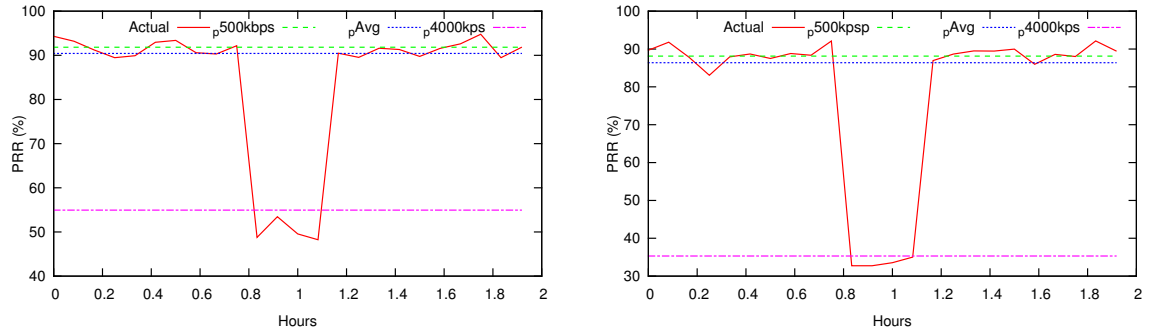
Interference Level	Worst-Case Error	Average Error
$500kbit/s$	4.97%	2.32%
$1000kbit/s$	10.07%	4.13%
$2000kbit/s$	13.10%	6.62%
$4000kbit/s$	9.66%	2.7%
$8000kbit/s$	15.07%	2.41%
$10000kbit/s$	1.85%	0.91%

average error and worst-case error increases. At an interference level of $2000kbit/s$ the worst-case is 13.10% and the average is 6.62%. The error eventually falls as the *PRR* approaches 0%, this can either be due to the increased level of interference as seen with $10000kbit/s$ of interference intensity or with increased packet transmission size at lower intensity as with $4000kbit/s$.

***IDLE-PDF* Variations:** In Section 3.3.3 we have shown that the *IDLE-PDF* measurement must be carried out at a point in time at which the interference of interest is present. In this experiment we show the impact of changes in the *IDLE-PDF* on predicted packed delivery rates. In this experiment we create controlled Wi-Fi interference in the testbed which varies over time. The experiment consists of transmission of $0.5Mbit/s$ for a duration of $50min$. Thereafter we increase the level of interference by increasing the transmission speed on the Wi-Fi network to $4Mbit/s$ for a duration of $20min$. Finally we drop the level of interference back to $0.5Mb/s$ for a duration of another $50min$. Before starting the experiment we create the interference levels of $0.5Mbit/s$ and $4Mbit/s$ and record both the *IDLE-PDFs* using a sampling duration of $5min$ twice over 2 hours taking the average for each. These two *IDLE-PDFs* are used with our Monte Carlo Simulator to determine expected *PRR* for both interference environments.

Figure 3.15 shows the results of the experiment for two packet sizes ($L_1 = 5byte$, $L_2 = 20byte$). The achieved *PRR* over time (using a $5min$ sliding window) is shown. In addition the *PRR* using the two different *IDLE-PDFs* is shown as well. It can be seen that the achieved *PRR* is dropping in the period of increased interference. For both interference levels very accurate *PRR* are predicted when using the respective *IDLE-PDF* (a maximum difference in *PRR* of 5.1% for 500Kbps and 6.7% is observed). The experiment shows also that it is essential to capture a suitable *IDLE-PDF* for prediction of achievable network performance (see discussion in Section 3.3.3). If it is for example important to predict worst-case *PRR* the *IDLE-PDF* representing high interference levels would be suitable. However, this would require that the worst-case *IDLE-PDF* was observed before network deployment.

Reactive Interference We have shown that the variable packet size model can yield very accurate predictions. This is based on the assumptions that whilst interference may slightly vary overtime, the interference captured is similar to that experienced by the nodes at runtime.



(a) Packet Size $L = 5byte$

(b) Packet Size $L = 20byte$

Figure 3.15: Predicted and measured PRR in an environment with variable interference. Wi-Fi traffic is used as source of interference ($0.5Mbit/s$ for a duration of $50mins$, $4Mbit/s$ for a duration of $20min$, $0.5Mbit/s$ for a duration of $50mins$).

In the last sub-section we demonstrated that in many environments interference patterns may change significantly in which case multiple model instances are required. A second assumption was also made in that the signal generated by the node at run time has little to no effect on existing interference. When both of these assumptions hold, the protocol model can produce highly accurate predictions.

In some environments these assumptions may not hold. Interference from communicable sources such as WIFI or Bluetooth can react to interference to improve the quality of their communication. If the energy output from our node transmissions is detected by such systems, it will be considered as interference and these systems will react accordingly. During our lab experiments so far, we have used minimal transmission power settings to emulate longer links, this has had the side effect of reducing the likely hood that our transmission have had any effect on the WIFI interferer. Boosting the transmission power so the WIFI interferer is more likely to detect our signals will generated a reaction. Such systems can react in a number of possible ways including; to back-off, to change encoding or modulation techniques or to modify transmission frequency. Each of these reactions will have an impact on the interference experienced by the node at transmission time.

Modifications to the interference experienced at run time to the interference measured at by the environmental capture tool will have an effect on the accuracy of the predictions made. Such impact can have a random effect making predictions either too optimistic or pessimistic. Back-offs will increase the likely hood of a packet which would have been interfered will getting through making the predictions too pessimistic. Changes in modulation or increased error coding will increase the size of WIFI transmissions reducing the size of idle periods which would make predictions to optimistic.

Such effects can be seen in Figure 3.16 which depicts the results of an experiment conducted in a research office in Lancaster. Real-world interference from the campus WIFI network was used. A series of tests was performed back to back where for each test the environment was first measured for 5 minutes before measuring PRR for 5 minutes. The transmission power selected

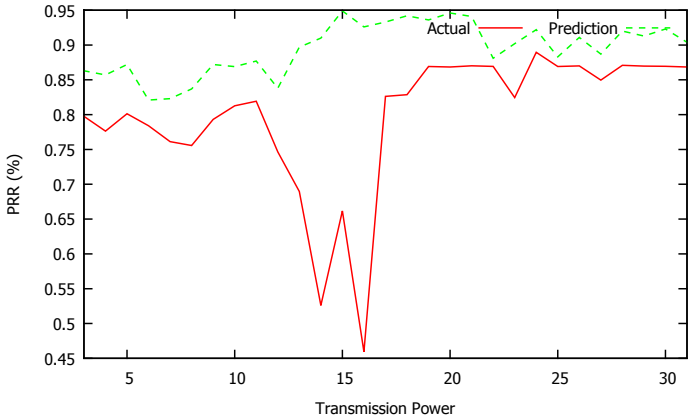


Figure 3.16: Predicted and actual *PRR* for different transmission power level under real-world interference).

was incremented with each test from level 3 to 31. Slight fluctuations in the predictions over the experiment was seen. This is expected as a real-world WIFI network was used whose traffic may fluctuate. The actual PRR in the first instances mirrored the changes in the prediction, the prediction are accurate as seen in the earlier sections. However, at transmission power 8 the difference between the prediction and actual results started to deviate where the measured PRR saw a significant drop. There was a recovery in the measured PRR when the transmission power reached 15. As can be seen from the figure, PRR fluctuates as transmission power increases; we surmised that the drop in PRR may be caused by WIFI retransmission that is corrupting our transmissions. Due the vastly difference in time-scales of WIFI and 802.15.4, multiple WIFI transmissions could occur during the transmission of a single 802.15.4 packet. As transmission power increases, our signal may have then been detected by WIFI and different strategy such as back off may have been implemented. To confirm these observations further work is necessary. At this stage we believe this unpredictable element may limit the use of variable packet size to environments where much of the interference is non-reactive such as in factories or where any reactive system cannot hear the transmissions of our nodes.

3.4 Radio Energy Prediction

The radio hardware in sensor nodes usually has the highest power consumption. The transmission of packets requires the radio to monitor the channel and transmit only during clear periods, whilst reception of packets requires also the channel to be monitored in order to detect the presence of new data, and it is the monitoring of the channel that causes the highest power draw over the runtime of the device.

This is especially true in cases where the radio band in question is also busy with traffic from other sources which are not part of the network. Zigbee, Wi-Fi, and other 2.4 GHz ISM band radios all occupy the same frequency ranges, and will often cause unnecessary wake-up events for the device, leaving the radio perpetually in an *idle listening* state, attempting to read data when none relevant will occur.

Some protocols do attempt to mitigate the effect of idle listening through reducing the check rates, or the duration of the checks made on the channel. ContikiMAC, which uses the radio RSSI level as a channel checking mechanism is significantly affected by persistent idle listening states. In environments without additional radio sources, the noise caused through the RF characteristic of the environment can also significantly extend the idle listening state, and cause unnecessary energy consumption.

A model for predicting how the energy budget of a device will be affected by it's radio behaviour during these idle states is required to allow accurate estimations of the runtime of the device, and to tune the radio behaviour to better hit its target energy budgets. This model should provide a mechanism to:

- Reduce the channel check frequency in busy, or noisy environments
- Increase the channel check rate during known idle spaces.

3.4.1 Protocol Behaviour

In this section ContikiMAC is taken as an example of a modern MAC protocol as a case study. The protocol is examined to provide an understanding of how energy is consumed in a MAC protocol.

ContikiMAC nodes attempt to keep the radio in sleep mode for as long as possible, only waking periodically to check for data on the channel. As the receivers are known to wake at specific intervals, ContikiMAC implements an optimisation known as *Phase Lock* (PL) which attempts to delay transmission until the receiver is expected to be checking the channel. Because this is all done by the transmitter, it has no effect on the energy usage at the receiver.

When waiting for data, nodes sample the channel state twice, 0.5ms apart; if either of these samples return a busy state, the receiver keeps the radio powered up to receive any data from the channel. Transmitters also perform a channel check to determine if the channel is clear to send.

To maximise the chance of successful reception, messages are (re)transmitted 6 times at a fixed interval in phase with the time when the receiver is supposed to be listening, and each transmission is followed by a CCA - this is known in ContikiMAC as *strobing*. Receivers may report an ACK for any strobe, and in doing so terminate the transmitter's strobing sequence - marking a successful transmission.

When a receiver is checking the channel for new data, it performs two distinct operations - shown graphically in Figure. 3.17. These operations are referred to as the *signal detection* or *packet reception* phase.

In the *signal detection* phase, the device performs two CCA energy detection checks (represented as $C = CCA$ in Figure 3.17) each consuming approximately $300\mu s$ of radio on time. If either of these checks sees significant activity, the device will proceed to the second *packet reception* phase, otherwise the device returns the radio to a sleep state.

When performing the *packet reception* phase, the device keeps the radio constantly active to receive any data from the air. First, the device re-runs the CCA check procedure, if any significant energy is detected, the radio is kept active to receive the packet data. Throughout the actual reception, the device checks periodically that data continues to be available, otherwise when enough consecutive checks return an 'idle' channel state, the device considers the channel

empty and abandons the reception attempt and returns to sleep. Alternatively, if a complete packet is received, then the radio can also return to sleep. Each run consumes approximately $620\mu S$ of radio on time.

This sequence illustrates the problems with a busy environment: if a *CCA* check indicates that a channel is 'active' state (in either phase), the device will be kept in an *idle wait* state. In the first phase, this interference causes unnecessary wakeup events. In the second, *packet reception* phase, the interference causes the device to continue to wait for real data, expending energy for the duration. This problem is especially prevalent for nodes co-located with other $2.4Ghz$ spectrum networks, as initially 'valid' data may arrive, but be in a totally different protocol.

3.4.2 Protocol Models

The energy expenditure for a given radio configuration can be calculated by examining how long the radio will be active for using a specific MAC or energy usage scheme.

To determine the *active* and *idle listening* periods for a protocol, a model should describe how the radio behaves during it's operation. As *idle listening* dominates the energy budget usage for the radio, we exclude the active period and any individual transmission and reception costs and simplify the model to only consider the listening period.

A protocol model can be implemented using either a closed form method or through the use of a Monte-Carlo method. The choice depends on the complexity of the protocol and the specific environmental model used. Continuing the use of ContikiMAC as a case study, in this section we will first show how to derive a closed form protocol model for ContikiMAC before showing a Monte-Carlo approach.

Closed Form Method: In this section we describe analytically the channel check behaviour of ContikiMAC in a noisy environment. We develop a closed form solution for the expected radio receiver on time during a channel check sequence (referred to as $E(p)$) by analysing the ContikiMAC channel check state machine as described in Figure 3.17. E is a function of the parameter p which denotes the probability of the channel being busy. p is known by reducing the interference *PDF's* models developed in WP1 to a Boolean busy or idle state, we can determine the *busy/idle time* proportions and estimate the chance of a packet being available for reception. $E(p)$ is composed of three elements –E_{ii}, E_{ib} and E_b)– as follows:

$$E(p) = E_{ii} + E_b + E_{ib} \tag{3.29}$$

E_{ii} represents the case where ContikiMAC's first CCA returns clear followed by a clear result from the second CCA. 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_{CCA1} and T_{CCA2} ($T_{CCA1} = T_{CCA2} = 294\mu s$ on our Tmote test platform). Thus, the term E_{ii} is given as:

$$E_{ii}(p) = (1-p)^2 \cdot ((T_{CCA1} + T_{CCA2})) \tag{3.30}$$

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* (with duration T_{CCA3}).

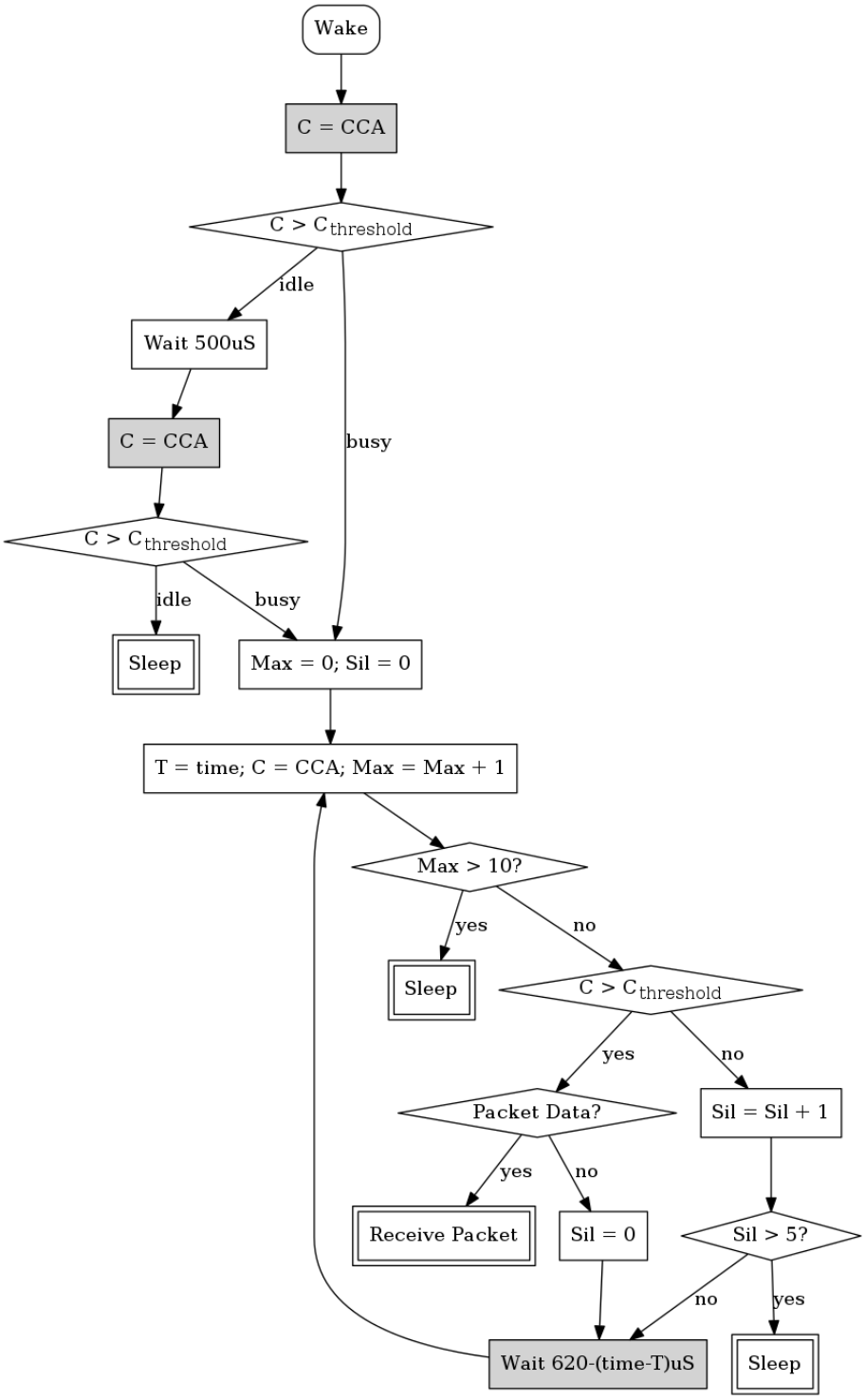


Figure 3.17: The ContikiMAC packet detection sequence. The first part of the sequence consumes $\approx 300\mu S$ of radio on time for each CCA check (represented by $C = CCA$). The second phase includes the rest of the graph and consumes $\approx 620\mu S$ for each run. Operations which cost significant time (depending on the specific radio in use) are marked in grey.

Variable	ContikiMAC Default	Description
T_{CCA}	$122\mu S$	Time to perform a CCA check
T_{CCA3}	$622\mu S$	Time to perform a CCA check when the radio is active
N_{MAX}	11	Maximum attempts to receive a packet
N_{SIL}	6	Silence period to determine the channel is 'quiet'
T_{WAIT}	$500\mu S$	Time to wait between CCA checks to keep in *phase lock*
$INITIAL\_CCA$	2	CCA checks to before entering the listening phase
$RADIO\_PREP$	$172\mu S$	Time to configure the radio

Table 3.5: The default ContikiMAC variable values

In this procedure the node evaluates if a detected channel activity is part of an incoming transmission. A maximum number of N_{max} CCA's are carried out, with $N_{max} = 11$ for a default ContikiMAC configuration. The procedure may terminate before N_{max} CCA's are carried out if N_{sil} consecutive clear CCA's are encountered, with $N_{sil} = 6$ for a default ContikiMAC configuration. Between each CCA a delay of T_{wait} 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 3.5, exactly 7 possibilities exist for the procedure to terminate before the maximum number of $N_{max} = 11$ 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:

$$\begin{aligned} E_b(p) \;=\; & p \cdot (1-p)^{N_{sil}} \cdot \Big((N_{sil} \cdot (T_{CCA3} + T_{wait}) + T_{CCA1}) \\ & + \sum_{m=1}^{(N_{max}-N_{sil}-2)} \sum_{n=1}^{m} \big[p^n \cdot (1-p)^{(m-n)} \\ & \cdot \big((N_{sil} + m) \cdot (T_{CCA3} + T_{wait}) + T_{CCA1} \big) \big] \Big) \\ & + p \cdot \Big(1 - \big((1-p)^{N_{sil}} \\ & + \sum_{m=1}^{(N_{max}-N_{sil}-2)} \sum_{n=1}^{m} \big[p^n \cdot (1-p)^{(m-n)} \big] \big) \Big) \\ & \cdot \big((N_{max} - 1) \cdot (T_{CCA3} + T_{wait}) + T_{CCA1} \big) \end{aligned} \tag{3.31}$$

E_{ib} represents the case where ContikiMAC's first CCA returns clear but the second CCA 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:

$$
\begin{aligned}
E_{ib}(p) \quad = \quad & p \cdot (1-p)^{N_{sil}+1} \cdot \Big(\big(N_{sil} \cdot (T_{CCA3} + T_{wait}) + \\
& T_{CCA1} + T_{CCA2} \big) \\
& + \sum_{m=1}^{(N_{max}-N_{sil}-2)} \sum_{n=1}^{m} \big[p^n \cdot (1-p)^{(m-n)} \\
& \cdot \big((N_{sil}+m) \cdot (T_{CCA3} + T_{wait}) \\
& + T_{CCA1} + T_{CCA2} \big) \big] \Big) \\
& + p \cdot \Big(1 - \big((1-p)^{(N_{sil}+1)} \\
& + \sum_{m=1}^{(N_{max}-N_{sil}-2)} \sum_{n=1}^{m} \big[p^n \cdot (1-p)^{(m-n)} \big] \big) \Big) \\
& \cdot \big((N_{max}-1) \cdot (T_{CCA3} + T_{wait}) \\
& + T_{CCA1} + T_{CCA2} \big)
\end{aligned} \tag{3.32}
$$

ContikiMAC carries out a channel check procedure with a frequency of f (with $f = 8 \cdot 1/s$ for the default ContikiMAC). Thus, the idle energy consumption $e(p)$ of a node in a radio interference environment with a busy channel probability of p can be given as:

$$e(p) = E(p) \cdot f \tag{3.33}$$

For ContikiMAC we obtain $P(0) = 0.471\%$ and $P(1) = 5.211\%$. 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 is 5.211%.

Monte-Carlo Method For many MAC protocols, a closed form solution is difficult to implement. As an alternative to the closed form solution, a *Monte-Carlo* method can be used. In contrast to closed form solutions, the *Monte-Carlo* solver allows us to approach other MAC protocols with ease - using a simple emulation of channel checking phase implementation. Furthermore the *Monte-Carlo* method also allows more complex environmental models such as *PDF* or even real-world interference traces to be used.

Monte-Carlo methods emulate the protocol processes against the chosen environmental model to calculate the energy consumption. This process should be repeated a sufficient number of times to obtain the desired prediction accuracy. For our ContikiMAC example, we have emulated the channel check processes described in Figure 3.17 against the simplified busy to idle ratio model derived from the *PDF* models.

We choose to implement the *Monte-Carlo* solver in PHP but any scripting language would have been capable of running these same steps. For ContikiMAC, the *Monte-Carlo* solver takes three input parameters; the probability that the channel is in use (p), channel checks per second (c) and test duration in seconds (t) and functions as illustrated in Algorithm 1. As we only consider the energy consumption of the channel check process, we will only model this component of the protocol. Within the solver, the channel check process is implemented as a

function that returns the emulated radio on time for that channel check, and is a direct port of the ContikiMAC source code. The solver calls this function $c \times t$ times.

Our implementation of the ContikiMAC channel check function executes each of the steps described in Figure 3.17 which uses a series of CCA tests to evaluate the state of the channel and how long the radio should remain active. For our purposes the CCA test is implemented as a function and utilises the provided variable p to evaluate if the CCA test should pass or fail, and each CCA test is considered independently. The function records the time the radio would be on with each emulated step, returning this time on function completion.

In the initial phase of the process, two emulated CCA tests are performed each with a probability of p that the CCA reports a busy channel. If either of these return a channel busy state, the function proceeds with the second phase of the process, otherwise the node simply returns to sleep mode. For each of these tests, a radio on time of $320us$ is added to the total time.

In the second phase of the process, a loop is used which first increments a counter representing the number of tests, MAX, before performing a simulated CCA test. If idle, this increments a count representing the number of silence periods, *silence*, or if busy, clears this counter. If the variable MAX is above 10, or the *silence* value is above 5, the function will terminate returning the total radio on time. After the test of these two variables, $620us$, the total cost of an iteration of the loop, is then added to the total on time.

With each call of the channel check process, the solver sums the total radio on-time for each check, and at the end of the simulation calculates and outputs the percentage of time the radio would have been active. The solver can be further automated with a series of different environmental models and channel check rates.

For ContikiMAC the channel check function utilises p to determine the success of each CCA test. For other MAC protocols it may be necessary to utilise a reconstructed statistically representative trace of the interference present such that the processes of the MAC protocol can be replayed against, and the Monte-Carlo method facilities this procedure.

3.4.3 Validation and Verification (D-2.3)

To validate this approach we decided to examine the Monte-Carlo method with the ContikiMAC protocol. We have evaluated how well radio energy can be predicted extensively at multiple locations around the Lancaster Campus over short and long test runs. The predictions made by the model have been shown to have high levels of accuracy. In this section we document a subset of our results from experimental deployments performed in; *The Interference Lab*, a *Meeting Room*, and a *Research Lab*. A further set of results from the second integrated demo will be presented later in the project.

The details of these experiments are described in the following sections.

Intererence Lab Results: The lab consists of a 5 meter-square room, unoccupied, and situated as far as possible from as much uncontrollable interference. The room itself is situated in one of the far corners of the departmental building, and is one of the quietest (both physically, and electromagnetically) places available.

Input: Busy probability: *busy_prob*; *checks_per_second*; *test_duration*
Result: The total energy expenditure, *total_eng*
Let: cca_clear(p) → (random(0:1) > p);

total_eng = 0; first = 0; silence = 0; chan_busy = 0; periods = 0;

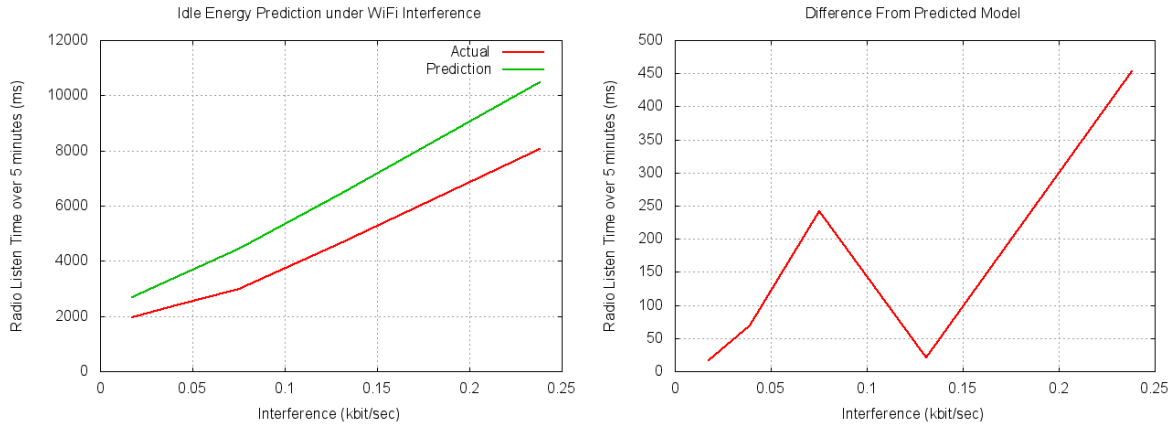
for $i = 0 \rightarrow INITIAL\_CCA$ **do**
  total_eng = total_eng + T_{CCA};
  if *cca_clear(busy_prob)* == *FALSE* **then**
    chan_busy = 1;
    break;
  end
end
if *chan_busy* == *1* **then**
  while *TRUE* **do**
    if *first != 0* **then**
      if *cca_clear(busy_prob)* == *TRUE* **then**
        silence = silence + 1;
      else
        silence = 0;
      end
      first = first + 1;
      periods = periods + 1;
      if *(silence >= N_{sil}) OR (periods >= N_{max})* **then**
        break;
      end
      total_eng = total_eng + T_{CCA3} + T_{wait};
    end
  end
end

Algorithm 1: The Monte-Carlo solver algorithm. The precise operation of the solver is highly parametrized, such that it can be adapted for use in other, non-ContikiMAC applications.

First Phase Test: This experimental configuration included a WiFi access point in one corner, with a client situated in the opposite corner to generate interference across the deployment. The transmitted data for interference was generated using the iperf tool.

A single node running ContikiMAC with standard settings - at an 8Hz channel check rate - was placed at the center of the room and monitored for activity as progressively higher interference was applied.

Eleven, 5-minute long tests were run sequentially, with each interference rate tested one after the other. The first test used tools from WP1 to determine the environmental interference in the lab, with the subsequent 10 tests using ContikiMAC. The ContikiMAC tests measured the radio energy consumption - excluding the transmit energy - such that we only record the *idle energy*.



(a) Time spent idle listening vs. interference

(b) Deviation from the model vs. interference

Figure 3.18: Interference lab results without node transmission

Second Phase Test: Once the previous test configuration had established a baseline, we ran the tests in the same configuration as before, but included data transmitted over the ContikiMAC network from another node placed at 2 meters from the first.

For this phase, all radio activity was monitored for energy consumption, namely for *idle listening*, *packet reception*, and *transmission*.

As before, 11 tests were performed, with the first test used to measure the interference present in the environment. The results of this experiment can be seen in Figure 3.19 and closely follow the trend seen in the previous phase. As the interference increases - along with transmission rate decreasing for the nodes - the accuracy of the model increases, providing accurate energy predictions for high interference rates. We see the largest deviation from our model when interference is very low, as the interference becomes increasingly unpredictable.

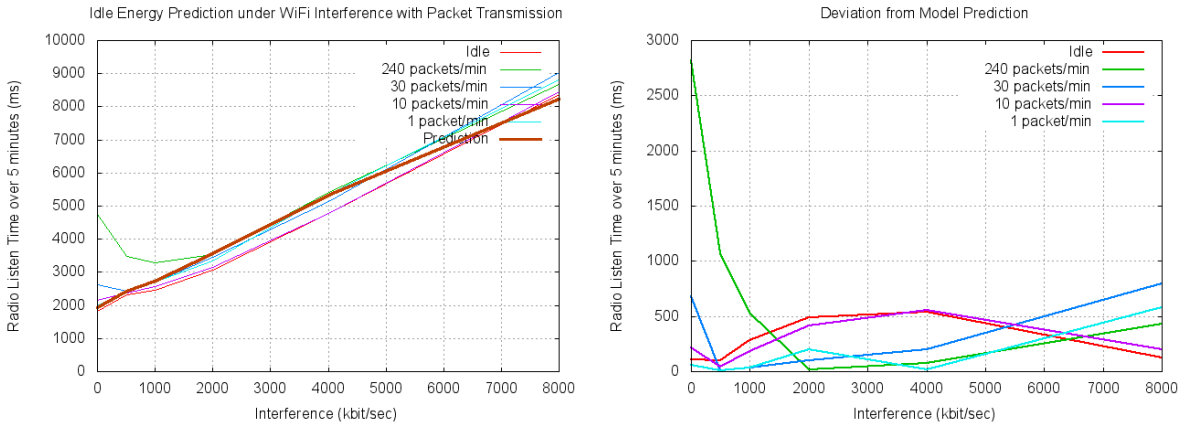
With an increase in interference, we see that idle cost becomes progressively more dominant, and increasing numbers of packets may be corrupted or backed-off due to the busy channel state.

Meeting Room Results: This and the following experiment were performed over longer periods of time. The first experimental deployment was done in a meeting room, with two nodes deployed.

One node used the tools described in WP1 to record the environmental interference at minutely intervals, with the second node running ContikiMAC - again, with default settings - recording it's idle energy consumption at 5-minute intervals.

We also calculate a rolling hourly average for idle energy consumption and interference levels which were used to predict the idle energy consumption.

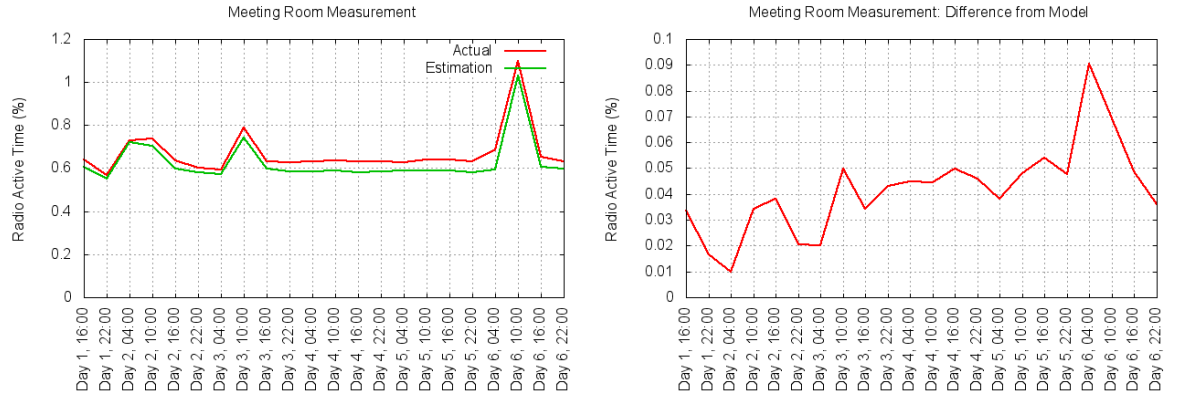
The results - as shown in Figures 3.20(a) and 3.20(b) - show a high correlation with the model's expected interference, however the measurements here are continuous, and an initial sample taken at the start of the experiment (as per the last experimental configuration) would have resulted in a higher error rate.



(a) Time spent idle listening vs. interference

(b) Deviation from the model vs. interference

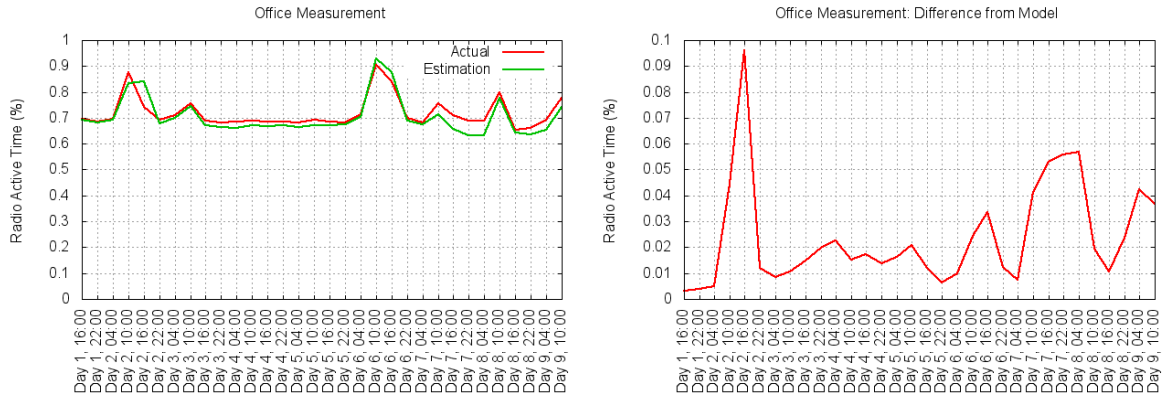
Figure 3.19: Interference lab results with node transmission



(a) Actual and Predicted radio activity

(b) Deviation from the model

Figure 3.20: Meeting Room Results: Prediction vs. Actual



(a) Actual and Predicted radio activity

(b) Deviation from the model

Figure 3.21: Office Results: Prediction vs. Actual

Research Lab Results: The research lab experiments were performed in an RF-noisy environment, and were run over a full week, but were otherwise in an identical configuration as the meeting room. The results - Figures 3.21(a) and 3.21(b) show that the model maintained it's accuracy over the duration, predicting a 0.91% error rate, and recording a 0.69% error rate in reality.

4 Models for Protocols Tackling Temperature

4.1 Temperature-Aware MAC

We have observed, studied, and presented the phenomena that are caused by the variation of temperature on low-power sensor nodes. In this chapter we introduce the models that describe these phenomena, and predict the performance of temperature aware MAC protocols. The presentation is divided in three steps. First, we recap the platform model presented in WP1, which maps temperature effects to changes in signal strength (RSSI). Second, we provide a model that maps RSSI to packet delivery ratio. Thus, with the second step we map temperature effects to actual reception rates. But the second step does not consider the clear channel assessment process (CCA). Hence, we complete our model in the third step by characterizing the impact of temperature in the CCA.

4.1.1 The impact of temperature on received signal strength

First, let's recall the impact of temperature on signal strength, as was explained in deliverable D-2.1 [46]. Many studies have shown that as temperature increases, link quality decreases and the network connectivity is altered [4, 6, 7]. Experiments carried out by Boano et al. [12] show that this signal attenuation is comprised of three distinct components; the signal attenuation on the transmitter, the signal attenuation on the receiver, and the decrease of the noise floor sensed by the receiver. The effect of each component can be seen in Figure 4.1. We first heat the transmitter and receiver separately, and then both at the same time. When the temperature increases, the RSSI decreases significantly, with the highest impact occurring when both nodes are heated at the same time. There are two effects due to the received signal attenuation. First, the packet reception rate (PRR) of a link between affected nodes is reduced. Second, the clear channel assessment at the MAC layer becomes inefficient due to packet collisions and the unsuccessful wake-up of nodes [13]. The aim of our work is to model the effect of temperature on the PRR and CCA.

A model for the impact of temperature on signal strength. In [46], we used a simple model for the signal attenuation between a transmitter-receiver pair,

$$\begin{aligned} SNR(dB) &= P_t - PL - P_n \\ &= (P_t - P_n) - (P_t - P_r), \end{aligned} \tag{4.1}$$

where PL is the path loss between the pair, P_t the transmission power, P_r the received power, and P_n the noise floor at the receiver.

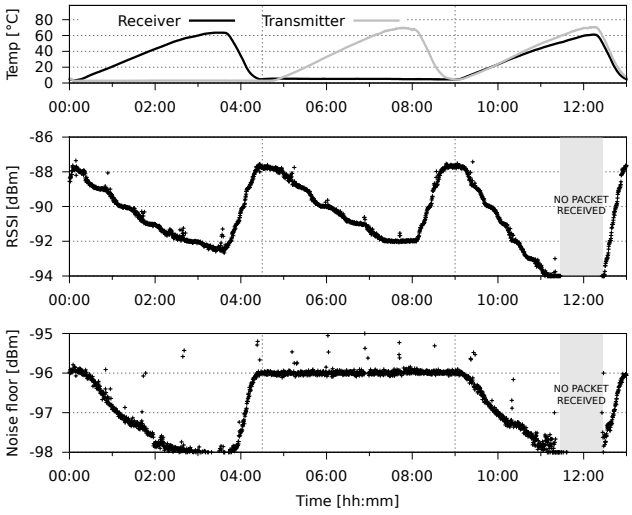


Figure 4.1: Impact of temperature on the received signal strength of a single link.

For a given sensor platform we defined α, β, and γ as constants with units dB/K, and T_t, T_r as the temperature in Kelvin of the transmitter and receiver respectively. The effect of temperature on SNR was defined as:

$$\begin{aligned} SNR &= (P_t - \alpha\Delta T_t) - (PL + \beta\Delta T_r) \\ &\quad -(P_n - \gamma\Delta T_r + 10\log_{10}(1 + \tfrac{\Delta T_r}{T_r})) \\ &= P_t - PL - P_n - \alpha\Delta T_t \\ &\quad -(\beta - \gamma)\Delta T_r - 10\log_{10}(1 + \tfrac{\Delta T_r}{T_r}) \end{aligned} \tag{4.2}$$

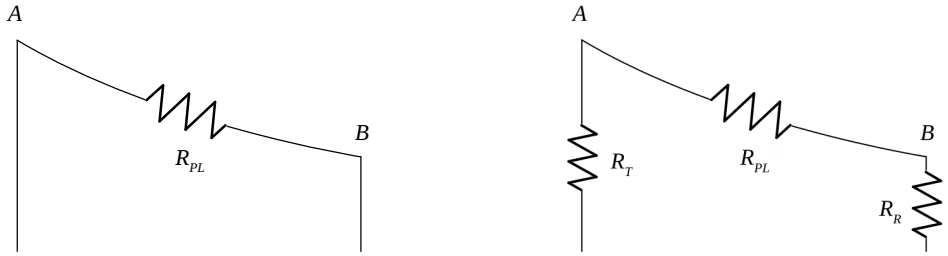
The parameters α, β, and γ are computed empirically using the slopes of the linear trends that were observed for each platform when running the experiments. In Figure 4.2, we illustrate the effect of each parameter by using a simple electrical circuit analogy that captures the characteristics of the previous model. The attenuation of the signal over the channel is modeled with resistor R_{PL}. The attenuation of the signal due to the transmitter and the receiver is modeled with resistors R_T and R_R respectively.

4.1.2 Modelling the Packet Reception Rate

The model that we have explained so far only captures the effect of temperature on RSSI. We need an upper layer abstraction that will better describe the network. This is the packet reception rate (PRR). In order to acquire a model for the PRR of a link, first we will introduce the basic channel and radio models, and then we will enhance them with the effects of temperature.

The basic channel model. We are going to use the log-normal shadowing path loss model for the signal propagation, which is given by:

$$PL(d) = PL(d_0) + 10nlog_{10}(\frac{d}{d_0}) + X_\sigma \tag{4.3}$$



(a) Without temperature effects (b) With temperature effects

Figure 4.2: A simple electrical circuit equivalent model for a single link.

where d is the transmitter-receiver distance, d_0 a reference distance, n the path loss exponent (rate at which the signal decays), and X_σ a zero-mean Gaussian RV with standard deviation σ. These parameters are obtained through curve fitting of empirical data. Figure 4.3 is an example of an analytical propagation model for n=4, σ=3, $PL(d_0)$=55 dB and an output power of 0 dB.

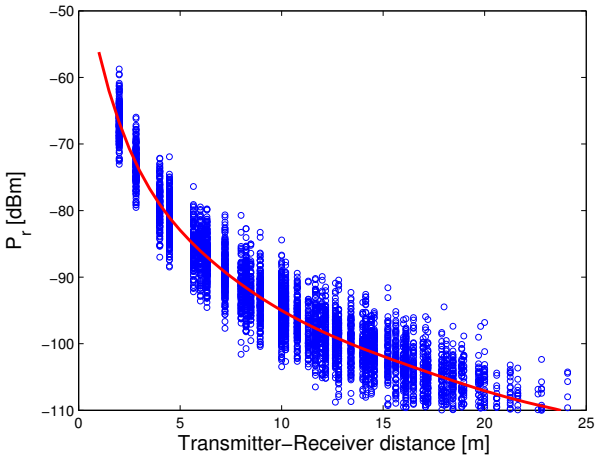


Figure 4.3: Analytical channel model.

This model provides the relation between signal attenuation and distance. We now need to connect the RSSI with the PRR of the link. We accomplish this by using a model for the radio platform.

The basic radio model. The model for the radio is dependent on the encoding used. In general, the probability of receiving a packet is:

$$p = (1 - p_e)^N \tag{4.4}$$

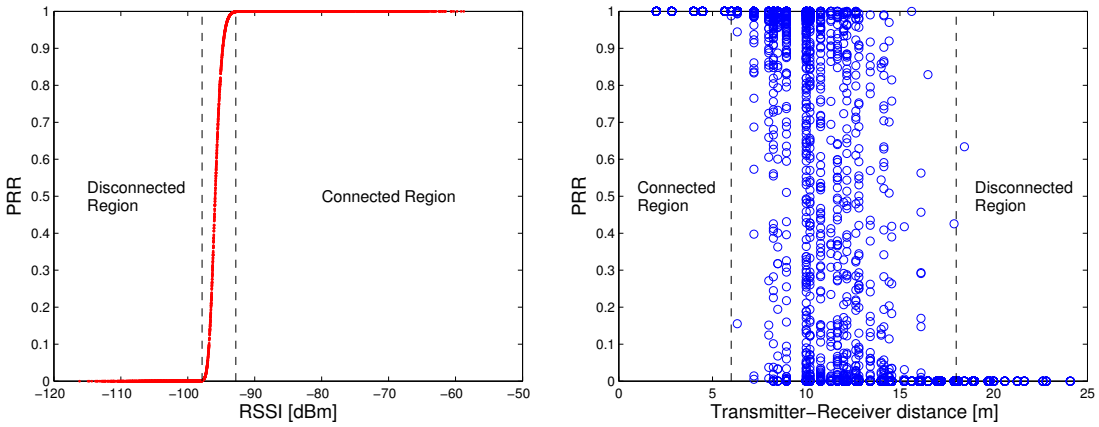
where p_e is the probability of bit error and N is the length of the packet. p_e depends on the modulation scheme used and it is a function of the received signal strength:

$$p_e = f(SNR) \tag{4.5}$$

In the end the generalized PRR model for the radio is:

$$p = (1 - f(SNR))^N \tag{4.6}$$

The sigmoid curve that is derived of this model is shown in Figure 4.4(a). Notice that in the area between -93 and -98 dBm, the PRR is between zero and one. This area is called the transitional region of the radio receiver. The other two are the connected and the disconnected regions, having a PRR of one and zero respectively. We will show that temperature modifies the limits of these regions, due to the attenuation of RSSI. Figure 4.4(b) demonstrates the physical interaction between the channel and radio model, as seen in Equation 4.6. The sigmoid curve that is derived of this model is shown in Figure 4.4(a). Notice that in the area between -93 and -98 dBm, the PRR is between zero and one. This area is called the transitional region of the radio receiver. The other two are the connected and the disconnected regions, having a PRR of one and zero respectively. We will show that temperature modifies the limits of these regions, due to the attenuation of RSSI. Figure 4.4(b) demonstrates the physical interaction between the channel and radio model, as seen in Equation 4.6.



(a) Radio model - RSSI

(b) Radio model - Distance

Figure 4.4: Analytical model for the radio (a). PRR vs Distance (b).

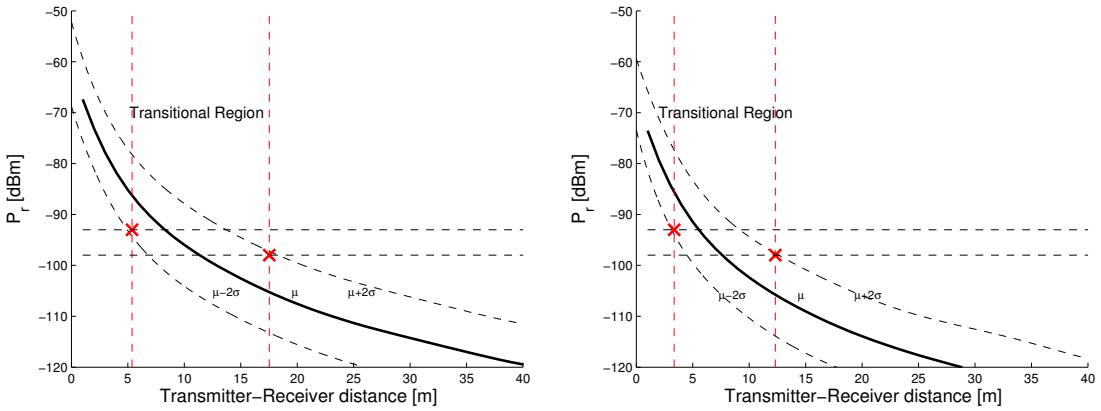
The last property we need to evaluate is the noise floor, P_n. The noise floor depends on the radio and on environmental properties such as temperature and interference [37]. We will not use a model since we have measured it at an ambient temperature of 25°C at -96 dBm [12].

The enhanced temperature-aware model. Using Equation 4.2, we can derive the complete model for the packet reception rate of a single link when the temperature variation on the

transmitter and the receiver are respectively ΔT_t and ΔT_r :

$$p = (1 - f(P_t - PL - P_n - \alpha\Delta T_t - (\beta - \gamma)\Delta T_r))^N \tag{4.7}$$

We simulate this model for a $\Delta T_r = \Delta T_t$ of 0 and 40°C. Figure 4.5 demonstrates the effects of temperature on the signal decay over the channel. The first effect that we notice is the shrinkage of the regions. At the initial temperature, the connected region has a radius of 5 meters, and the transitional a radius of 12.5 meters. After increasing the temperature at both the transmitter and receiver, these regions shrink to 3.5 and 9 meters respectively. This is a reduction of 30% and 28%.



(a) Without temperature effects, $\Delta T = 0°$ (b) With temperature effects, $\Delta T = 40°$

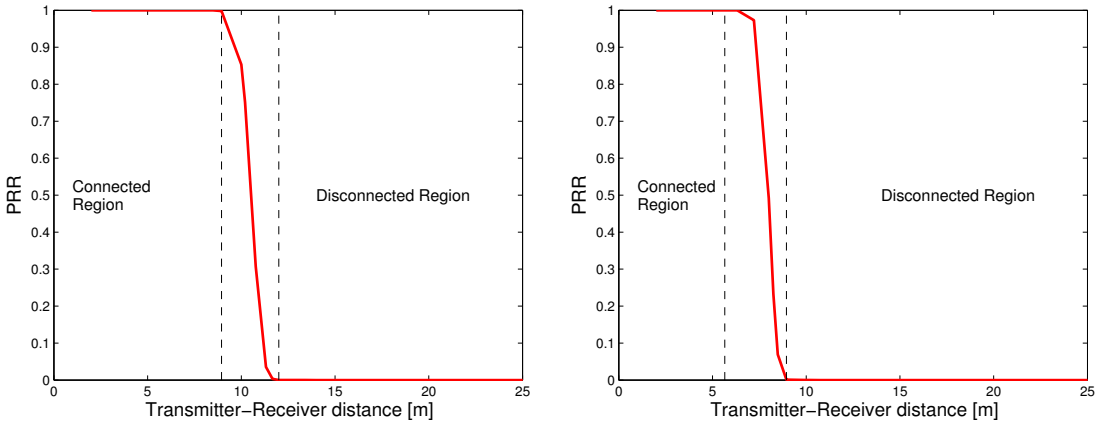
Figure 4.5: Effect of temperature on the channel model.

A better understanding of this dramatic effect can be seen in Figure 4.6. Imagine a neighboring node being at a distance of 9 meters, inside the connected area, with a PRR of 1. After increasing the temperature, the same node will fall into the disconnected region, having a PRR of 0 just because of the effect of temperature.

When looking at this effect from the network's perspective, the problem is magnified. The probability of a packet being successfully received after h hops when the PRR for each hop is p, is p^h. Thus nodes in the transitional area, that have a PRR < 1 may get disconnected from the rest of the network as the total PRR will decrease significantly.

4.1.3 Modelling the CCA Threshold

With the previous two steps we have a solid model for the wireless link, but there is no model for the protocol yet. Before receiving a packet, a node coming out of sleep checks if the signal strength of the channel is above a fixed threshold, denoted T_{CCA}. If the channel's signal strength, P_r, is greater than T_{CCA} the node infers that an ongoing transmission is present and remains awake to receive the packet. But under high temperatures, the node will measure a lower signal strength $P_r^{'} = P_r - \beta\Delta T_r$. If $P_r^{'}$ is lower than the fixed CCA threshold, the node will believe that there is no ongoing packet transmission, while in fact there is one. This effect



(a) Without temperature effects, $\Delta T = 0°$ (b) With temperature effects, $\Delta T = 40°$

Figure 4.6: Effect of temperature on the radio model.

reduces the delivery rate. Next we derive a model to adjust the CCA dynamically to overcome this problem. In our analysis and evaluations, the initial CCA is set to -77 dBm, which is the default value used by ContikiMAC.

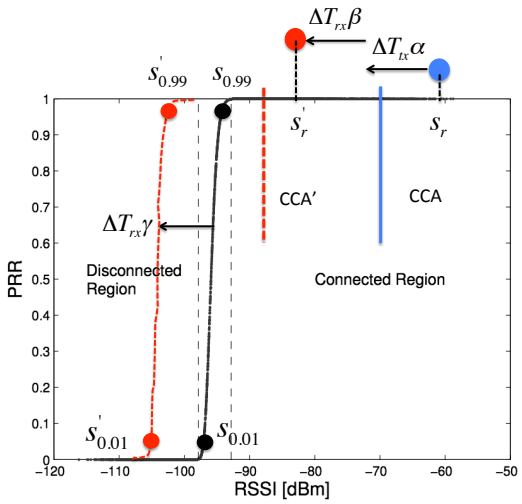


Figure 4.7: Analytical model for dynamic CCA adjusting for TempMAC

Scenario. We consider n nodes sending information to a central node (sink) that is 1-hop away, i.e. the nodes and the sink form a star topology. The transmitters wake up periodically and asynchronously –every t_{pkt}– to send packets to the sink. The receiver (sink) wakes up either in an asynchronous manner –every t_{wakeup}–; or in a synchronous manner, i.e. the receiver learns

the wake up times of the transmitters and wakes up some time before. Upon waking up, the receiver measures the signal strength in the channel. If the measured signal is above the Clear Channel Assessment threshold (CCA), then the receiver deems the channel to be active (i.e. a packet is on the air) and starts the decoding process. If the measured signal is below the CCA threshold, the receiver deems the channel as idle and goes back to sleep.

Challenge. In Work Package 1, we showed that temperature affects the quality of links. The higher the temperature, the lower the signal strength. This means that the link quality of some transmitters may fall below the CCA threshold. Thus, the sink may observe less neighbors that the original n, reducing the overall delivery rate. The goal of our model is to adjust the CCA threshold of the sink in such a way that it preserves the original number of neighbors n.

The intuitive idea behind our model is depicted in Figure 4.7. Initially, a good link (blue dot at -60 dBm) may have an rssi that is high enough to be above the CCA value (vertical blue line at -70 dBm) and to get a packet reception of 100% (connected region). But due to temperature effects at the transmitter, receiver, or both; the rssi may fall below the CCA value, and hence, the link will be lost. To overcome this problem, the CCA needs to be adjusted according to temperature effects, as shown by the two arrows: $\Delta T_{tx}\alpha$ (due to the transmitter being heat up) and $\Delta T_{rx}\beta$ (due to the receiver being heat up). Another important effect that our model should capture is that the transition phase of the radio response (black sigmoid curve) will shift left (towards the red sigmoid curve). In effect this shift implies that the radio can receive signals with lower rssi values.

Counteracting the temperature effects of the transmitters. Let us denote ΔT_i as the increase in temperature at transmitter i. According to the platform models derived in Work Package 1, an increase in temperature maps to a decrease in signal strength $\Delta T_i\alpha$, where α is a platform dependent parameter. Hence, to maintain the connectivity with all n transmitters, the new threshold CCA' needs to be:

$$CCA' = \min_{i=1,\ldots,n} \{CCA - \Delta T_i\alpha\} \tag{4.8}$$

Counteracting the temperature effects at the receiver. Denoting ΔT_{rx} as the increase in temperature at the sink, and β and γ as the platform dependent parameters, the threshold CCA' needs to be updated in the following manner:

$$CCA' = \min_{i=1,\ldots,n} \{CCA - \Delta T_i\alpha - \Delta T_{rx}\beta\} \tag{4.9}$$

The role of radio sensitivity. From a practical perspective, there is a limit on the minimum value that the CCA' threshold can take. That value depends on the radio response and can be obtained from equation 4.7:

$$p = (1 - f(s_r - s_n))^b \tag{4.10}$$

Where p is the probability of receiving a packet, s_r is the measured signal strength in dBm, s_n is the noise floor in dBm, $(s_r - s_n)$ is the signal to noise ratio in dB, and b is the number of bits in the packet. This equation has a sigmoid trend similar to the one depicted in Figure 4.4(a).

Denoting $s_{0.99}$ as the signal strength that leads to a delivery rate of 0.99, and $s_{0.01}$ as the corresponding signal strength for a delivery rate of 0.01, we can define three reception regions. A connected region, where the received signal strength is above $s_{0.99}$. A disconnected region,

the signal strength is below $s_{0.01}$. And a transitional region where the delivery rate drops monotonically between 1 and 0. If the receiver (sink) is affected by temperature effects, the sigmoid curve shifts to the left by $\Delta_s\gamma$.

Now let us redefine our notation to capture temperature effects on all parameters: $s'_r = s_r - \Delta T_i\alpha - \Delta T_{rx}\beta$, $s'_{0.99} = s_{0.99} - \Delta T_{rx}\gamma$ (similar for $s'_{0.01}$). Now with this notation we proceed to define the delivery rate $f_i()$ provided by the dynamic CCA' approach for each link i as:

$$f_i(CCA', s'_r) = \begin{cases} 1 & \text{if } \max\{CCA', s'_{0.99}\} < s'_r \\ p & \text{if } s'_{0.01} \leq CCA' < s'_r \leq s'_{0.99} \\ 0 & \text{if } CCA' < s'_r < s'_{0.01} \end{cases}$$

Determining the initial CCA. The last important point that our model should cover is the ability of selecting an appropriate initial CCA value. The requirement for the initial CCA value is simple: a CCA value that guarantees that a group of links are reliable with and without temperature effects. Let us denote ρ as the reliability requirement, which can take values between 0 and 1. For example, a $\rho = 0.9$ would indicate that we want the system to guarantee that a link will be above 90% delivery rate, with and without temperature effects.

Let us denote $pmf_i()$ as the probability mass function of rssi values for link i, and $rssi^i_\rho$ as the highest rssi value that leads to a cumulative probability mass of $(1-\rho)$ for link i. This basically means that to achieve reliability ρ on link i, $rssi^i_\rho$ should be on the connected region of the radio response and it should also be above the CCA threshold. Formally, these two conditions can represented by the following inequality: $rssi^i_\rho > \max\{CCA, s_{0.99}\}$.

Now let us assume that we have a set of receivers at low temperatures, and we would like to connect only to the links whose reliability will remain above ρ at high temperatures. What initial CCA threshold should we use? We could select a very high and conservative CCA threshold, but this choice may blacklist links that would otherwise be strong enough to be received with and without temperature effects. If on other hand, we select a very low CCA threshold, we risk selecting links whose reliability will go below ρ once high temperatures kick in. The initial CCA value should hence be set to:

$$CCA_{init} = \min_{i \in n}\{rssi^i_\rho > s_{0.99} + \Delta T_i\alpha + \Delta T_{rx}(\beta - \gamma)\} \tag{4.11}$$

In other words, the minimum $rssi^i_\rho$ that will remain in the connected region even when high temperatures are present.

4.1.4 Validation and Verification (D-2.3)

This section uses empirical data to validate the theoretical models presented in the previous sections. We will first validate the results for Section 4.1.2, and then Section 4.1.3.

First, we focus on validating the effects of temperature on the transitional region. This case represents the baseline, since no MAC protocol is used on top. Figure 4.8 shows the radio sensitivity without temperature effects, which matches the theoretical trend in Figure 4.6. Recall that the radio response determines the connected, disconnected and transitional regions. The radio sensitivity curve was obtained by having seven nodes transmitting 100-packet bursts with different power levels to a single receiver. For each one of these bursts, we calculated the

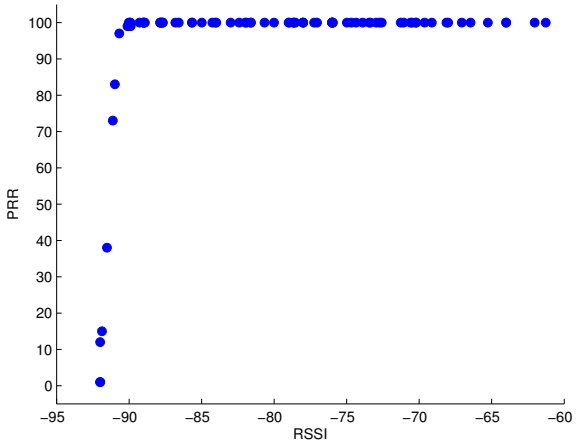


Figure 4.8: Empirical radio response without temperature effects.

delivery rate (prr), i.e. how many of the 100 packets were received, and the average rssi value of the received packets. Each point in Figure 4.8 represents a single burst tuple $< rssi, prr >$

Subsequently, a series of experiments were conducted to capture the effects of temperature on the radio's receiver response without any MAC layer. To this end, the eight nodes with heating lamps in TempLab are used. Among these eight nodes, one node (207) is selected as the receiver, and the other seven nodes send data packets periodically to that receiver. Each of the seven nodes can reach the receiver directly in a single hop, thus no routing layer is employed. A new data packet is sent every five seconds by each one of the seven nodes. Each experiment is 40 minutes long, but considering that most of this time is used to heat up and cool down the nodes, the results for high temperature use only the 10 minutes in the middle where nodes reach the maximum temperature of 60°.

Fig. 4.9 shows the results of four different experiment: i) no node is heated, ii) only the transmitters are heated, iii) only the receiver is heated, and iv) all nodes, transmitters and receiver, are heated. Our validation should show two outcomes. First, that temperature reduces the reception rate (Equation 4.7). Second, that the radio sensitivity curve shifts to the left when the receiver is heated (which is the change due to γ in Figure 4.7, from the black curve to the red curve). We show these two results in Figure 4.9. The x-axis represents the rssi, and the y-axis the number of packets received for a given rssi. The different colors of the histograms (bars) are used to identify each individual transmitter. First, we observe that increasing the temperature –either at the transmitter, receiver or both– (i) shifts all histograms to the left, because the temperature decreases the rssi (Equation 4.2), and (ii) reduces the delivery rate (Equation 4.7), captured by the lower height in all bars. Second, heating the receiver makes radio more tolerant to receiving low-rssi packets, since the radio response moves to the left. This phenomena can be observed in the two bottom graphs of Figure 4.9. In these two graphs, the receiver is heated and we observe packet reception below -92 dBm, which is part of the disconnected region ($prr = 0$) when the receiver is not heated.

After showing the effects of temperature on the packet reception rate without any MAC

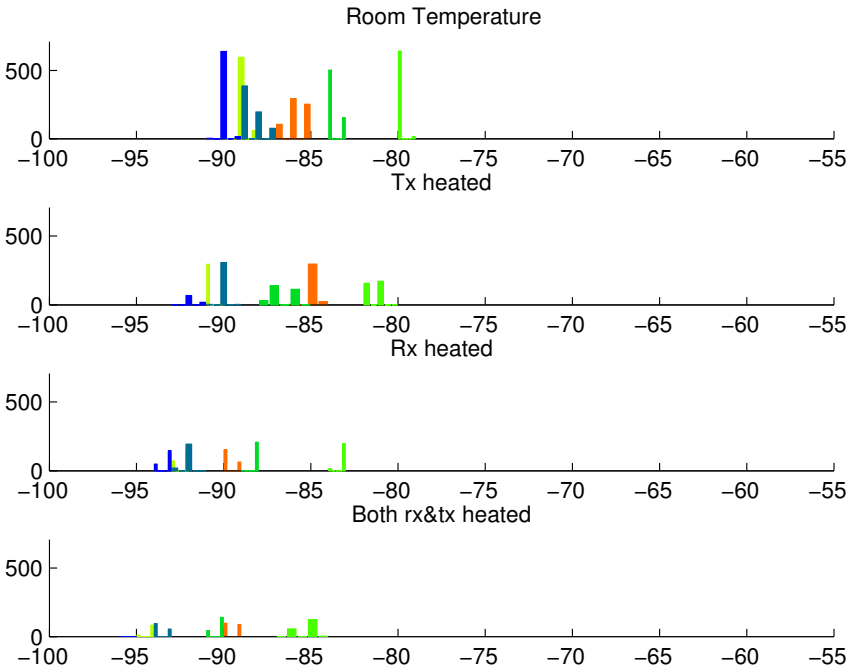


Figure 4.9: Effects of temperature on reception rates without any MAC protocol. The x-axis represents the RSSI and y-axis the number of packets received for each RSSI value. The different colors of the histograms (bars) are used to identify the packets of each receiver.

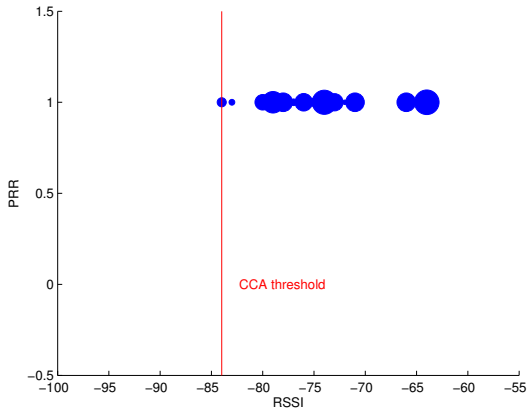


Figure 4.10: Empirical evaluation of the CCA threshold. The size of the bubbles denotes the number of packets received for each RSSI value.

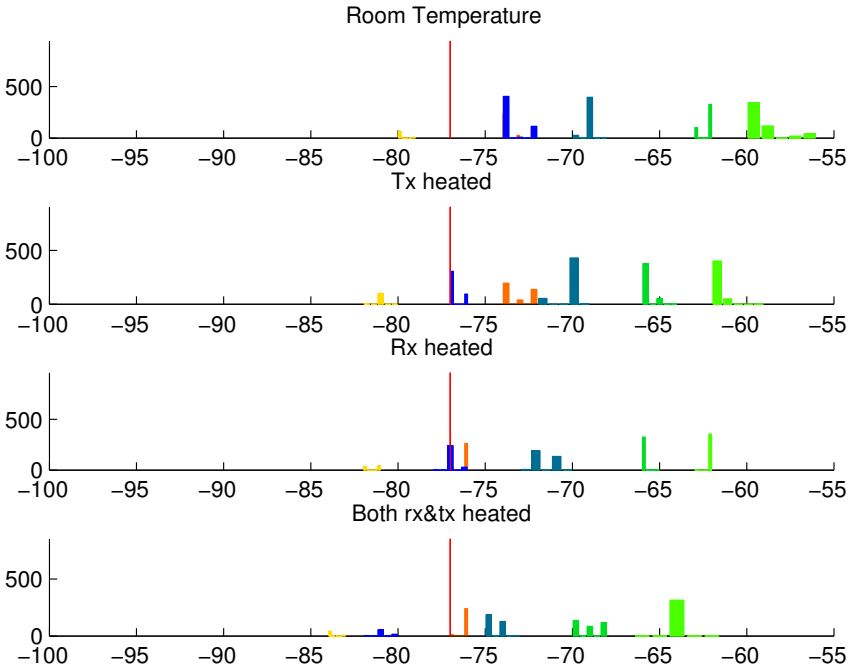


Figure 4.11: Effects of temperature on reception rates with ContikiMAC. The x-axis represents the RSSI and y-axis the number of packets received for each RSSI value. The different colors of the histograms (bars) are used to identify the packets of each receiver.

protocol, we repeat the same set of experiments using ContikiMAC. This is done to capture the detrimental effects of temperature on CCA-based protocols. ContikiMAC uses CCA to allow nodes to remain asleep and conserve power by ignoring any radio signal received below the CCA threshold. Unlike the transition phase of the radio response, which is a sigmoid curve, the CCA threshold has a much sharper decay. Figure 4.10 shows the effect of the CCA threshold on data packets. Note the abrupt change in the reception rate, no packets are received below the CCA value.

Figure 4.11 shows the same four experiment carried out previously (in Figure 4.9) but with ContikiMAC. The red vertical lines indicate the CCA threshold. To stress the capacity of the network, nodes use a much higher data rate: 2 packets per second. The figure shows that the reception rate reduces significantly as the transmitters and receiver are heated up. There are two trends that are important to highlight. First, contrary to the radio response –which tolerates lower rssi signals when the receiver is heated–, the CCA threshold is a hard value, and hence, has a stronger effect in reducing the packet reception rate. Second, we observe that some packets are received below the CCA threshold, which in principle should not be possible. This phenomenon occurs because, after receiving a packet, ContikiMAC remains awake for some more time in case other packets arrive. For these latter packets no CCA check is performed, and hence, they could be received at lower RSSI values. Since our experiments use a very high data rate, there is a high chance of "piggybacking" low-rssi packets after getting above-CCA-rssi packets.

Finally, it is important to mention that the benefits of (i) dynamically adjusting the CCA to overcome the loss in data delivery and (ii) setting an optimal initial CCA value, will be presented in D4.4; when TempMAC is used in one of the integrated experiments.

4.2 TempLife

When a sensor network is deployed, we fundamentaly care about three main outcomes: to obtain as much data as possible (high livery rate), to obtain data as fast as possible (low latency), and to obtain data for as long as possible (long lifetime). Thus far, our protocols have focused on analyzing the delivery rate and latency. With TempLife, we aim at modelling the impact of temperature on the network lifetime.

The network lifetime is a function of the lifetime of individual nodes, and nodes belonging to the same network can have widely different lifetimes. In general, the lifetime of a node depends on three parameters: the energy level of the batteries; the individual computation, sensing and communication load; and the efficiency of the communication primitives. The energy level of batteries however changes with temperature, and the relationship between these two metrics is complex and non-homogeneous. High temperatures may provide a short boost in the short term (because the internal resistance is reduced), but they reduce the lifetime of the battery in the long term. Low temperatures increase the internal resistance, and hence, reduce the flow of current. Batteries usually have an ideal (room) temperature at which performance is maximized. Adding to all this non-linear behavior, the performance of batteries vary depending on their manufacturing material.

Considering that we were not able to derive battery models in WP1 (due to lack of time), our lifetime models provide only a general framework. To bypass the complexity of battery models,

we emulate different initial values for battery capacities in our evaluation. This approach gives initial insights on the effects that changes in the battery's capacity can have on the lifetime of the network. With this general framework, battery models could be later introduced.

4.2.1 Network Lifetime

Considering that most of the applications in sensor networks focus on data gathering, we focus our analysis on two data collection paradigms: (pseudo) shortest-path trees, such as the CTP protocol, and opportunistic forwarding, such as ORW. In data collection applications, nodes are deployed in the scenario of interest, and they need to transmit information in a multi-hop manner to a particular node in the network called *sink*. Figure 4.12 depicts the simple idea behind data collection.

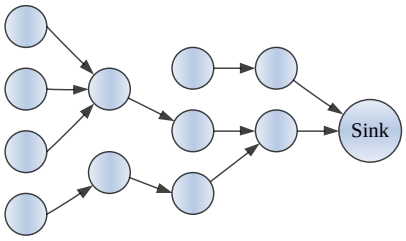


Figure 4.12: An example for data collection applications

Once deployed, we would like the network to deliver data for as long as possible. That is, we would like to maximize the network's lifetime. In most WSN applications, maximizing the network's lifetime is central because nodes usually do not have access to a continuous power supply, and they have to run on batteries to fullfill the given task. Limited by the size of nodes, batteries cannot be too big, and hence, energy becomes a scarce resource where every joule should be spent discreetly [3]. Temperature changes would aggravate this phenomena due to the randomness introduced in the lifetime of each individual battery.

Network Lifetime and its Key Components

The network lifetime is a function of the lifetime of individual nodes, and nodes belonging to the same network can have widely different lifetimes. Given the dependence on individual node lifetimes, the network lifetime can be defined in many ways. Common definitions include the time passed until the first node in the network dies, or the time when the last node dies.

The lifetime of a node depends on two components: the initial energy level of the batteries and the individual computation, sensing and communication loads. The computation load is usually negligible and most sensors are very energy efficient. Hence, wireless communication is usually the most energy-hungry component.

Considering that radio communication is the main energy drain, the lifetime of the network depends on three main aspects: (i) the cost of transmitting a packet from one node to its neighbor (communication primitive), (ii) the total amount of packets transmitted by all nodes

in the network (transmission load), and (iii) the even distribution of packet transmissions among nodes (load balance). Many works, mainly theoretical, have proposed a wide range of techniques to optimize the load balance of the network. But, these studies do not consider (i) the different types of communication primitives present in actual protocols, (ii) that specific implementations can change the total number of transmissions in the network, and (iii) that environmental effects, such as temperature, can introduce further uncertainty on the lifetime.

Challenges

The aim of this report is to study the lifetime of current protocols considering –in a comprehensive manner–: communication primitives, transmission loads and load balancing. This initial framework will allow us to gain insights into the effects of phenomena affecting the lifetime of batteries, such as temperature. Analyzing the lifetime of state-of-the-art protocols is not trivial because they follow conflicting design guidelines. The de-facto standard collection protocol used for the past 10 years, *Collection Tree Protocol (CTP)* [22], aims at minimizing the number of transmissions rather than to achieve load balancing or to use efficient communication primitives. Under these circumstances, a few nodes end up being heavily loaded because they have to perform many (and expensive) transmissions. Overall, these design guidelines lead to a fast depletion of energy in those heavily loaded nodes, which is especially detrimental for the lifetime of large-scale networks [27]. On the other hand, recent protocols, like *Opportunistic Routing for Wireless Sensor Networks (ORW)* [26], obtain a more balanced routing by using efficient communication primitives, but at the cost of increasing the number of transmissions (compared to CTP). Given that ORW is a more recent protocol, it has not been used as widely as CTP and it has not been thoroughly investigated either.

Problem Statement

Considering the current situation, there is an open question that has not been clearly answered yet:

Which data collection method is better in terms of network lifetime? One prioritizing number of transmissions over communication efficiency and load balance (CTP) or vice versa (ORW)?

The problem is further complicated because network lifetime is not a clearly defined terminology. Depending on the specific application, different definitions can be applied. Thus it would be interesting to investigate how these two protocols behave across these various definitions. In this report, we do exactly that, we consider the entire spectrum of network lifetime: from the first node that dies, to the last one, including all the fraction of nodes within these two extremes.

Contributions

This work contains mainly two parts. First, an analytical framework is proposed to understand the cost of communication primitives and load balance in CTP and ORW. Second, experiments are conducted in testbeds to reveal insights about the death process of a network. The overall contributions are:

- Improving the existing energy model for CTP, which enhances the accuracy of the model by up to 95% (Section 4.2.2).
- Creation of a new model for ORW (Section 4.2.2), and a comparison between the CTP and ORW models (Section 4.2.2).
- Showing the effect of energy budgets, density, and scale on the lifetime of ORW and CTP (Section 4.2.3). From our results we infer that under high densities, CTP will be more vulnerable than ORW to phenomena affecting battery levels (such as temperature), while at low densities the opposite is expected to be true.

4.2.2 Lifetime Model

Considering that most public testbeds do not allow direct measurements of energy consumption, researchers have proposed indirect methods to estimate the energy consumption of nodes. By and large, the most widely used method to estimate energy consumption is to keep track of the duty cycle of the radio. In most scenarios radio communication accounts for most of the energy drain, hence, by keeping track of the percentage of a time a radio is kept 'on', i.e. by keeping track of its duty cycle, we can estimate the node's energy consumption.

In this chapter, we introduce the existing duty cycle model for CTP and show how we improve the model. Then we exploit the new CTP model to create a model for ORW, after which a comparison is made between the two models. Throughout our work, we consider that both protocols employ the low power listening method defined by BOX-MAC-2.

Model for CTP

Initially, most theoretical studies assumed a very simple model for the radio. The radio was assumed to be always 'on', consuming a constant energy in this stage, and an extra consumption of energy was added during packet transmissions. The extra energy used during transmissions depended on the distance between the transmitter and receiver, the longer the distance, the higher the energy used. Once BOX-MAC-2 was designed, this model became obsolete for two reasons. First, in BOX-MAC the radio is kept mainly 'off', and second, there is not much difference between the energy used for transmission and reception, because the distances covered by sensor nodes are very short. The authors of BOX-MAC [28] developed a simple duty cycle model for low power listening, and later, the authors of the Broadcast Free Collection Protocol [36] proposed a refined version after considering CTP's features. These models will be explained in more detail later.

Our models and the SoA models assume that all nodes are duty cycled with the same wakeup interval except the sink, which is always on. Having the sink always 'on' is a fair and common assumption, since most sensor networks deployments actually do this in practice.

We mentioned in the introduction that nodes in the same network can have widely different lifetimes. In order to capture this difference, nodes are divided into three groups, as shown in Figure 4.13: sink neighbors (nodes within one hop from sink), leaves (nodes with forwarding load smaller than 1.5[1]) and relays (the rest of the nodes).

[1]Ideally a leaf node should have a forwarding load of one, namely only forwarding its own packet. But in

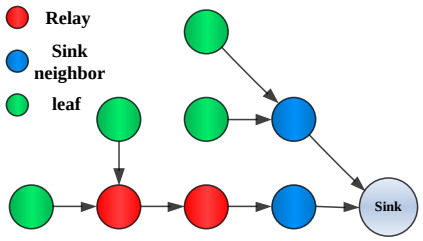


Figure 4.13: Node division

Given that our work extends the model described in [36], we will first describe that model.

Existing Model The model introduced in [36] considers the five basic communication primitives contributing to CTP's duty cycle: (1) CCA (clear channel assessment) events, (2) Beacon transmissions (3) Beacon receptions, (4) Unicast transmissions, and (5) Unicast receptions. We will now introduce the five aforementioned primitives in detail. Table 4.1 summarizes our notation, and Figure 4.14 and Figure 4.15 will help us understand the beacon and unicast primitives.

(1) *CCA events*
Nodes keep their radios off most of the time but they wakeup at every interval t_w to perform a channel assessment. The goal of this channel assessment is to see if there are any packets being transmitted in the channel. Denoting t_c as the duration of the channel assessment, the contribution of CCA events to the duty cycle can be denoted as

$$\Delta_{rc} = \frac{t_c}{t_w} \tag{4.12}$$

(2) *Beacon transmissions*
At every interval t_{IBI}, nodes send out beacons to broadcast their routing status. Given that nodes wake up every t_w to check the channel, nodes have to transmit their beacons for a duration of t_w to ensure that every neighbor receives the beacon, as shown is Figure 4.14. Hence, the contribution of beacon transmissions to the duty cycle is

$$\Delta_{bs} = \frac{t_w}{t_{IBI}} \tag{4.13}$$

(3) *Beacon receptions*
When the network enters a steady routing state, the number of beacons received within a beacon interval t_{IBI} should be equal to the number of neighbors node i has. Denoting t_{rx}

practice a leaf node will occasionally forward other node's packet. We use this threshold to separate real leaves and relays

Symbol	Value	Description
t_w	0.25s $\sim$ 1s	Time of wake up interval
t_c	12.5 ms	Time required to perform CCA
t_{rx}	25/35 ms	Time for receiving a packet[2]
t_{tx}	26/36 ms	Time required to send a packet by the sink's neighbors
t_{IBI}	8 min	Time of beacon interval. Under stable status, one node will send a beacon every 8 minutes
t_{IPI}	1 min	Time of packet interval. Throughout our experiment it's set to 1 minute

Table 4.1: Symbols and their default values

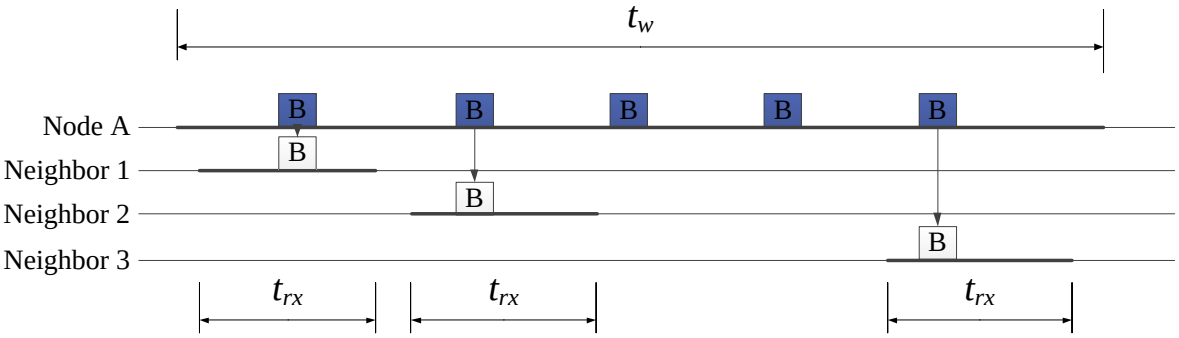


Figure 4.14: A typical broadcast event in CTP

as the time required to receive a packet (shown in Figure 4.14), and N_i as the number of neighbors, the contribution of these reception events to the duty cycle is

$$\Delta_{br} = \frac{t_{rx}}{t_{IBI}} N_i \tag{4.14}$$

(4) *Unicast transmissions*
In CTP, every node has a parent that it should forward information to until the information reaches the sink. A node will either transmit its own packet (generated every t_{IPI}), or forward the packets that are generated by other nodes. Figure 4.15 shows a typical unicast process with BOX-MAC. A child node wants to send a packet to its parent node. Any other nodes hearing the ongoing transmission will extend the 'on' time of the radio, but will ignore the packet at the end (because the packet is not intended for that specific node). When the parent node wakes up and detects the packet, it will send an acknowledgement. This process is repeated at each hop until the packet reaches the sink.

This primitive's contribution to the duty cycle is related to two parameters: the amount of time the radio needs to be on until the parent wakes up (rendezvous time), and the link quality (if the link is of poor quality the radio will need to be kept on for a longer time

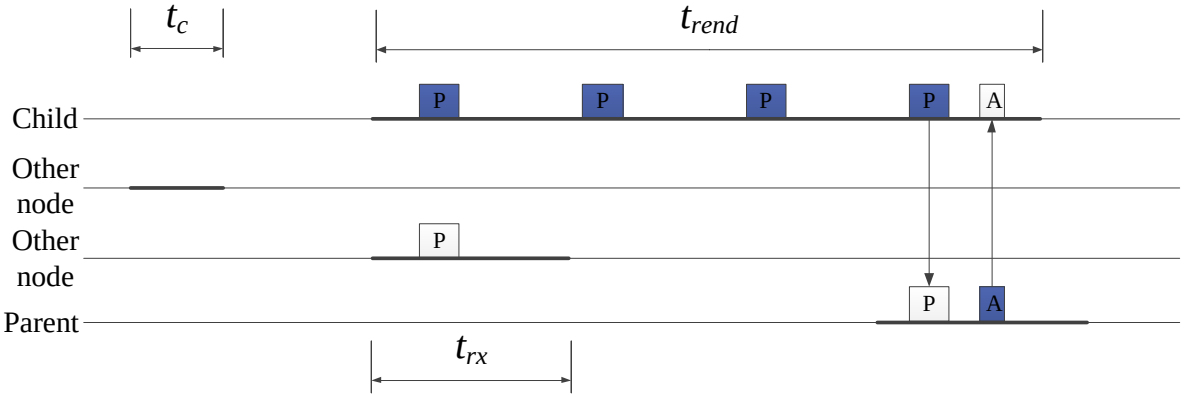


Figure 4.15: A typical unicast transmission in CTP. The bold lines represent the radio 'on' time of the nodes.

to accommodate retransmissions). To simplify the problem and to capture the essence of low power listening methods, here we assume that all links are perfect, i.e. 100% reliable. This is a fair assumption, because CTP tends to select very good links. Considering that the wakeup time of a potential parent is uniformly distributed within t_w, the expected rendezvous time is $t_w/2$. Denotting F_i as the forwarding load of node i, the contribution to unicast transmissions to the duty cycle is:

$$\Delta_{us}^{i} = \frac{t_w/2}{t_{IPI}} F_i \tag{4.15}$$

The above equation is only valid for relay nodes and leaf nodes. Considering that the sink's neighbors do not need to wait for the sink to wake up (because the sink is always on), the unicast sending time is just t_{tx}. Hence, the contribution to unicast transmissions for the sink's neighbors is:

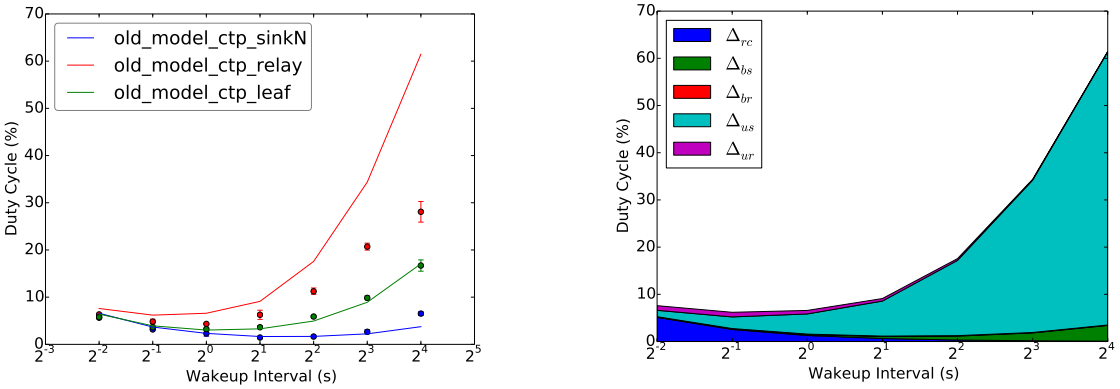
$$\Delta_{us}^{\prime i} = \frac{t_{tx}}{t_{IPI}} F_i$$

(5) *Unicast receptions*
Letting L_i denote the total number of packet receptions (both intended and unintended) at node i, the contribution of unicast receptions to the duty cycle is

$$\Delta_{ur}^{i} = \frac{t_{rx}}{t_{IPI}} L_i \tag{4.16}$$

Summing up Equations 4.12 to 4.16 we get the final expression for the overall duty cycle of a node:

$$\Delta_{dc}^{CTP} = \Delta_{rc} + \Delta_{bs} + \Delta_{br} + \Delta_{us} + \Delta_{ur} \tag{4.17}$$



(a) Comparison between existing model and real measurements

(b) Duty cycle breakdown into the 5 primitives for relay nodes

Figure 4.16: Existing model's performance

Improved Model We found that the previous model is not that accurate. We conducted experiments in the Indriya testbed [18] with different wakeup intervals t_w. During the experiments we monitored the duty cycle of nodes (using an internal timer to track the time the radio was on), and we also collected information for the required parameters in the model, namely F_i, N_i and L_i. Figure 4.16(a) depicts a clear difference between the modeled values and real measurements. Despite the good match that model's parameters have for leaves and sink neighbors, we observe a dissimilarity as high as 148% for relay nodes. But relays are the most critical group of nodes for network lifetime due to its high forwarding load, and hence we need a more accurate model for them.

After breaking down the energy consumption of nodes into the 5 primitives mentioned in Section 4.2.2, we obtain Figure 4.16(b). We can clearly see that Δ_{us}, namely *unicast transmissions*, constitute the major part of the duty cycle. Thus we can draw the conclusion that the duty cycle of these events is overestimated.

The reason for this overestimation is shown in Figure 4.17. The effect depicted in this figure is also mentioned in [28] but it was neglected and not included in the model:

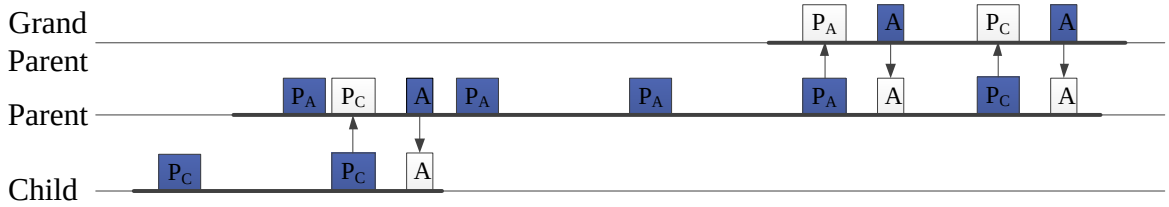


Figure 4.17: One transmission with multiple packets

To explain this phenomenon, we used a two hop communication consisting of a child, parent and grandparent nodes. Before the parent node successfully delivers its packet to the grand-

parent node, the child node generates a packet and sends it to the parent. At this time, the parent node will first acknowledge the child's packet, and then put the packet into its queue. Once the parent and grandparent establish a connection, the parent node will transmit both packets during a single session (instead of using two sessions: one for each packet). Since the rendezvous time (t_w) is much longer than the time required to transmit a packet, this event is equivalent to transmitting two packets for the cost of one. Due to this event, the forwarding load F_i used in Equation 4.15 is inappropriate and overestimated.

We update the SoA equations with this effect. First, let us assume that during one rendezvous session a node can transmit f_{extra} additional packets besides the original one. Then, Equation 4.15 becomes

$$\Delta_{us} = \frac{t_w/2}{t_{IPI}} \frac{F_i}{1 + f_{extra}} \tag{4.18}$$

We now model the circumstances under which f_{extra} occurs. This effect can be divided into the following two scenarios, illustrated in Figure 4.18:

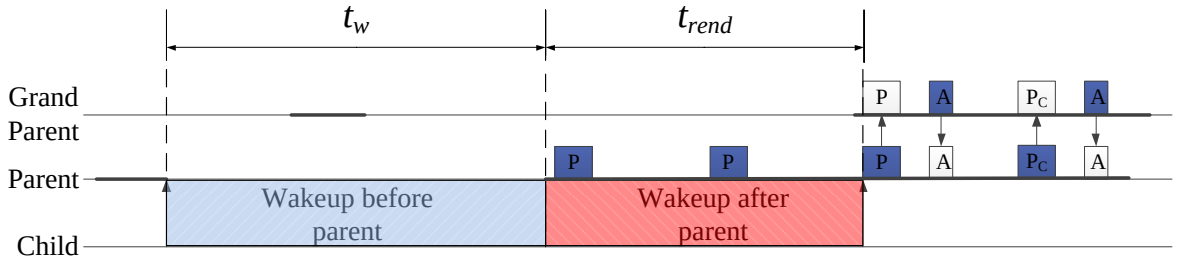


Figure 4.18: The timing that triggers "one transmission with multiple packets" for CTP

(1) If the child node wakes up before the parent node, then it has a window of opportunity of at most t_w to transmit its packet to the parent during the current wakeup period. This window of opportunity is denoted as the blue area in Figure 4.18. Any time longer than t_w will result in letting the parent node receive the packet in the previous wakeup period.

(2) If the child node wakes up after the parent node, then the parent node has an expected rendezvous time of $t_{rend} = t_w/2$ before transmitting its packet to the grandparent. Hence, the child node must wake up within this period, i.e within the red area. Otherwise, the child would need to wait for the subsequent wake up period.

Thus the total time available for the child node to inject a packet to cause a "multi-packet transmission" is $t_w + \frac{1}{2}t_w$. Considering that a node generates a packet every t_{IPI}, the corresponding probability for this phenomenon to occur is $p = \frac{t_w + \frac{1}{2}t_w}{t_{IPI}}$. Letting Q be the size of the node's transmission queue, and D_i be the number of children of node i, denoting $S = min(Q, D_i)$, the expected value of f_{extra} is given by:

$$f_{extra} = \sum_{k=1}^{S} k C_{D_i}^{k} p^k (1-p)^{D_i - k}$$

But considering that the child node itself may also hold multiple packets, we can use the actual forwarding load F_i to substitute D_i[3]. Then we get:

$$f_{extra} = \sum_{k=1}^{S} kC_{F_i}^{k} p^k (1-p)^{F_i - k} \tag{4.19}$$

We now can get a new model by substituting Equation 4.19 into Equation 4.18 to replace Equation 4.15. Figure 4.22 shows the results of applying our new model. For the sink's neighbors the values remain unchanged because the sink is always "on", and leaves are almost not influenced (for they seldom forward packets). But Equation 4.18 evidently improves the quality of the model for relay nodes.

Model for ORW

For ORW, the model is quite similar to the one of CTP, except for the fact that ORW doesn't have beacon events. Strictly speaking, ORW does have something similar to beacons, but it is only used at the very beginning and when a node cannot find a route. Thus it can be neglected. CTP on the other hand, uses beacons aggressively at the beginning, and once the routing topology is formed, it uses beacons every eight minutes to maintain the routes. Similar to what we did in Section 4.2.2, we propose the following duty cycle model for ORW:

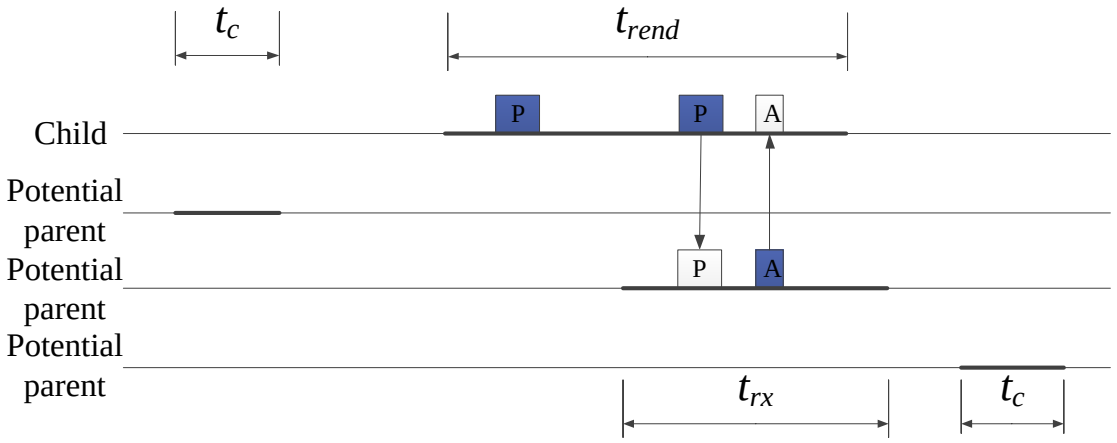


Figure 4.19: A typical anycast transmission in ORW. The bold lines represent the radio 'on' time of the nodes.

(1) *CCA events*
Same as CTP. Please refer to equation (4.12).

(2) *Anycast transmissions*
Figure 4.19 shows a typical anycast process. Compared to unicast, nodes in ORW will

[3]Since the computation of combinatorials require an integer, we use the rounding result of F_i. But for convenience, we still use the symbol F_i, and now $S = min(Q, F_i)$.

utilize any node that (1) wakes up first and (2) provides routing progress towards the sink (instead of selecting a fixed parent). In this way, ORW significantly shortens the rendezvous time because ti does not need to wait until the *designated* parent wakes up. The disadvantage of this routing method is that ORW tends to use routes that are longer than CTP's, which increase the total number of transmissions in the network. Letting P_i be the number of potential parents of node i, it can be proved that the expected rendezvous time is $\frac{t_w}{1+P_i}$ if we assume 100% reliable links [21].

ORW also has the same "multi-packet transmission" effect mentioned in Section 4.2.2, because both protocols adopt the same MAC. Figure 4.20 depicts the effect using the same classification used for CTP (before and after the wake up of the parent):

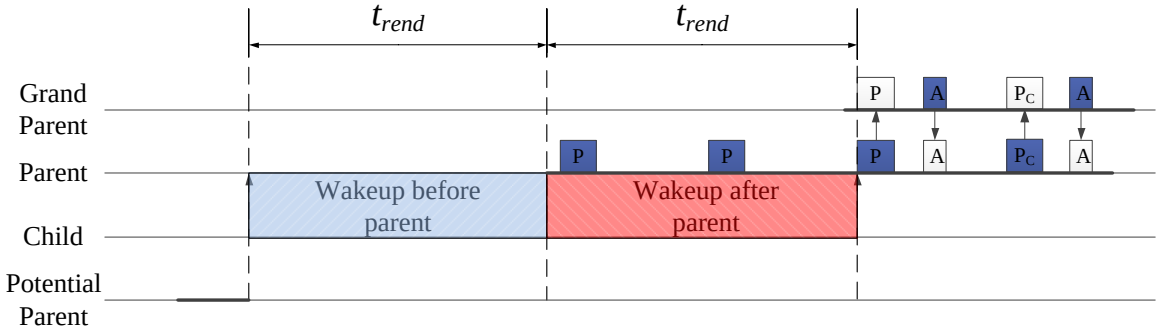


Figure 4.20: The timing that triggers "one transmission with multiple packets" for ORW

(1) The child node wakes up before a potential parent. To trigger the multi packet effect in ORW, a node can wakeup at most t_{rend} before its potential parent. Any time longer than t_{rend} will result in letting another potential parent acquire the packet.
(2) The child node wakes up after its potential parent. Under this circumstance, a node also has t_{rend} to transmit a packet that triggers a multi-packet effect.

Thus the total time available for a child node to inject a packet is $2t_{rend}$, which means that the probability of this effect is $p = \frac{2t_w}{(1+P_i)t_{IPI}}$. Considering this probability, f_{extra} is given by:

$$f_{extra} = \sum_{k=1}^{S} kC_{F_i}^k p^k (1-p)^{F_i - k}$$

And the final contribution of this primitive to the duty cycle is:

$$\Delta_{as} = \frac{t_w}{(1+P_i)t_{IPI}} \frac{F_i}{1+f_{extra}} \tag{4.20}$$

Again, considering that the sink is always on, for sink neighbors the equation simply becomes

$$\Delta_{as} = \frac{t_{tx}}{t_{IPI}} F_i$$

(3) *Anycast receptions*
Same as CTP. Please refer to Equation (4.16), but here we represent it with Δ_{ar}.

Overall, the final equation describing the duty cycle of nodes running ORW is:

$$\Delta_{dc}^{ORW} = \Delta_{rc} + \Delta_{as} + \Delta_{ar} \tag{4.21}$$

Comparison of models

In this section we will make a preliminary comparison between the models derived for CTP and ORW in the previous sections. Table 4.2 lists the primitives used in the models. From the information in this table we can derive two clear results:

(1) For broadcast related primitives, ORW outperforms CTP under all conditions (because ORW has no broadcast events during its steady state).

(2) For the other primitives, the two protocols have an almost identical energy consumption, except for the complex unicast and anycast transmissions of non-sink neighbors nodes (top sub-row of "Uni/Anycast Tx").

	CTP	ORW
CCA events	$\frac{t_c}{t_w}$	$\frac{t_c}{t_w}$
Beacon Tx	$\frac{t_w}{t_{IBI}}$	N/A
Beacon Rx	$\frac{t_{rx}}{t_{IBI}} N_i$	
Uni/Anycast Tx	$\frac{t_w}{2t_{IPI}} \frac{F_i}{1+f_{extra}}$	$\frac{t_w}{(1+P_i)t_{IPI}} \frac{F_i}{1+f'_{extra}}$
	$\frac{t_{tx}}{t_{IPI}} F_i$	$\frac{t_{tx}}{t_{IPI}} F'_i$
Uni/Anycast Rx	$\frac{t_{rx}}{t_{IPI}} L_i$	$\frac{t_{rx}}{t_{IPI}} L'_i$

Table 4.2: Comparison of each factor between CTP and ORW

Therefore we will breakdown the nodes into three groups, sink's neighbors, relays and leafs to offer a deeper view about the Uni/Anycast transmission on different parts of the network. Table 4.3 lists the results after removing common factors in each row:

Table 4.3 reveals some insights about both protocols (considering only Uni/Anycast transmission):

(1) If a node has only one parent, then ORW performs identical to CTP for leaves. With two or more potential parents, ORW is much better in terms of duty cycle. Hence, overall, the higher the density the better ORW should perform.

(2) The sink's neighbors' Uni/Anycast transmissions are only related to their forwarding load. Thus a more balanced routing protocol can put off the first occurrence of failure among sink neighbors. In general, ORW is more balanced than CTP, hence ORW should do a better job in maximizing the minimum lifetime of the sink's neighbors.

	CTP	ORW
Leaves	1/2	$\frac{1}{P_i+1}$
Sink neighbors	F_i^{ctp}	F_i^{orw}
Relays	$\frac{1}{2(1+f_{extra})}F_i^{ctp}$	$\frac{1}{(P_i+1)(1+f'_{extra})}F_i^{orw}$

Table 4.3: Comparison of Uni/Anycast transmission primitives among leaves, sink neighbors and relays

However, for relays the model indicates a complex interaction among load balance, communication primitives and particular phenomena, such as the multi-packet effect. Therefore, to obtain a clear comparison, we ran three one-hour experiments with a wakeup interval 1s. Then we extracted the parameters from the traces for the three types of nodes and averaged them. The results are listed in Table 4.4.

	CTP			ORW		
Classes	SN	RL	LF	SN	RL	LF
Forwarding Load	5.96	5.13	1.01	4.22	2.95	1.24
# of Parents	1	1.20	1.04	1.21	10.26	1.21
f_{extra}	N/A	0.1	0.01	N/A	0.01	0

Table 4.4: Parameter-based evaluation of duty cycle. SN: sinks neighbors, RL: relays, LF: leaves.

From Table 4.4 we can get the following information:

(1) For ORW the effect of multiple transmissions in one rendezvous session can be neglected.

(2) CTP's relays have higher average load than ORW. Now, considering that Equation 4.18 and Equation 4.20 also capture the cost of communication primitives (besides forwarding load), we hyphotesize that relay nodes in CTP have a double burden: they perform many transmissions and each transmission is more expensive than in ORW.

4.2.3 Validation and Verification (D-2.3)

In this chapter we first show testbed results that validate the model constructed in Chapter 4.2.2. Next we present additional experiments related to network lifetime and analyze the

results. We'll show how the network lifetime is affected by the energy budget, node density and network scale. Last we'll discuss some peculiar phenomena at low densities.

Experimental Setup

This section introduces the common settings for all the experiments. The settings that are particular to each type of experiment will be discussed in the corresponding sections.

General Conditions We try to keep each protocol "as it is", but there are still some things we modify. The main discrepancies between the default implementation and our modifications are list next, together with the reasons for these changes.

- *CCA Time*
  This is a parameter that controls the duration of the Clear Channel Assessment (CCA), that is, the amount of time that the node remains awake every time it wakes up. According to [28], the CCA time should be $t_c = t_{backoff} + t_{ack}$, where $t_{backoff}$ is the CSMA back off time, and t_{ack} is the delay of the acknowledgement. The default value is set to 6 ms, which may lead to the following undesirable scenario, depicted in Figure 4.21:

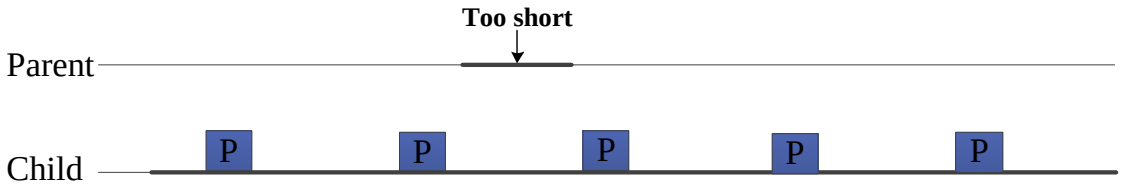


Figure 4.21: CCA time is so short that parent node misses the packets

  Since the default CCA time is shorter than the interval between two packets, it's possible that the parent node wakes up just between two consecutive transmissions of the child. If this occurs, the child node has to wait for at least another t_w to rendezvous with its parent. For the next time there is also no guarantee that they will detect each other. This is not a desired behavior. It is advised in the TinyOS source code to increase this value according to the specific platform. Thus, we increase the CCA time to about 12 ms, which is just long enough to avoid this potential problem.

- *Packet Interval*
  This parameter controls how frequently a node generates data. The default values for CTP and ORW are 2 packet/min and 0.25 packet/min, respectively. In our experiments they are both set to 1 packet/min, which provides a data rate that is common to sensor networks applications, while still guaranteeing that enough data packets are transmitted to analyze the lifetime of the network (some public testbeds only allow slots of 30 minutes and hence the data rate can not be too low).

We always employ the data collected ten minutes after the startup of the network to ensure that the results are not influenced by the high variability of the starting phase.

Testbed Specification We conduct our testbed experiments in Indriya, a publicly available testbed containing 100 active nodes (July, 2014). The sink is located at the corner of the testbed to ensure the largest possible diameter of the network in terms of number of hops.

Model Validation

This section validates our model with empirical results from the Indriya testbed. Results for CTP and ORW are presented separately in the following sections. To validate our models for both protocols, we used the following procedure: while running the protocols we measured the duty cycle of the nodes, which represent the ground truth, and we also measured the required parameters for our models (forwarding load, listening load and number of potential parents). The goal is to observe if the duty cycle estimated by using these parameters is similar to the actual duty cycle measured at the nodes.

Results for CTP We measure the duty cycle under different wakeup intervals, ranging from 250 ms to 16 s. The duty cycle is calculated as $\Delta = \frac{t_{on}}{t_{all}}$, where t_{on} is the total time when the radio is on, and t_{all} stands for the duration of the measurement. Every experiment lasts for one hour in total. For each wakeup interval we run the experiment three times. The average duty cycles and their corresponding standard deviations are plotted in Figure 4.22. The dashed lines represent the SoA model while the solid lines represent our new model. Table 4.5 quantifies the improvement of our model compared to the SoA model. The improvement of our model is calculated as:

$$improvement = \frac{|\Delta_{oldmodel} - \Delta_{real}| - |\Delta_{newmodel} - \Delta_{real}|}{\Delta_{real}}$$

where Δ_{real} represents the real measured duty cycle, $\Delta_{oldmodel}$ stands for the value calculated by the SoA model, and $\Delta_{newmodel}$ is the value calculated by our model.

$t_w(s)$		2^{-2}	2^{-1}	2^0	2^1	2^2	2^3	2^4
$\frac{\Delta_{model}-\Delta_{real}}{\Delta_{real}}(\%)$	Old Model	19.83	28.37	52.17	45.26	56.38	66.03	118.81
	New Model	19.05	25.36	43.21	29.81	15.83	-3.24	6.43
Improvement(%)		3.90	10.59	17.19	34.14	71.92	95.09	94.59

Table 4.5: Comparison between old and new models against real measurements in detail for CTP

From the results we can clearly see that our new model successfully:

- Reduces the overestimation in the old model by 95.09% for relay nodes. The smallest error against real measure values is only 3.24%.

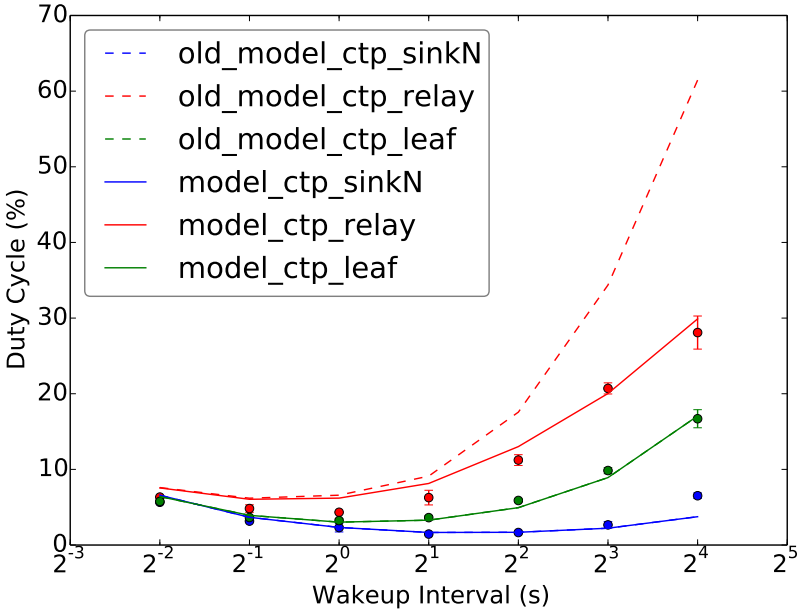


Figure 4.22: Result of new model after applying the fix

- Maintains good performance for leave nodes and sink neighbors.

Thus we claim that our model is a good improvement over the SoA.

Results for ORW To have a fair comparison with CTP, the experimental setups for ORW are identical to the ones for CTP. Please refer to Section 4.2.3 for more details. Given that there is no existing model for ORW, we quantify the advantages of identifying the problem described in Section 4.2.2 by comparing two models: the model that does not consider the effect in Section 4.2.2 is called "direct ported model", and the one considering this effect is called the "new model". Figure 4.23 and Table 4.6 provide our results.

Similarly to CTP, the model for ORW also reduces the overestimation of duty cycle for relays. This improvement confirms that the multiple-packets-transmission event is a phenomenon that needs to be considered.

However, unlike CTP results, we observe an underestimation of the duty cycle for leaves nodes. We hypothesize that this occurs due to our assumption of perfect links. This assumption is not a problem for CTP because CTP tends to choose links with high quality. But, ORW utilizes a pool of potential parents whose links' quality change continuously in time. This means that at a given time, the actual number of parents that are qualified to forward the packet is less than the total number of parents observed up to that time. For P_i we use the total number of parents, and thus, we overestimate this parameter. Recalling Equation 4.20, we see that if P_i is overestimated, it will result in an underestimation on Δ_{as}. Therefore, when applying our model, a penalty on the potential number of parents should be taken into consideration, depending on the link communication quality.

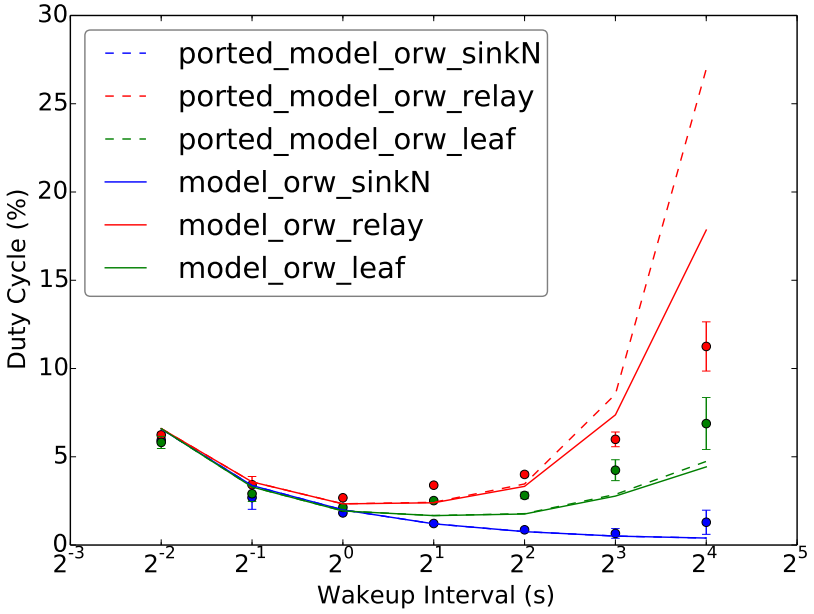


Figure 4.23: Comparison between the direct ported model and the new model for ORW

$t_w(s)$		2^{-2}	2^{-1}	2^0	2^1	2^2	2^3	2^4
$\frac{\Delta_{model}-\Delta_{real}}{\Delta_{real}}(\%)$	Old Model	5.74	4.73	-12.72	-28.61	-13.49	42.71	139.72
	New Model	5.74	4.70	-12.87	-29.37	-16.91	23.18	58.61
Improvement(%)		0.07	0.62	-1.13	-2.64	-25.29	45.72	58.05

Table 4.6: Comparison between the direct ported model and the new model against real measurements for ORW

Experiments on network lifetime

In this section we conduct experiments related to network lifetime, and explain the results using the models constructed in Chapter 4.2.2. We use the radio 'on' time as a proxy for energy consumption. Each node is given a fixed amount of energy budget in terms of radio 'on' time. Once a node exhausts its budget, it will shut down as if it "dies". In this way we can capture the dying process of the network.

There are many definitions of network lifetime related to specific applications. We cannot give an evaluation that covers every aspect. In our evaluation we focus on the following points:

- Illustrate the dying process using the total number of nodes alive
- Illustrate the dying process using the number of total connected nodes that have paths to the sink
- Illustrate the dying process for different classes of nodes (sink neighbors, leaves and relays)

Influence of Energy Budget Considering that we do not have battery models, in this section we present results for different values of initial energy to reveal the influence of the energy budget on network lifetime. In the next section, we describe briefly how temperature effects can reduce the lifetime of batteries. Thus, these two sections give a rough idea of hot temperature can affect the lifetime of data collection networks.

Figure 4.24 to Figure 4.27 depict the dying process of the network when the energy limits are 128s and 256s. Figure 4.24 and Figure 4.26 reflect the dying process in terms of the total number of nodes alive. Every point shows the average result from three experiments, and the standard deviations.

To have a more detailed view of the dying process, Figure 4.25(a) and Figure 4.27(a) compare the total number of received packets by the network (top graph), the total throughput of the network (middle graph) and the dying process of each group of nodes (bottom graph). Figure 4.25(b) and Figure 4.27(b) show the relationship between the dying time and the number of hops to the sink. Nodes with higher loads have darker colors.

From the results we can see that, almost at any point, either in terms of the total number of nodes alive or from the perspective of different classes of nodes, ORW has longer lifetimes than CTP. This is not surprising, as in Chapter 4.2.2 we already showed that for every individual node, each transmission in ORW takes shorter rendezvous time than CTP. And from Figure 4.25(b) we can see a more evenly distributed load for ORW, which helps to avoid some nodes from dying too fast.

Overall, having a longer lifetime allows ORW to deliver more packets for the same initial energy budget. And as the energy budget increases, the advantage of ORW becomes stronger.

For the experimental setups in this section, we did not observe a big difference between the lifetimes of connected nodes (having a path to the sink) and all nodes (connected and disconnected), shown in Figure 4.28. This is probably due to the good connectivity of the network. With a high density, nodes can easily find new parents when some of its neighbors die. If the density is low, there would be fewer nodes that can provide routing progress, thus the number of potential parents P_i would decrease for ORW. Equation 4.20 predicts that if P_i is close to 1, then the advantage of ORW's anycast will diminish. So it would be interesting to explore how the network would perform with lower densities, which will be addressed in the following section.

Influence of Temperature on Battery Lifetime If we consider the previous section as a function f(battery's energy) that maps the battery's energy level to the network lifetime, this section gives an *initial qualitative description* of what could be a function $g(temperature)$ that captures the effect of the environmental temperature on the battery's energy level. Hence, with extra work, the combination of these two functions $f(g(temperature))$ would capture the effect of temperature on the network lifetime.

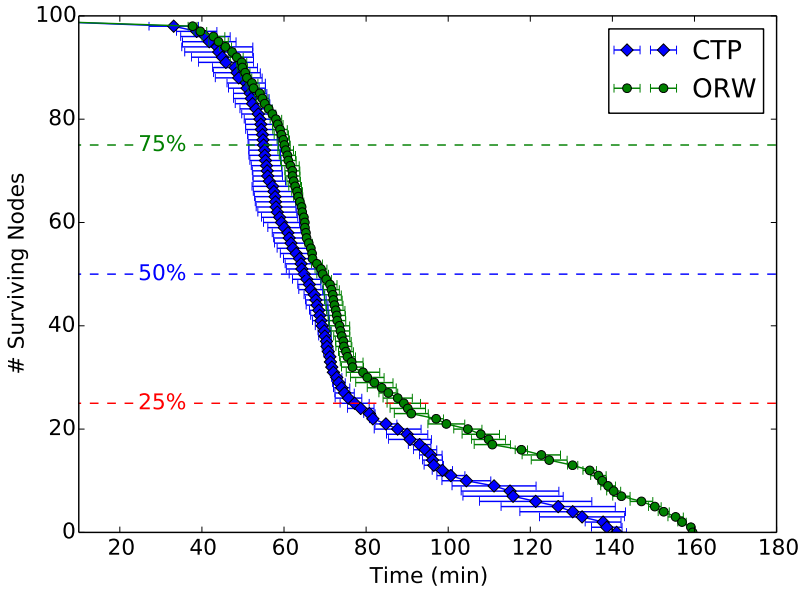
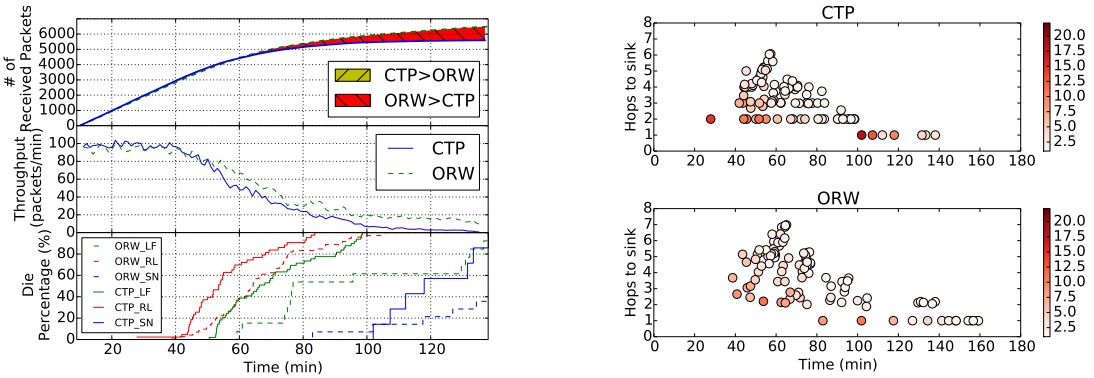


Figure 4.24: Dying process with energy limit 128s in terms of all nodes



(a) Dying process of different classes of nodes

(b) Relationship among dying time, hops to sink and load

Figure 4.25: Dying process with energy limit 128s of one run

Typically batteries are made to work optimally at room temperatures; from $25°C$ to $27°C$ [1, 2]. This means that the size and internal chemistry of a battery is optimized for best performance at room temperature. Any significant variations in the room temperature results in either a lower battery capacity or shorter lifetime. In the short term, operating a battery at higher temperatures improves its capacity as high temperature result in lowering the battery's internal resistance and speeding up the chemical metabolism. However, if high temperatures persist for a long period of time, the lifetime of the battery is reduced [1, 2]. The exact relation between temperature and battery lifetime, depends on the type and manufacturer of the bat-

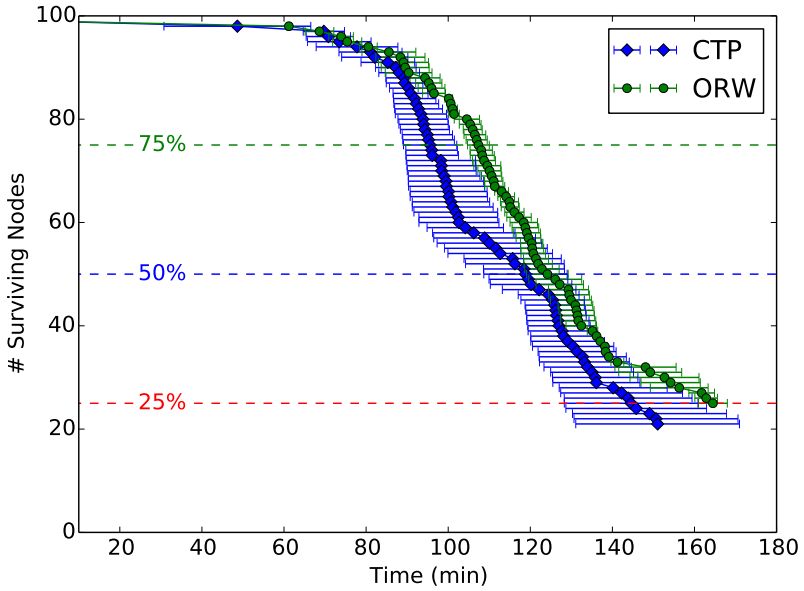
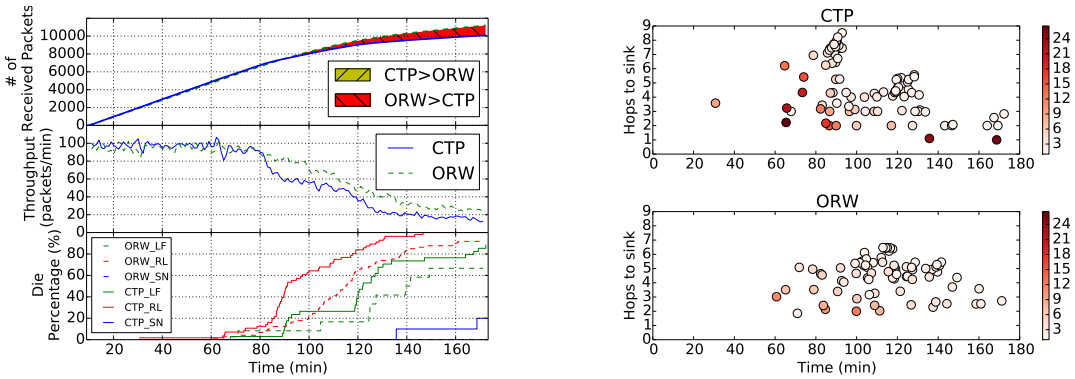


Figure 4.26: Dying process with energy limit 256s in terms of all nodes



(a) Dying process of different classes of nodes

(b) Dying process in terms of all nodes

Figure 4.27: Dying process with energy limit 256s of one run

tery. As a guideline, every $8°C$ increase in temperature reduces the life of a lead acid battery in half. That is, a lead acid battery that would last a period t at $25°C$ (room temperature) would only last for $t/2$ if operated at $33°C$. According to our lifetime models, high temperatures would accelerate the death of all nodes, in particular *relay* nodes, which are the nodes whose death bootstraps the overall death process. Also, asynchronous shortest-path-based protocols, like CTP, would lead to even shorter network lifetimes, as compared to asynchronous opportunistic protocols like ORW.

In contrast to high temperature effects, colder than room temperatures reduce the capacity

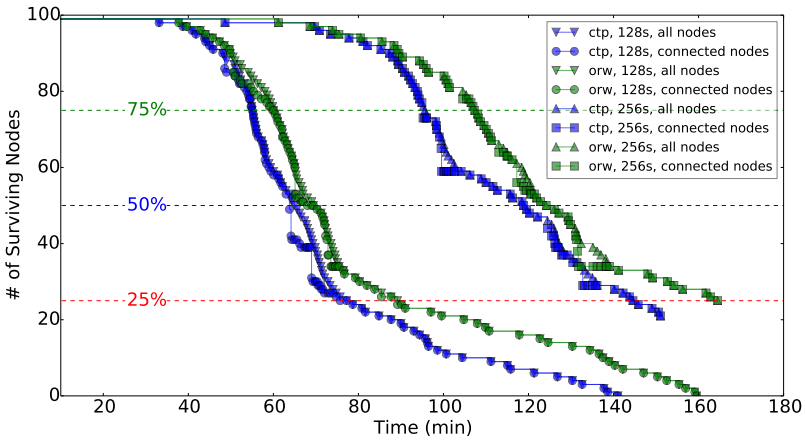


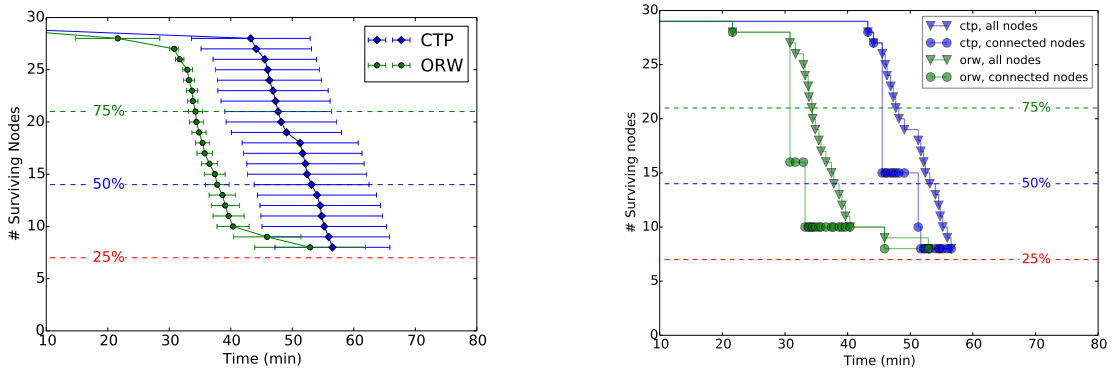
Figure 4.28: Lifetime of connected nodes vs all nodes

of a battery but increases its lifetime. This is because cold temperature increases the internal resistance of the battery by slowing down the chemical metabolism. A lead acid battery that would provide 100% percent capacity at $27°C$ will typically deliver only 50% at $-18°C$ [2]. The general relationship of low temperature and battery performance holds when a battery of different type is being used, however the specifics are different for other types of batteries. For instance, in [43], the authors report that a plug-in hybrid electric vehicle (PHEV) battery's lifetime reduces 3 years (or 10%) when used in Phoenix as compared to Miami, USA. This is because, the summer in Phoenix was on average $5.5°C$ hotter than summer of Miami. In [38], a battery lifetime model for wireless sensor networks is proposed, however this model does not discuss in detail how battery lifetime is affected by the variation in temperature. Overall, lower temperatures may affect the correct operation of the electronics, because of the lower input current, but the lifetime should increase.

Influence of Network Density In these experiments, we select one third of the total nodes in the network in an even manner. Under this condition, 29 connected nodes are reported. All other parameters are kept the same as in the previous section. The energy limit is set to 128s.

Figure 4.30 shows the result after reducing the node density. Different from Figure 4.28, there is a significant difference in the dying process. In sparse networks, extending the lifetime of nodes becomes more critical than in dense networks, because some nodes may be the only choice towards the sink for their children. This "bottleneck" effect explains why there is a sharp drop in throughput (Figure 4.30(a)).

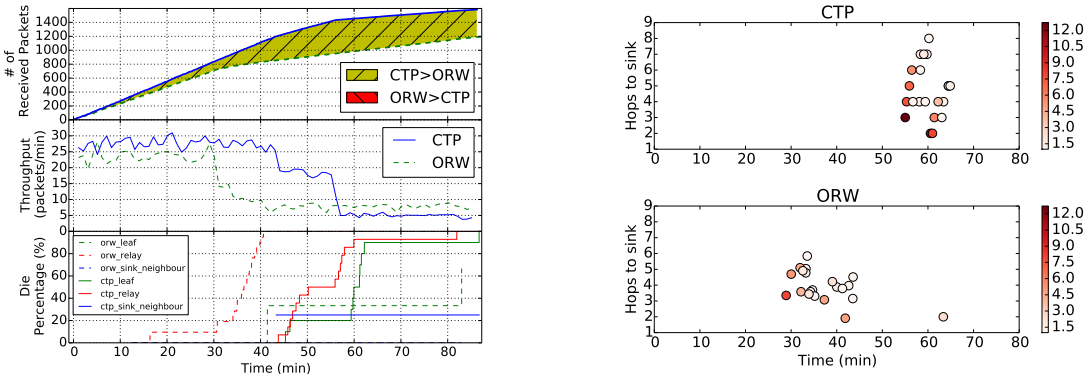
Under low densities, we can see clearly that ORW performs worse than CTP in at least two aspects: energy consumption and throughput. For energy consumption, although both protocols die earlier than the experiments in the previous section, ORW suffers more. This is counter intuitive, since even with P_i equals one, our model suggests that ORW should approximate CTP's performance, rather than performing evidently worse. Another important aspect we should notice is that ORW never reaches the maximum possible throughput, while CTP does a good job. This indicates that ORW may suffer from some inherent issues that may be related to actual implementation rather than the design of the protocol. Before we go



(a) Dying process in terms of all nodes

(b) All nodes vs Connectivity

Figure 4.29: Dying process after reducing the density, with energy limit 128s



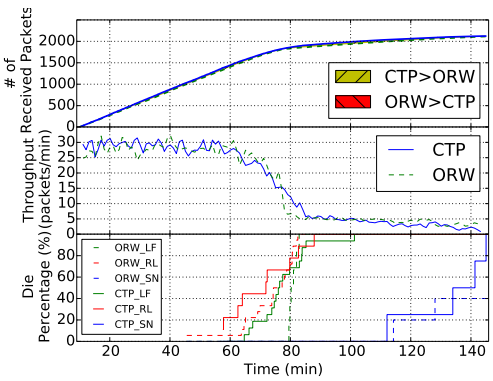
(a) Dying process in terms of all nodes

(b) Relationship among dying time, hops to sink and load

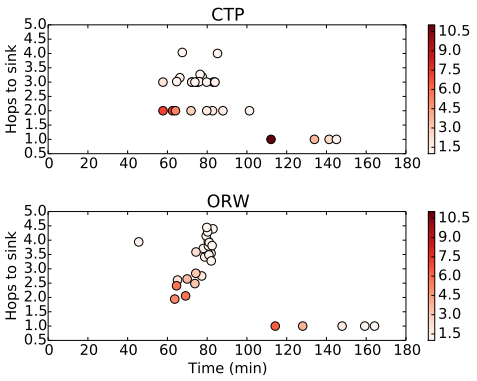
Figure 4.30: Dying process with energy limit 128s of one run, 29 nodes with low density

into a further discussion, first let us prove that it's indeed density and not network scale what influences the performance of the two protocols.

Influence of Network Scale In this section, we run experiments with the same number of nodes used in Section 4.2.3 (29 nodes). We carefully select the nodes so that the density is close to the environment in 4.2.3. Figure 4.31(a) shows that for the most part, the curves of ORW and CTP overlap with each other, suggesting that their performance is similar. Compared to Figure 4.30, the only difference is density. Thus we can confirm that it is a low-density rather than a small-scale what makes ORW perform worse. In order to have a deep understanding about why this happens and to capture some details that cannot be traced by the logging inside the code, we decided to perform some simulations.



(a) Dying process of different classes of nodes



(b) Relationship among dying time, hops to sink and load

Figure 4.31: Dying process with energy limit 128s of one run, 29 nodes with high density

Discussion on Peculiar Phenomenon

We run our simulations in Cooja [33], a simulator that was developed for the Contiki operating system. We use the *Unit Disk Graph Model with exponential distance loss* as the radio model. This is an ideal transmission model that does not hold in practice, but it is a good model to identify problems in the implementation of protocols. Each node has a start delay chosen randomly within 10s to avoid synchronization. The network is a 6x6 grid, in which every node can talk to at most eight neighbors around it. We simulated ORW for 1 hour without energy limits. Through simulations we identify the following possible reasons why ORW performs worse in low-density networks:

(1) *Parent becoming child*

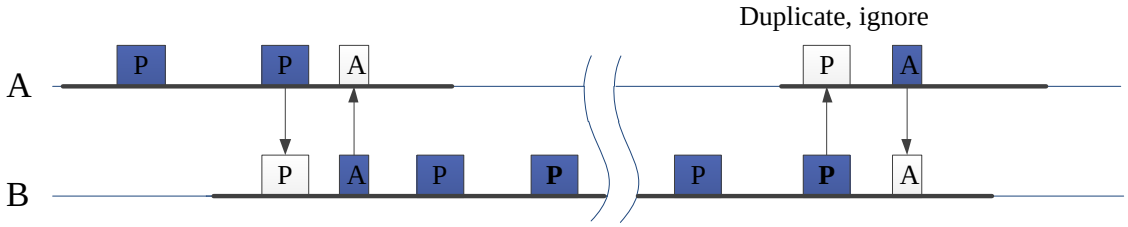


Figure 4.32: Parent becoming child

Figure 4.32 depicts one of the reasons why ORW performs poorly at low densities. Node A successfully delivers the packet to B, but due to the low density, B cannot find a suitable parent to forward this packet. B has to keep on transmitting until it receives an acknowledgement from some node. If the transmission lasts too long, a penalty will be added to B's routing metric. When B's metric increases beyond a certain threshold, even the original child A will be able to satisfy the requirement, i.e. a loop occurs. At this time, if A receives

the packet, it will acknowledge it, because its own routing metric is qualified to forward this packet. Given that node A has already seen this packet before, the packet will be regarded as a duplicate and will be ignored, leading to a packet lost.

(2) *Receiving multiple packets and sending a single ack to the wrong node*

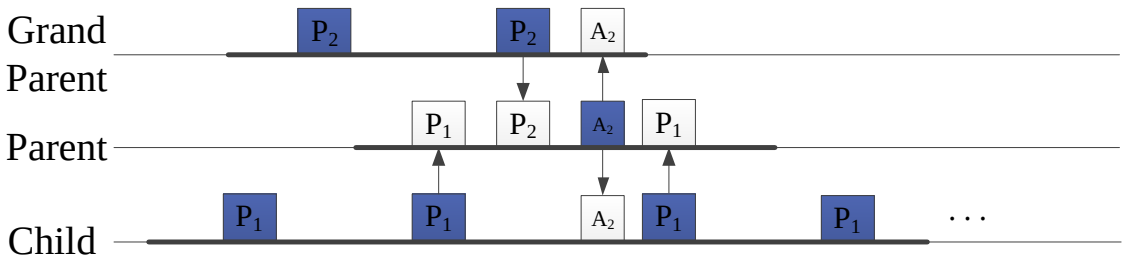


Figure 4.33: Wrong acknowledgement

Figure 4.33 shows another undesirable effect. After the parent node receives the child's packet, a grandparent's packet arrives before the parent issues the required acknowledgement to the child. Normally, the parent node should only acknowledge the packet from child node, but in this case, it wrongly sends out an acknowledgement for the packet from the grandparent node.

This event directly stops the grandparent's transmission, since it receives an acknowledgement and there is no need to continue transmitting the strobe of packets. But the parent node is not able to forward this packet, as it has a higher (worse) routing metric than its grandparent. This cause the packet from grandparent to be lost. In the meantime, the child node continues transmitting its packet, because no acknowledgement has been received for it. This causes an extra transmission time, which increases the duty cycle of the child node.

(3) *Although an acknowledgement is received, the original sender keeps on transmitting*

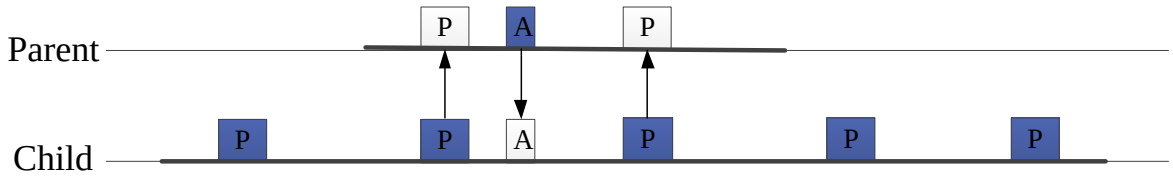


Figure 4.34: Original sender still keep sending after receiving an acknowledgement

As shown in Figure 4.34, although the child node receives the acknowledgement, instead of

stopping the transmission of packets, it still keeps sending. This leads to an unnecessary long transmission time, and consequently, to a higher energy consumption.

(4) *An acknowledgement is sent when receiving the packet for the second time instead of the first*
Instead of acknowledging the packet immediately upon the first reception, the parent node sends out the acknowledgement after the second time it receive the packet (Figure 4.35). Sometimes this is not a problem (except for the extra unnecessary transmission), but in some instances we observe packet suppressions by the parent, which leads to packet losses.

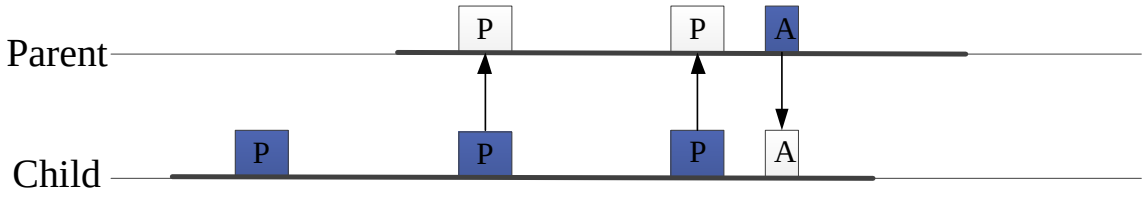


Figure 4.35: Acknowledgement is sent when receiving the packet for the second time

Overall, scenarios 1, 2, and 4, lead to packet losses. Even though all these scenarios can occur in high density networks, sparse networks have a higher probability of triggering such events. Scenario 3 does not necessarily lead to a packet loss, but it increases the nodes' duty cycle since the nodes needs to transmit for an extra time that is not needed. Among the aforementioned scenarios, scenario 2 can also be found in CTP, thus we hypothesize that it may be related to the MAC layer's implementation or hardware issues. The rest of scenarios only occurs in ORW, thus we owe them to the flaws in the implementation of ORW.

5 Conclusions

This deliverable gives closure to the multi-level modeling approach proposed in the Description of Work. A central argument of our consortium has been that *to understand and overcome the pernicious effects of temperature and interference on the operation of sensor and actuator networks, we need to model first the environment, then the platforms deployed on the environment, and finally the protocols operating on the platforms.* The initial efforts in WP1 (Environmental and Platform models) paved the way for the modeling and validation of our protocols in WP2 (Tasks 2.3 and 2.4).

Overall, our contributions can be classified in two macro groups. First, to overcome interference, we modeled and validated four protocols that are intrinsically connected to each other: JAG, MiCMAC, Estimation of Packet Reception Rate, and Radio Energy Prediction. Considering that the main detrimental effect of interference is that of dropping packets, nodes should first try to identify a channel with low interference. To this end, we modeled an agreement protocol that would allow nodes to jump to a common low-interference channel (JAG) and a protocol that jumps over many different channels until encountering a low-interference one (MiCMAC). For scenarios where all channels are under interference, we propose (i) a variable packet size method that minimizes the packet loss by (probabilistically) adjusting the packet size to the length of the idle periods, and (ii) a radio energy prediction method that optimizes the use of channel checks to reduce the energy consumption of nodes. Second, to overcome temperature, we modeled two protocols. One model captures the impact of temperature on the delivery rate (TempMAC). This model includes the various parameters derived for the platform model in WP1. The other protocol, TempLife, is a general framework to analyze the impact of temperature on the network's lifetime.

Bibliography

[1] "How heat and loading affect battery life," http://batteryuniversity.com/learn/article/how_heat_and_harsh_loading_reduces_battery_life, accessed: 2014-10-2.

[2] "Discharging at high and low temperatures," http://batteryuniversity.com/learn/article/discharging_at_high_and_low_temperatures, accessed: 2014-10-2.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[4] K. Bannister, G. Giorgetti, and S. K. S. Gupta, "Wireless sensor networking for hot applications: Effects of temperature on signal strength, data collection and localization," in *Proc. of the 5th Workshop on Embedded Networked Sensors (HotEmNets)*, 2008.

[5] C. A. Boano and K. Römer, "External radio interference," in *Radio Link Quality Estimation in Low-Power Wireless Networks*, ser. SpringerBriefs in Electrical and Computer Engineering - Cooperating Objects, N. Baccour, A. Koubâa, C. A. Boano, L. Mottola, H. Fotouhi, M. Alves, H. Youssef, M. A. Zuniga, D. Puccinelli, T. Voigt, K. Römer, and C. Noda, Eds. Springer International Publishing, Jul. 2013, pp. 21–63.

[6] C. A. Boano, J. Brown, Z. He, U. Roedig, and T. Voigt, "Low-power radio communication in industrial outdoor deployments: The impact of weather conditions and ATEX-compliance," in *Proc. of the* 1^{st} *International Conference on Sensor Networks Applications, Experimentation and Logistics (SENSAPPEAL)*, 2009.

[7] C. A. Boano, J. Brown, N. Tsiftes, U. Roedig, and T. Voigt, "The impact of temperature on outdoor industrial sensornet applications," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 3, pp. 451–459, Aug. 2010.

[8] C. A. Boano, T. Voigt, N. Tsiftes, L. Mottola, K. Römer, and M. A. Zúñiga, "Making sensornet mac protocols robust against interference," in *Proceedings of the 7th European conference on Wireless Sensor Networks*, ser. EWSN'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 272–288. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-11917-0_18

[9] C. A. Boano, K. Römer, F. Österlind, and T. Voigt, "Realistic simulation of radio interference in COOJA," in *Adjunct Proceedings of the* 8^{th} *European Conference on Wireless Sensor Networks (EWSN), demo session*, Feb. 2011, pp. 36–37.

[10] C. A. Boano, T. Voigt, C. Noda, K. Römer, and M. A. Zúñiga, "JamLab: Augmenting sensornet testbeds with realistic and controlled interference generation," in *Proceedings of the* 10^{th} *International Conference on Information Processing in Sensor Networks (IPSN)*, apr 2011, pp. 175–186.

[11] C. A. Boano, M. A. Zúñiga, K. Römer, and T. Voigt, "JAG: Reliable and predictable wireless agreement under external radio interference," in *Proceedings of the* 33^{rd} *IEEE International Real-Time Systems Symposium (RTSS)*. IEEE, Dec. 2012, pp. 315–326.

[12] C. A. Boano, H. Wennerström, M. A. Zúñiga, J. Brown, C. Keppitiyagama, F. J. Oppermann, U. Roedig, L.-Å. Nordén, T. Voigt, and K. Römer, "Hot Packets: A systematic evaluation of the effect of temperature on low power wireless transceivers," in *Proceedings of the* 5^{th} *Extreme Conference on Communication (ExtremeCom)*, Aug. 2013, pp. 7–12.

[13] C. A. Boano, K. Römer, and N. Tsiftes, "Mitigating the adverse effects of temperature on low-power wireless protocols," Under submission., 2014.

[14] 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, pp. 2:1–19, nov 2012.

[15] J. Brown, B. McCarthy, U. Roedig, T. Voigt, and C. J. Sreenan, "Burstprobe: Debugging time-critical data delivery in wireless sensor networks," in *Proceedings of the* 8^{th} *European Conference on Wireless Sensor Networks (EWSN)*, feb 2011, pp. 195–210.

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

[17] K. R. Chowdhury and I. F. Akyildiz, "Interferer classification, channel selection and transmission adaptation for wireless sensor networks," in *Proceedings of the IEEE International Conference on Communications (ICC)*, jun 2009, pp. 1–5.

[18] M. Doddavenkatappa, M. C. Chan, and A. L. Ananda, "Indriya: A low-cost, 3d wireless sensor network testbed," in *Testbeds and Research Infrastructure. Development of Networks and Communities*. Springer, 2012, pp. 302–316.

[19] A. Dunkels, "The ContikiMAC Radio Duty Cycling Protocol," Swedish Institute of Computer Science, Tech. Rep. T2011:13, 2011.

[20] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Proceedings of the* 1^{st} *International Workshop on Embedded Networked Sensors (EmNetS)*, Tampa, FL, USA, nov 2004.

[21] E. Ghadimi, O. Landsiedel, P. Soldati, and M. Johansson, "A metric for opportunistic routing in duty cycled wireless sensor networks," in *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2012 9th Annual IEEE Communications Society Conference on*. IEEE, 2012, pp. 335–343.

[22] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2009, pp. 1–14.

[23] J.-H. Hauer, V. Handziski, and A. Wolisz, "Experimental study of the impact of WLAN interference on IEEE 802.15.4 body area networks," in *Proceedings of the* 6^{th} *European Conference on Wireless Sensor Networks (EWSN)*, feb 2009, pp. 17–32.

[24] J.-H. Hauer, A. Willig, and A. Wolisz, "Mitigating the effects of RF interference through RSSI-based error recovery," in *Proceedings of the* 7^{th} *European Conference on Wireless Sensor Networks (EWSN)*, feb 2010, pp. 224–239.

[25] G. J. Huang, G. Xing, G. Zhou, and R. Zhou, "Beyond co-existence: Exploiting wifi white space for zigbee performance assurance," in *Proceedings of the* 18^{th} *IEEE International Conference on Network Protocols (ICNP)*, oct 2010, pp. 305–314.

[26] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, "Low power, low delay: opportunistic routing meets duty cycling," in *Proceedings of the 11th international conference on Information Processing in Sensor Networks*. ACM, 2012, pp. 185–196.

[27] Y. Liu, Y. He, M. Li, J. Wang, K. Liu, and X. Li, "Does wireless sensor network scale? a measurement study on greenorbs," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 10, pp. 1983–1993, 2013.

[28] D. Moss and P. Levis, "Box-macs: Exploiting physical and link layer boundaries in low-power networking," *Computer Systems Laboratory Stanford University*, 2008.

[29] B. A. Nahas, S. Duquennoy, V. Iyer, and T. Voigt, "Low-Power Listening Goes Multi-Channel," in *Proceedings of the International Conference on Distributed Computing in Sensor Systems (IEEE DCOSS 2014)*, Marina Del Rey, CA, USA, May 2014.

[30] C. Noda, S. Prabh, M. Alves, C. A. Boano, and T. Voigt, "Quantifying the channel quality for interference-aware wireless sensor networks," *ACM SIGBED Review*, vol. 8, no. 4, pp. 43–48, nov 2011.

[31] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*, Tampa, Florida, USA, Nov. 2006.

[32] F. Österlind, J. Eriksson, and A. Dunkels, "Cooja timeline: a power visualizer for sensor network simulation," in *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Zurich, Switzerland, 2010.

[33] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*. IEEE, 2006, pp. 641–648.

[34] W.-B. Poettner, H. Seidel, J. Brown, U. Roedig, and L. Wolf, "Constructing schedules for time-critical data delivery in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 10, no. 3, 2014.

[35] D. Puccinelli and M. Haenggi, "Reliable data delivery in large-scale low-power sensor networks," *ACM Trans. on Sens. Netw.*, vol. 6, no. 4, 2010.

[36] D. Puccinelli, S. Giordano, M. Zuniga, and P. J. Marrón, "Broadcast-free collection protocol," in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*. ACM, 2012, pp. 29–42.

[37] T. S. Rappaport *et al.*, *Wireless communications: principles and practice*. prentice hall PTR New Jersey, 1996, vol. 2.

[38] C. Rohner, L. M. Feeney, and P. Gunningberg, "Evaluating battery models in wireless sensor networks," in *Wired/Wireless Internet Communication*. Springer, 2013, pp. 29–42.

[39] H. Shariatmadari, A. Mahmood, and R. Jäntti, "Channel ranking based on packet delivery ratio estimation in wireless sensor networks," in *Proceedings of the Wireless Communications and Networking Conference (WCNC)*, apr 2013, pp. 59–64.

[40] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "An analysis of a large scale habitat monitoring application," in *ACM SenSys*, 2004.

[41] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong, "A macroscope in the redwoods." in *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, 2005.

[42] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI)*, Seattle, USA, November 2006.

[43] T. Yuksel and J. Michalek, "Evaluation of the effects of thermal management on battery life in plug-in hybrid electric vehicles," in *Society of Automotive Engineers World Congress*, 2012.

[44] S. Zacharias, T. Newe, S. O'Keeffe, and E. Lewis, "Identifying sources of interference in rssi traces of a single IEEE 802.15.4 channel," in *Proceedings of the 8th International Conference on Wireless and Mobile Communications (ICWMC)*, jun 2012, pp. 408–414.

[45] M. Zimmerling, F. Ferrari, L. Mottola, T. Voigt, and L. Thiele, "ptunes: runtime parameter adaptation for low-power mac protocols," in *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, 2012, pp. 173–184.

[46] M. Zúñiga, , C. Boano, J. Brown, C. Keppitiyagama, F. Oppermann, P. Alcock, N. Tsiftes, U. Roedig, K. Römer, T. Voigt, , and K. Langendoen, "D-1.1 - report on environmental and platform models," http://www.relyonit.eu/, RELYonIT: Research by Experimentation for Dependability on the Internet of Things, Grant Agreement no: 317826, Tech. Rep., Jun. 2013.

[47] M. Zúñiga, F. Aslam, I. Protonotoarios, K. Langendoen, C. Boano, K. Römer, J. Brown, U. Roedig, N. Tsiftes, and T. Voigt, "D-2.1 - report on optimized and newly designed protocols," http://www.relyonit.eu/, RELYonIT: Research by Experimentation for Dependability on the Internet of Things, Grant Agreement no: 317826, Tech. Rep., May 2014.