RELYonIT
dependability for the internet of things

Research by Experimentation
for Dependability on the
Internet of Things

SEVENTH FRAMEWORK
PROGRAMME

D-1.2 – Report on Learning Models Parameters

Grant Agreement no: 317826
www.relyonit.eu

Date: November 14, 2013

Author(s) and affiliation: James Brown (ULANC), Ibrahim Ethem Bagci (ULANCS), Utz Roedig (ULANC), Marco Antonio Zúñiga (TUD), Carlo Alberto Boano (TUG), Nicolas Tsiftes (SICS), Kay Römer (TUG), Thiemo Voigt (SICS), and Koen Langendoen (TUD).

Work package/task: WP1
Document status: Final
Dissemination level: Public
Keywords: Wireless sensor networks, temperature, interference, received signal strength, timing, analytical models, collection tools.

Abstract This documents presents (i) refinements to the environmental and platform models that we presented in Deliverable D-1.1, (ii) data collection tools to parametrise these models, and (iii) methods for model bounding and aggregation. We first summarize the models presented in Deliverable D-1.1, and then show how we have enhanced and extended them, focussing on temperature, radio interference, received signal strength, and timing.
Each of these four aspects is then considered individually: we present tools that can gather data to instantiate the models for a particular environment or hardware platform, and then show how the collected datasets can be used in practice to instantiate each model. We further derive bounding methods for each model, and investigate how models derived from datasets collected on individual nodes can be aggregated into a single model for the deployment environment or hardware platform.

Disclaimer

The information in this document is proprietary to the following RELYonIT consortium members: Graz University of Technology, SICS 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

Executive Summary

The performance of IoT systems and applications is strongly affected by two key components: the surrounding environment and the hardware platform. Denoting $\mathcal{N}$ as the performance of a network, $\mathcal{E}$ as the behaviour of the environment and $\mathcal{H}$ as the behaviour of the hardware platform, the relationship among the network performance, the environment, and the hardware platform is given by $f : \mathcal{E}(e), \mathcal{H}(h) \rightarrow \mathcal{N}$. While in Deliverable D-1.1 [16] our focus was on modelling the environment and the platform (i.e., our goal was to formally derive the behaviour of the environment $\mathcal{E}$ and of the hardware platform $\mathcal{H}$), in this deliverable our focus is on the model parameters e and h.

Our goal is twofold. First, we aim to learn the model parameters e and h for a certain deployment site. We presents tools to gather temperature, interference, signal strength, and timing data at a given deployment location using off-the-shelf WSAN nodes. In some cases, especially for hardware platform parameters, these tools are run in specialized testbed infrastructures (such as the one designed in WP4 [5]) allowing us to carry out repeatable experiments in a controllable environment and to significantly reduce the experimentation time. We show examples of how to use these tools to gather datasets from which the the model parameters e and h are learnt, carefully describing each step.

Second, we derive bounding and aggregation methods for each model. This is a necessary step as the utility of a model is often low without model bounds. For instance knowing the average clock drift of one particular platform provides little use without worst and best case bounds to configure guard times. Aggregation is also fundamental for simplifying the use of models. Considering models produced by each mote individually would add an unnecessary complexity, and combining models captured by different nodes into a single model instance can drastically simplify their use.

This document provides a detailed descriptions of these two aspects, and details on the learning of model parameters for temperature, interference, signal strength, and timing data, respectively.

1 Introduction

In Deliverable D-1.1 [16] we have developed environmental and platform models that can be used to predict the performance of sensor networks and to automatically set an optimal protocol configuration. However, the parameters of these models may differ from one environment to another or between different platforms. Obtaining the correct parameter values to characterize a certain deployment environment or platform is a non-trivial task. For environments, human domain experts with sufficient experience and deep knowledge of the environment may be able to estimate those parameters, but this requires substantial effort and needs to be verified. For platforms, values in data-sheet may provide some insights but detail is often insufficient and have little consideration of environmental conditions.

This deliverable presents work towards devising algorithms and tools to learn these parameters in order to support domain experts. The idea is to install a cost-efficient measurement system in the target deployment environment prior to the actual deployment of the WSAN. These measurement systems consist of a small number of WSAN nodes that are installed for a short amount of time to sample the environmental and platform parameters of interest. The data collected can then be used to compute the model parameters from these observations. In some cases, particularly for platform parameters, the tools can be run in the lab under controlled conditions to reduce the time needed for data collection. Tools will be developed to look at these specific aspects: temperature, interference, RSSI, and system timing.

Not only is data collection important, but also model bounding and aggregation. The utility of a model is often low without model bounds, for instance knowing the average clock drift of one particular platform provides little use without worst and best case bounds to configure guard times. Aggregation is also important for simplifying the use of models. Considering models produced by each mote individually adds an unnecessary complexity and combining models captured by different nodes into a single model instance can drastically ease their use.

Chapter two presents a summary in addition to refinements of the models presented in deliverable D1.1 [16]. Chapter three presents a study on temperature showing how a tool can be constructed to collect deployment temperature data and how to instantiate the model using real-deployment data. Chapter four focuses on radio interferences examining how interference data can be collected and how the radio interference environmental model can be instantiated. Chapters five and sifx look at the platform aspects of received signal strength and timing respectively, presenting a data collection tool and an investigation into model instantiation, aggregation and bounding. The final chapter presents our conclusions for this deliverable.

2 Environmental and Platform Models

The performance of sensor networks is determined by two key components: the environment and the hardware platform. In turn, the behaviour of these two components (environment and platform) is dictated by a set of parameters. Formally, denoting $\mathcal{N}$ as the performance of a network, $\mathcal{E}$ as the behaviour of the environment and $\mathcal{H}$ as the behaviour of the hardware platform, the relationship among the network performance, the environment and the hardware platform is given by:

$$f: \quad \mathcal{E}(e), \mathcal{H}(h) \rightarrow \mathcal{N} \tag{2.1}$$

where f is the protocol model that we develop as part of WP2. In other words, the performance of the sensor network $\mathcal{N}$ is a function of the environment $\mathcal{E}$ and the hardware platform $\mathcal{H}$.

Our goal is twofold. First, we want to learn in a simple manner the model parameters e and h (modelling $\mathcal{E}$ and $\mathcal{H}$ was the focus of Deliverable D-1.1 [16]). Second, based on these parameters, we want to provide bounds on the performance of the network.

In the last four months (since D-1.1), we have focused mainly on temperature effects (as opposed to interference effects) for the same reason provided on D-1.1: scientific novelty. Studying the effect of interference is an active research area in the community, while temperature has received much lesser attention in spite of the (potentially) dramatic changes that it can cause on the network performance, as described in D-1.1 [16].

In this chapter we summarise the key findings from Deliverable D-1.1 (Environmental and Platform Models) and provide refinements which have been made to the models over the last four months.

2.1 Environmental Models

In D-1.1 [16] we have identified two environmental aspects that can have a significant influence on the performance of IoT systems and applications: temperature and radio interference. The following two subsections will present a recap of our initial findings presented in D-1.1, plus the progress made on top of them.

2.1.1 Temperature

Thermal properties. In deliverable D-1.1, we described the behaviour of the environment $\mathcal{E}$ in terms of four thermal properties (models): hotness, periodicity, change of rate, and maximum and minimum temperature range [16]. We now provide a brief summary of these properties.

Hotness. Given that temperature has a detrimental effect on the transmission range and timing accuracy of a node, it is important to characterise the *hotness* of a node. Given the

probability mass function of temperature $p_i(t)$, the hotness of a node H_i can be defined by the expectation:

$$H_i = \sum_t t * p_i(t) \tag{2.2}$$

Periodicity. Considering that temperature profiles usually follow periodic patterns[1], it is important to identify the periodicity of these patterns to exploit the times of the day where the temperature is the lowest (because the performance of the network is better at these "colder" times). Denoting f_i and g_i as the time series of the temperature observed by node i at two different days, we use cross correlation functions to quantify the periodicity of a node P_i:

$$(f_i \star g_i)[n] = \sum_{m=-\infty}^{\infty} f_i[m] g_i[n+m] \tag{2.3}$$

$$P_i = \max_n (f_i \star g_i)[n] \tag{2.4}$$

Change of Rate. One of the aims of our consortium is to design protocols that can adapt to environmental changes. To design the appropriate mechanism we need to quantify how fast the environment changes, because the network properties will change at the same rate. To capture the maximum rate of change on a node i, we identify the steepest slope of the temperature series f_i:

$$R_i = \max_t (f_i(t + \Delta t) - f_i(t)) \tag{2.5}$$

Max/min Temperature Range. Arguably one of the most important characteristics of the environment to be modelled are the maximum and minimum temperatures recorded on each node i, because they bound the performance of the network. Given a trace f_i recorded at a node i, these properties are easily obtained:

$$(\max_i^t, \min_i^t) = (\max\{f_i\}, \min\{f_i\}) \tag{2.6}$$

The progress made for this deliverable, in the realm of temperature effects, can be described as follows. In deliverable D-1.1, we modelled the relevant properties of the nodes (hotness, periodicity, rate of change and max/min range) *given* the behaviour of the environment $\mathcal{E}$ (temperature traces). In this deliverable, we investigate the model parameters e required to *(re)create* the environment $\mathcal{E}$ itself.

Identifying the model parameters. Using thermodynamic equations, we derived a model suitable to create temperature profiles for nodes. We focus on outdoor deployments where IR (infrared) radiation from the sun and air temperature are the most significant factors. In essence, objects heat up by absorbing solar radiation and cool down by constantly releasing

[1] The most obvious periodic pattern is the thermal change between day and night. Other less obvious changes are seasonal changes of temperature or changes caused by obstacles such as buildings or trees that cause shades at specific times of the day.

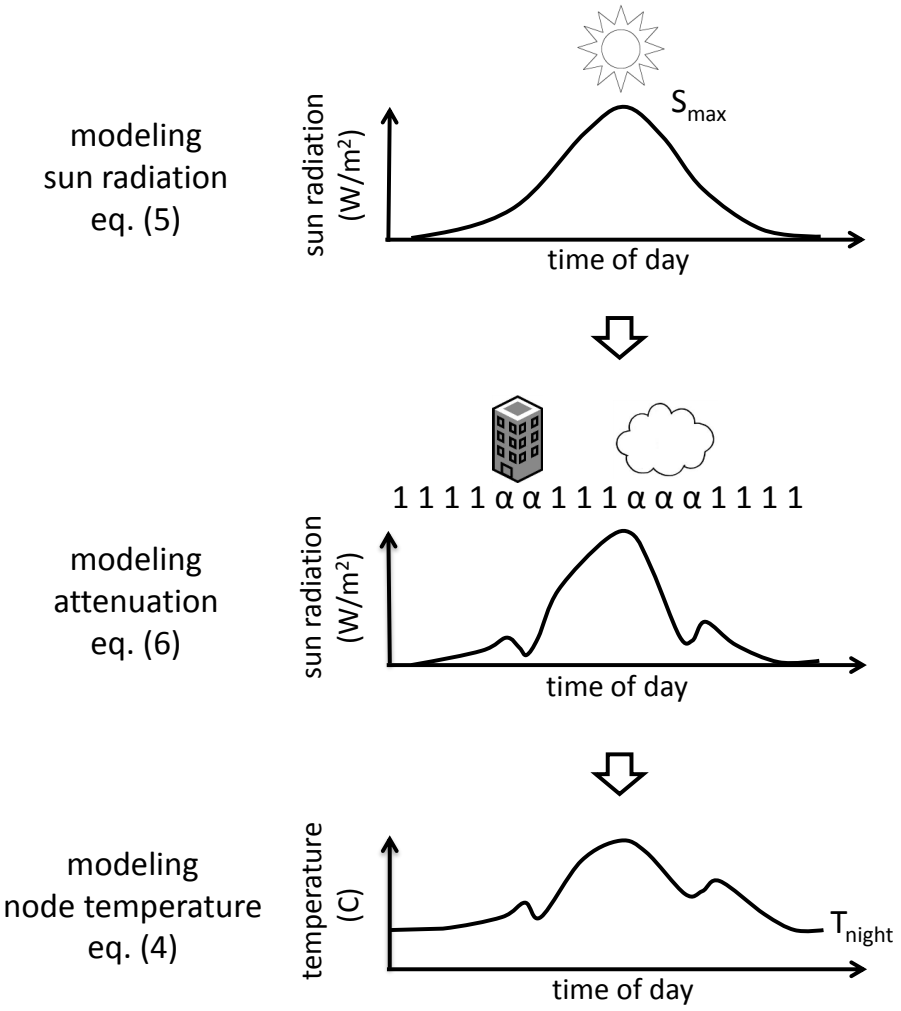


Figure 2.1: Model-based temperature profile generation.

energy to their surrounding. The balance between these processes determines the object temperature.

Before describing the theory behind the model, we provide a pictorial representation in Fig. 2.1. The model has three basic steps. First, we model the impact of sun radiation. This is done by assuming a clear sky, where the object absorbs the maximum possible IR radiation hitting its surface (top diagram in Fig. 2.1). Second, we model the impact of events blocking sun radiation, such as clouds and buildings. These events decrease the temperature of the object (middle diagram in Fig. 2.1). Finally, we put these two models together to recreate the temperature profile of a node, based on a simple set of model parameters e (bottom diagram in Fig. 2.1).

Energy absorption and dissipation. An object that is exposed to the sun, absorbs energy

according to:

$$E_{in} = S\alpha A\Delta t, \tag{2.7}$$

where S is the solar radiation, α is the attenuation of the solar radiation, A is the exposed area of the object and Δt is the amount of time in which the object was exposed to the solar radiation. On the other hand, objects release energy according to:

$$E_{out} = sT^4 A\Delta t, \tag{2.8}$$

where s is the Boltzmann constant and T is the temperature of the object in Kelvin.

Energy balance. Considering the energy absorption and energy dissipation of an object, its change of temperature ΔT is determined by the heat energy equation: $H = C_p m\Delta T = E_{in} - E_{out}$, where C_p is the specific heat of the object and m its mass. The temperature of an object cannot be less than air temperature at any given time t (T_t^{air}). Hence, at time $t + \Delta t$, the object temperature is given by:

$$T_{t+\Delta t} = \min\{\ T_t + \frac{(S_t\alpha_t - sT^4)}{C_p m}A\Delta t,\quad T_t^{air}\ \} \tag{2.9}$$

Considering a standard mote with parameters $m = 50$ grams, $C_p = 0.5\ \frac{J}{gC}$, $A = 20$ cm^2; the model only requires the sun radiation S_t, the air temperature T_t^{air} and the attenuation α_t ($0 \leq \alpha_t \leq 1$).

Sun radiation and cloud obstruction. In the absence of any obstructions, the sun radiation throughout the day can be modelled by a Gaussian-like shape [11]:

$$\begin{aligned} S_t &= \frac{S_{max}}{\max\{\mathcal{N}(0,\sigma)\}}\frac{1}{\sqrt{2\pi}\sigma}\exp^{-(t-\delta)^2/2\sigma^2} \\ &= S_{max}\exp^{-(t-\delta)^2/2\sigma^2}, 0 \leq t \leq 2\delta \end{aligned} \tag{2.10}$$

where S_{max} is the maximum sun radiation during the day, and $t = 0$ and $t = 2\delta$ represent the 00 hrs and the 24 hrs. The number of hours with sunlight (length of day) can be fine-tuned with σ and δ. To further simplify Eq. 2.9, instead of considering the air temperature throughout the day (T_t^{air}), we use only the minimum temperature in the day (i.e., the night temperature T_{min}).

Hence, the only information that we need to model is the maximum radiation and minimum *air* temperature (datasets about a given location are typically available from satellites and meteorological stations).

Few locations, however, receive constant sun radiation throughout the day. In most scenarios, clouds block the sun radiation and cause sudden variations of temperature. The length of clouds and the length of the clear sky between clouds are known to have exponential distributions $\lambda\exp^{-\lambda x}, x \geq 0$, with $\frac{1}{\lambda}$ representing the average cloud (or inter-cloud) length [13]. Denoting $\overrightarrow{\alpha}$ as an *attenuation vector* where all elements are α and its length is given by the exponentially random length of a cloud. And denoting $\overrightarrow{1}$ as a *clear-sky vector* where all elements are 1 (i.e., $\alpha = 1$) and its length is equal to the random length of an inter-cloud period; the variable α_t in Eq. 2.9 is the t^{th} element of the vector:

$$\overrightarrow{v} = \{\overrightarrow{\alpha_1}, \overrightarrow{1_1}, \ldots, \overrightarrow{\alpha_i}, \overrightarrow{1_i}, \ldots\}. \tag{2.11}$$

At each t in Eq. 2.9, the t^{th} element is used to capture the amount of sun radiation attenuated during the respective period Δt. The shade of events that are specific to the scenario of interest (trees, buildings, etc), can be included in $\overrightarrow{v}$ by inserting attenuation elements (α) in the vector.

The model allows the user to test a wide range of scenarios. The user can test the worst-case temperature with clear skies, generate shades of any length at any time (to test temperature gradients), and generate random instances for each node by varying the model parameters. The model can be easily coded using any programming or scripting language, we use Matlab. To compute a temperature value at time t, Eq. 2.9 is evaluated for the respective value of Δt.

2.1.2 Radio Interference

In deliverable D-1.1, we have described models for capturing radio interference. In particular we have illustrated how the interference can be modelled using the Cumulative Distribution Function (CDF) of idle and busy periods. We have shown that this approach is suitable for low-power wireless networks as knowledge on distributions of idle and busy periods can be exploited to design interference-aware protocols. This approach has also the key advantage of being usable with resource-constrained wireless sensor nodes: as we will discuss in Section 4.1, these distributions can be easily derived by performing RSSI sampling on off-the-shelf motes. We now discuss the importance of knowing these CDF distributions (for a longer discussion, please refer to D-1.1 [16]) and we describe how these distributions can be modelled.

The importance of the CDF of idle and busy periods. The success of a packet transmission is strongly correlated to the presence of radio interference in the environment. Transmissions can indeed only succeed if they occur when no other device (generating noise at a higher power) is active. To construct transmission schedules and configure protocol primitives such as back-off timing it is necessary to have knowledge about the duration of busy and idle periods. For example, a protocol would use short CCA back-off times and short payloads in an environment where interference occurs in short burst with short idle periods. Long CCA back-off times might instead be advisable in an environment where interference occurs in long bursts with large idle periods in order to minimise the radio activity.

Denoting $p_i(i)$ as the Probability Density Function (PDF) of the idle periods formed by the interference pattern, a protocol could for example select the optimal payload length by computing the probability of encountering an idle period of length i during transmission:

$$p_{idle\_period}(i) = \frac{ip_i(i)}{\sum_{i=1}^{\infty} ip_i(i)} \tag{2.12}$$

Similarly, denoting $p_b(i)$ as the PDF of the busy periods formed by the interference pattern, a protocol could for example select the optimal back-off time for clear channel assessment by knowing the probability of selecting a busy period of length i during transmission:

$$p_{busy\_period}(i) = \frac{ip_b(i)}{\sum_{i=1}^{\infty} ip_b(i)} \tag{2.13}$$

The Probability Density Functions (PDF) $p_i(i)$ and $p_b(i)$ can also be expressed as Cumulative Distribution Functions (CDF) $P_i(i)$ and $P_b(i)$, with $P_i(i) = \int_{-\infty}^{i} p_i(x)\,\mathrm{d}x$ and $P_b(i) = \int_{-\infty}^{i} p_b(x)\,\mathrm{d}x$. We use the CDF for our modeling approach discussed next.

Modeling the CDF of idle and busy periods. The distribution of idle and busy periods is dependent on the type of devices that are generating interference. Some devices, such as microwave ovens, generate periodic interference patterns with relatively long idle periods, while others, such as Wi-Fi stations, generate interference patterns with short idle periods of a highly variable length. When several interfering sources are present, interference occurs continuously and independently at a constant average rate. In these cases it is reasonable to model the CDF as an exponential distribution. Thus we use, when possible, the following models to capture idle and busy CDF:

$$P_i(i) = \begin{cases} 1 - e^{-\lambda \cdot i} & i \geq 0 \\ 0 & i < 0 \end{cases}$$
$$P_b(i) = \begin{cases} 1 - e^{-\lambda \cdot i} & i \geq 0 \\ 0 & i < 0 \end{cases} \tag{2.14}$$

However, there are cases in which the use of an exponential distribution is not appropriate. This applies for example to the interference generated by microwave ovens, as we will describe in detail in Section 4.1). In this case a generic CDF model cannot be used and we instead directly measure and capture the CDF in an empirical way.

2.2 Platform Models

In deliverable D-1.1 we identified that the Receiver Signal Strength Indicator (RSSI) and the timing of the system can be significantly affected by temperature changes. The following two subsections will summarize these platform aspects.

2.2.1 Receiver Signal Strength Indicator

As we have shown in [6], a raise in temperature can drastically reduce the Signal-to-Noise Ratio (SNR). A lower SNR means a lower link quality and a shorter radio link: a temperature change of 50°C is enough to reduce the SNR by more than 6 dB, which can change the packet reception rate (PRR) of a link from 100% to 0%. This may translate not only into a lower throughput, but also into a higher delay or even into a network partitioning.

Impact of Temperature on SNR. In deliverable D-1.1, we modelled the effect of temperature on SNR. Denoting PL as the path loss between a transmitter-receiver pair, P_t as the transmission power, P_r as the received power, and P_n as the noise floor at the receiver, the SNR is known to be:

$$\begin{aligned} SNR(dB) &= P_t - PL - P_n \\ &= (P_t - P_n) - (P_t - P_r) \end{aligned} \tag{2.15}$$

We modelled three main effects of increasing temperature on the signal strength of radio transmissions in Eq. 2.15. These effects where shown in deliverable D-1.1 with empirical measurements and they are (i) decrease in the transmitted power, (ii) decrease in the received power, and (iii) decrease in the noise floor.

A First-Order Model [6]. Denoting α, β, γ as constants with units dB/K, and T_t, T_r as the temperature in Kelvin of transmitter and receiver, the effect of temperature on SNR can be defined as:

$$\begin{aligned} SNR &= (P_t - \alpha\Delta T_t) - (PL + \beta\Delta T_r) \\ &\quad -(P_n - \gamma\Delta T_r + 10\log_{10}(1 + \tfrac{\Delta T_r}{T_r})) \\ &= P_t - PL - P_n - \alpha\Delta T_t \\ &\quad -(\beta - \gamma)\Delta T_r - 10\log_{10}(1 + \tfrac{\Delta T_r}{T_r}) \end{aligned} \tag{2.16}$$

The proportional relation between ΔT and the constants α (effect on transmitted power), β (effect on received power) and γ (effect on noise floor) is based on the empirical observations made in deliverable D-1.1. The term $10\log_{10}(1 + \frac{\Delta T_r}{T_r})$ is derived analytically from the well-known thermal equation. Changes in temperature have a higher impact on the transmitted and received powers (linear relation of α and β), than on the thermal noise (logarithmic relation). And, to some extent it is counter-intuitive that a higher temperature decreases the noise floor (negative sign of γ). This effect was also observed by Bannister, and he hypothesizes that it is due to the losses in the signal amplifier [2] [1]. That is, a higher temperature not only reduces the gain of the signal but also the gain of the noise, and hence, the received signal strength (RSSI) is lower for both.

The accuracy of our model depends on identifying the right values for α, β and γ. In our case, these parameters are given by the slopes of the linear trends observed in our empirical results. These parameters are platform dependant, and hence require a systematic and fine-grained evaluation.

2.2.2 System Timing

In Deliverable D-1.1 we derived two models to represent the temperature effects on timing, the first looked at RC-Clock type whilst the second on RT-Clock type.

RC-Clock Model. An RC-clock uses resistive and capacitive components whose profiles alter as temperature changes. These profile variations are linear and can be modelled using Equation 2.17. The circuit designed to have a frequency f_0 close to the target frequency at room temperature T_0. A_1 is a temperature coefficient with unit $\%/^\circ\mathrm{C}$.

$$f(T)/f_0 = A_1.(T - T_0) \tag{2.17}$$

RT-Clock Model. The outlined behaviour of different crystal cuts can be modelled using Equation 2.18. f_0 is the frequency of the RT-clock at room temperature T_0. $f(T)$ is the RT-clock frequency at temperature T. A_1 is a temperature coefficient with unit $\%/^\circ\mathrm{C}$. A_2 and A_3 are temperature coefficients with unit $\%/^\circ\mathrm{C}^2$ and $\%/^\circ\mathrm{C}^3$.

$$f(T)/f_0 = A_1.(T - T_0) + A_2.(T - T_0)^2 + A_3.(T - T_0)^3 \tag{2.18}$$

3 Environmental: Temperature

Temperature has been recognized as one of the key environmental aspects that can significantly affect IoT systems and applications. In this chapter we present tools that can be used to measure temperature at a deployment site at each node location. This temperature dataset will augment available meteorological information when instantiating the temperature model. This model will also be used (together with platform models) to aid offline protocol selection in WP3.

3.1 Collecting Temperature Data with Motes

Measuring temperature on motes is a simple exercise and wireless sensor nodes have been used to measure temperature for years. Most off-the-shelf wireless sensor nodes have embedded temperature reading capabilities so that one can periodically read temperature. One needs however to make sure that the temperature sensors used have been previously calibrated and that they have a reasonable accuracy (in the order of $\approx 0.5°C$). Sufficient sample points must also be recorded such that the variations in temperature can be accurately observed. Temperature typically changes slowly, hence measurements points taken every few seconds give a sufficiently high accuracy.

We develop an application that reads temperature using the Contiki operating system [8]. Every few seconds, the application samples temperature and outputs the recorded value in one out of three possible ways. Temperature can be outputted via RS232 to a PC, but this would require that every mote in the network is actually connected to a central machine. A second approach consists in storing the temperature readings into flash memory so that they can be read out during reprogramming. A third option is to transmit temperature to a central sink using wireless communication.

Instead of developing an application just to record temperature, it is advisable to collect temperature while collecting as well other environmental or platform datasets, such as timing or RSSI. Most of the applications evaluated within RELYonIT, indeed, measure temperature whilst measuring application specific data.

3.2 Instantiating the Temperature Model

In this section, the temperature model described in Section 2.1.1 will be instantiated. The focus of our model will be on bounding the temperature pattern for all nodes.

Motivation to obtain bounds. Figure 3.1 shows an example of temperature profiles collected using 16 off-the-shelf wireless sensor nodes deployed in Uppsala, Sweden [14] (each blue curve represents the temperature profile of a different node).

We can observe that even if nodes are in close proximity of each other (a few meters in this case), their temperature profiles can be vastly different. Figure 3.2 depicts the temperature

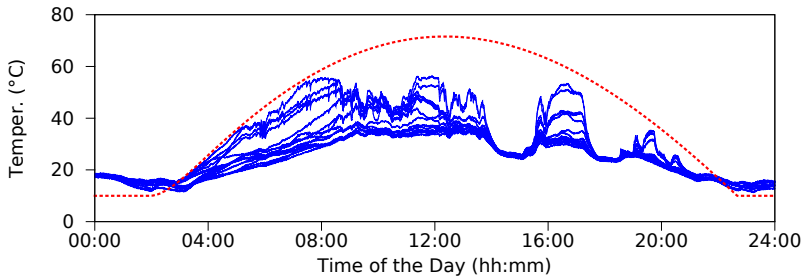


Figure 3.1: Temperature profiles over the course of a day of 16 nodes deployed in an outdoor setting (blue curves), and maximum temperature profile obtained with the model presented in Sect. 2.1.1 (red dashed curve).

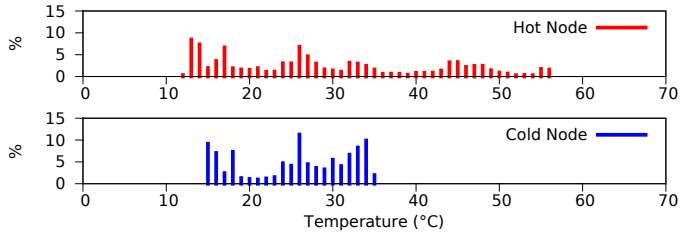


Figure 3.2: The temperature profile of nodes can be highly different even if nodes are in proximity of each other. Our model allows us to bound the temperature pattern over *all* nodes.

density of the "hottest" and "coldest" nodes in this deployment, and we can observe that the range of temperature varies significantly.

Considering that high temperatures can have a detrimental effect on network operation, in several instances it is important to consider the worst case scenario, i.e., an upper bound of the highest possible temperature. For example, if we only deploy a few test nodes that, by chance, happen to be located on the "coldest" areas of the deployment, protocols can fail if parts of the full-blown deployment are exposed to higher temperatures.

Obtaining the bounds. To avoid these undesirable events, we can use our temperature model. The latter only requires two parameters to obtain a *clear sky* temperature profile: (i) the maximum sun radiation which usually occurs at noon, and (ii) the minimum temperature that occurs during the night (which usually remains stable, as we can see in Figure 3.1 between 23:00 and 03:00).

In the case of the example shown in Figures 3.1 and 3.2, the traces were collected in Uppsala during August. During this period of the year, the maximum radiation is 800 W/m^2, whereas the minimum temperature during the night is 10°C. Inserting these values on Equations 2.9 and 2.10, respectively, we obtain the dashed red-curve presented in Figure 3.1. As we can observe, the model bounds the temperature patterns of *all* nodes in the network. Without the model, in this particular deployment, the error on the maximum expected temperature could

have ranged range from ≈40°C, if the "coldest" node would have been used as reference, to ≈20°C, if the "hottest" node would have been selected instead. As described in Deliverable D-1.1 [16], such differences in temperature can affect significantly the operation of a network.

Trace-based modelling. In the event that temperature traces are available, but no meteorological information is accessible (i.e., no sun radiation), the model could still be used to estimate the Gaussian curve bounding the curves under consideration. For example, given four temperature traces among the 16 presented in Figure 3.1, we could search (iteratively) the parameters of the model that bound the traces in the most accurate manner. This basic model could be used to generate other random traces. It is important to remark that this method is not ideal, because it would likely create an artificially lower bound. For example, the bound obtained if the four hottest traces are chosen would be different from the bound obtained if the four coldest traces are selected. Therefore, this method should be used as a last resort.

4 Environmental: Radio Interference

4.1 Measuring Interference

To get an overview of spectrum usage and to select an optimal radio channel, one typically carries out detailed interference measurement prior deployment. However, as interference pattern may change over time, one needs to verify at runtime that the environment did not change significantly (runtime assurance). It may be hence necessary to repeat interference measurements periodically within a deployment. To achieve this goal, it would be optimal to let sensor nodes carry out the interference measurements (this avoids the need for extra-hardware). There are two approaches to measure interference: one is to rely on passive sampling, whilst the second one exploits active probing [7]. For this deliverable, we exploit passive sampling. In the next paragraphs we discuss how to let an off-the-shelf sensor node carry out interference measurements with a sufficiently high sampling rate. Thereafter we illustrate how the collected datasets are used to instantiate the interference model presented in Section 2.1.2. Particular attention is given to capturing Wi-Fi traffic, as we consider this type of interference to be the most common interfering source in typical deployments.

4.1.1 Measurement Capabilities of Off-The-Shelf Sensor Nodes

Most off-the-shelf radios (e.g., the Texas Instruments CC2420 or CC1020 transceivers) provide hardware link quality metrics such as the Received Signal Strength Indicator (RSSI) and the Link Quality Indicator (LQI). These metrics provide information on signal strength and chip error rate upon packet reception, but are not designed for precise interference measurements. These metrics provide information that is limited to the instant at which a packet has been received. However, for proper interference measurements such single measurement points in time are not useful and a method is required to provide continuous measurements over long time periods. Fortunately, modern transceivers provide the capability to read RSSI values also in absence of packet receptions. These "RSSI noise floor" measurements taken in absence of packet transmissions can be used to quantify the level of interference at a given node. To get an accurate understanding of the ongoing interference, however, RSSI values need to be sampled at a high rate.

4.1.2 Required Sampling Rates

To detect short transmission periods such as the ones generated by Wi-Fi devices we need to achieve high sampling rates. This is a challanging task when using resource-constrained wireless sensor nodes. Hauer et al. [9, 10] have shown that 50-60 kHz is a frequency 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. Hence, we follow the approach used in [3, 4], and perform a

high-speed sampling of the RSSI register on Maxfor MTM-CM5000MSP[1] by boosting the CPU speed, optimizing the SPI operations, as well as by buffering and compressing the RSSI noise floor readings using Run-Length Encoding (RLE). Although the achievable 50 kHz 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 150 Mbit/s for 802.11b/g/n standards, respectively). However, since most Wi-Fi frames are data frames and typically contain higher layer headers, and since the IEEE 802.11n standard uses large PDUs to reduce preamble overhead [15], we can still capture a significant fraction of the Wi-Fi traffic [3] using this approach.

4.1.3 Sampling Error Detection

When pushing the performance of common nodes to the edge, we have to make sure to obtain meaningful RSSI readings. Boano et al. [3] have highlighted that RSSI noise floor readings captured at high sampling rates may return values that are significantly below the sensitivity threshold of the radio in three specific scenarios, namely: (i) when a narrow unmodulated carrier is transmitted, (ii) when microwave ovens are switched on, and (iii) in the presence of Bluetooth transmissions. The problem occurs due to the saturation of the Intermediate Frequency (IF) amplifier chain. They have indeed observed that maximum gain is used in the Variable Gain Amplifier (VGA) when incorrect RSSI readings are obtained.

To linearise the radio response for an arbitrary noise signal and hence avoid wrong RSSI readings, we activate the peak detectors in-between the amplifier stages so that their output is used by the AGC algorithm to compute the required gain. The latter is attained with VGA stages and the system switches in and out fixed gain stages as needed. In the CC2420, the peak detectors are controlled by the *AGCTST1* register, and can be configured as follows:

```
unsigned temp;
CC2420_READ_REG(CC2420_AGCTST1, temp);
CC2420_WRITE_REG(CC2420_AGCTST1,
(temp + (1 << 8) + (1 << 13)));
```

4.1.4 Lab and Deployment Based Interference Measurement

To derive the CDF 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 $R = 100.000.000$ samples). 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 μs.

We define a 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) and count how many consecutive RSSI readings fall above or below such threshold. We store the number

[1] We use Maxfor MTM-CM5000MSP nodes for our interference experiments. This node type is a typical sensor node with limitations and capabilities found on most off-the-shelf wireless sensor nodes.

of consecutive RSSI readings in which a channel remained idle or busy into two arrays $A_{idle}[i]$ and $A_{busy}[i]$, and as soon as the current channel state (idle, busy) differs from the previous one, we increment the corresponding array field. Because of the limited memory constraints of the nodes, we truncate the maximum duration of an idle or busy period to 125 ms.

```
0, 50
50, 175
175, 350
350, 500
500, 650
650, 800
800, 1000
1000, 1500
1500, 2250
2250, 3000
3000, 4000
4000, 5500
5500, 7000
7000, 8500
8500, 10000
10000, 12000
12000, 14000
14000, 17000
17000, 20000
20000, 26000
26000, 32000
32000, 50000
50000, 75000
75000, 125000
```

Listing 4.1: Length of timeslots.

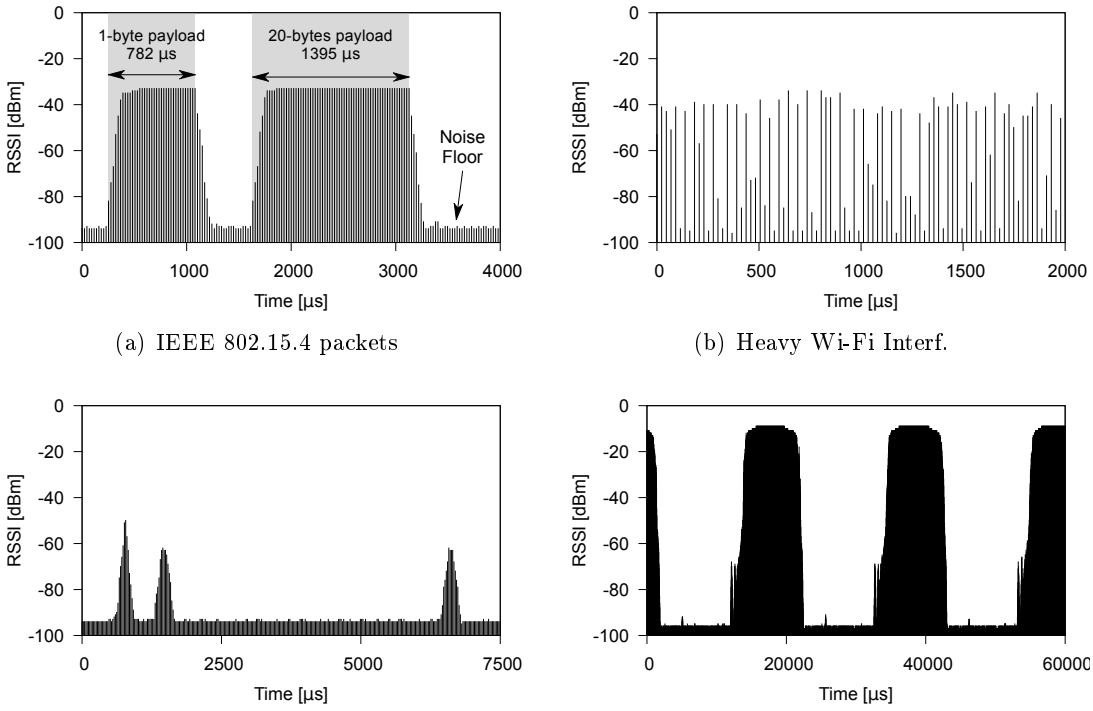
In our implementation, the first slot contains the idle or busy periods of duration between 0 and 50 μs, the second slot the idle or busy periods of duration between 50 and 175 μs, as shown in Listing 4.1. Also in this case, we truncate the maximum duration of an idle or busy period to 125 ms. The discretization of time is necessary due to the constrained memory resources of the sensor nodes. As we need to achieve a fast sampling rate, the use of external flash memory or USB backchannel is not a suitable option, and in order to quickly store the number of consecutive samples in the arrays, a lookup table is used.

4.2 Interference Types

When neither interference nor IEEE 802.15.4 communications are present, the measurement of the RSSI noise floor typically returns values in the proximity of the radio sensitivity threshold (e.g., in the range $[-100, -94]$ dBm for the CC2420 radio).

In the presence of IEEE 802.15.4 communications, the fast RSSI sampling returns a stable value corresponding to the strength and the length of the transmitted packet (Figure 4.1(a)). As packets have a constrained maximum payload size of 127 bytes according to the 802.15.4 PHY standard, a packet transmission at 250 Kbit/sec would not last more than 4.3 ms.

When other devices operating in the same frequency band of wireless sensor networks are active, bursts of interference signals (*busy periods*) alternate with instants in which the channel is clear (*idle periods*). The strength of the interference signals and the duration of idle and busy periods depend on the interfering source and on the specific context. For example, the interference patterns generated by Wi-Fi transmissions depend on the number of active users and their activities, as well as on the traffic conditions in the backbone. Figure 4.1(b) shows the outcome of fast RSSI sampling in the presence of heavy Wi-Fi interference (caused by a file



(a) IEEE 802.15.4 packets

(b) Heavy Wi-Fi Interf.

(c) Bluetooth interf.

(d) Microwave oven interf.

Figure 4.1: RSSI values measured using off-the-shelf wireless sensor nodes operating in the 2.4 GHz ISM band. Please notice the different scale of the x-axis [4].

transfer): it is indeed possible to identify RSSI values matching the radio sensitivity threshold between consecutive Wi-Fi transmissions.

Figure 4.1(c) shows an example of interference generated by Bluetooth. The latter uses an Adaptive Frequency Hopping mechanism to combat interference, and hops among 1-MHz channels around 1600 times/sec., hence it remains in a channel for at most 625 μs. Since Bluetooth channels are more narrow than the ones defined by the 802.15.4 standard, it may happen that communication in multiple adjacent Bluetooth channels affects a single 802.15.4 channel.

Figure 4.1(d) shows an example of the interference pattern caused by microwave ovens: high-power noise ($\approx$ 60 dBm) is emitted in the 2.4 GHz frequency band in a very periodic fashion. The period mostly depends on the power grid frequency, but can also slightly vary depending on the oven model. Works in the literature report a power cycle of roughly 20 ms (at 50 Hz) or 16 ms (at 60 Hz) with an active period of at most 50% of the power cycle [3, 12].

4.3 Model Instantiation

The interplay between idle periods and busy periods is complex because of the particular patterns of each interfering source. Some devices, such as microwave ovens, generate periodic interference patterns with relatively long idle periods (Figure 4.1(d)), while others, such as Wi-Fi stations, generate interference patterns with short idle periods of a highly variable length (Figure 4.1(b)).

In order to quantify precisely the distribution of idle and busy periods, we use the previously described tool to capture data used to determine the cumulative distribution function (CDF) of idle and busy periods in a given IEEE 802.15.4 channel. We continuously scan the radio channel and detect whenever the activity in the channel is above a configurable threshold R_{thr}. A channel is defined as busy if the RSSI is higher or equal than R_{thr} and idle otherwise. We then map the amount of consecutive RSSI samplings spent above or below R_{thr} to the time in which the medium remained busy or idle, respectively. This operation is relatively easy as one RSSI sampling always takes the same amount of time (we have used lookup tables to make sure the execution time remains approximatively constant).

Figure 4.2 shows the cumulative distribution function 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. Figure 4.3 shows the cumulative distribution function of idle and busy periods measured by a Maxfor MTM-CM5000MSP node in the presence of an operating microwave oven.

In the scenario shown in Figure 4.2, the probability of having an idle period longer than 2 ms is smaller than 5%. This implies that the chances that a burst of packet (e.g., a message-based handshake) successfully completes within an idle period are rather low. In order to escape interference, one would need to use short messages and send them as close as possible to each other, in order to increase the chances of fitting them into an idle period.

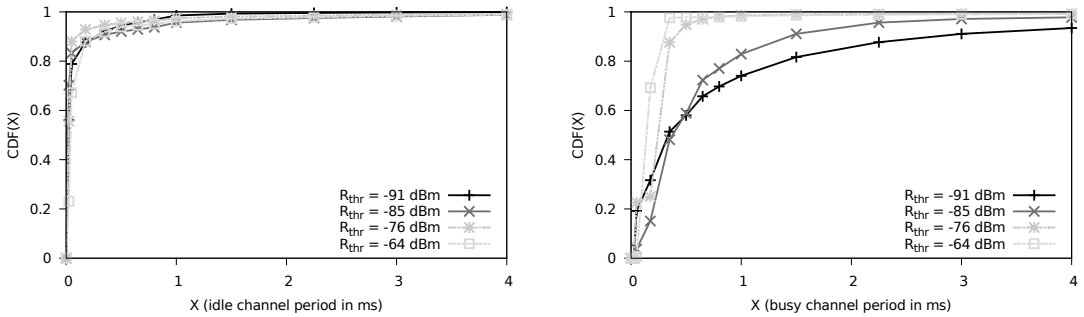
For common interference sources such as Wi-Fi devices, the CDF distribution follows an exponential distribution. For these cases, we instantiate the model described in Section 2.1.2, as described next.

4.3.1 Exponential Distribution of Idle and Busy Periods

We use the CDF shown in Figure 4.1(b) to demonstrate how to instantiate the model given by Equation 2.14. We fit the data points to the model given in Equation 2.14 to obtain the CDF parameter λ for idle and busy periods. The results for different threshold values are shown in Table 4.1. For a given threshold value R_{thr}, we show the outcome of the fitting and the residuals in Figure 4.5 (busy periods) and Figure 4.4 (idle periods). Table 4.2 shows the residual sum of squares (RSS) of CDF data and exponential model (the smaller the values, the better the fit). As it can be seen, although the residuals are biased, they are generally not very high, which suggests that the exponential distribution is suitable to model this type of environments.

4.3.2 Bounds

The interference patterns observed by a node will fluctuate over time and for this reason it might be beneficial to assume a more conservative CDF for busy and idle periods other than



(a) Idle periods (b) Busy periods

Figure 4.2: 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 [4].

	$R_{thr} = -91$	$R_{thr} = -85$	$R_{thr} = -76$	$R_{thr} = -64$	$R_{thr} = -46$
Idle periods	31.680	43.693	35.870	16.411	10.612
Busy periods	1.608	1.739	3.901	5.560	6.837

Table 4.1: λ values for the instantiated model of Idle and Busy CDF.

	$R_{thr} = -91$	$R_{thr} = -85$	$R_{thr} = -76$	$R_{thr} = -64$	$R_{thr} = -46$
Idle periods	0.027	0.047	0.018	0.046	0.063
Busy periods	0.060	0.021	0.100	0.094	0.177

Table 4.2: Residual sum of squares (RSS).

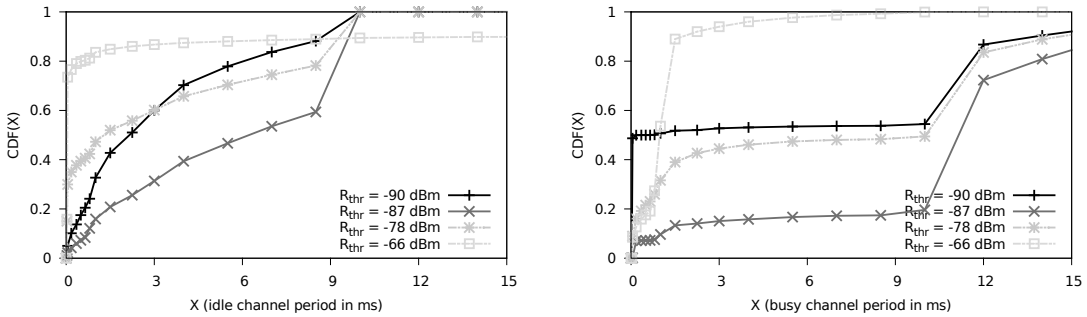
the one obtained by fitting the model to the measured data.

CDF bounds can be described by adjusting λ obtained via curve fitting. For example, one could simply multiply λ with a fixed value δ_u (upper bound) or δ_l (lower bound) to obtain bounds. However, in this case it is not clear what a good value for δ_u or δ_l would be.

To determine reasonable values for δ_u or δ_l we use the following approach. We increase δ_u starting at $\delta_u = 1$ until P percent of data points are below the bounding curve described as $1 - e^{-\delta_u \cdot \lambda \cdot i}$. We also allow to use a limit in terms of i up to which this bounding condition is evaluated. The lower bound curve determined by δ_l can be determined similarly. Obviously, these bounds can be determined for idle and busy CDF.

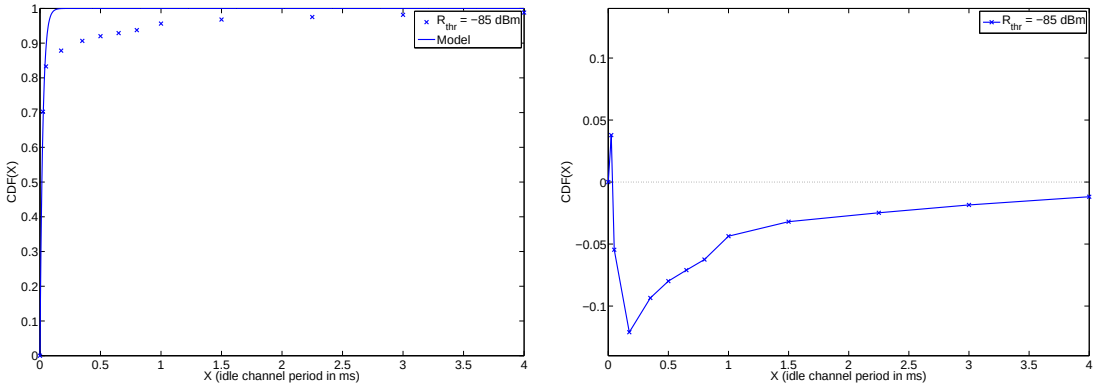
Selecting a high value P ensures that also extreme values are included within the bounds. Thus, it is ensured that extreme (but likely) patterns are covered by the bounds. Simply choosing a value for δ_u or δ_l without applying the described mehod would also be possible but in this case it might be that an unnecessary high and thus conservative bound is selected.

Figure 4.6 shows the bounds for idle and busy periods for the example described in the previous section (See Figure 4.4 and Figure 4.5). Depending on the application scenario either



(a) Idle periods (b) Busy periods

Figure 4.3: Cumulative distribution function (CDF) of idle and busy periods measured by a Maxfor MTM-CM5000MSP node in the presence of an operating microwave oven.



(a) Model fitting (b) Residuals

Figure 4.4: Model fitting for the idle period CDF with $R_{thr} = -85dBm$ and residuals.

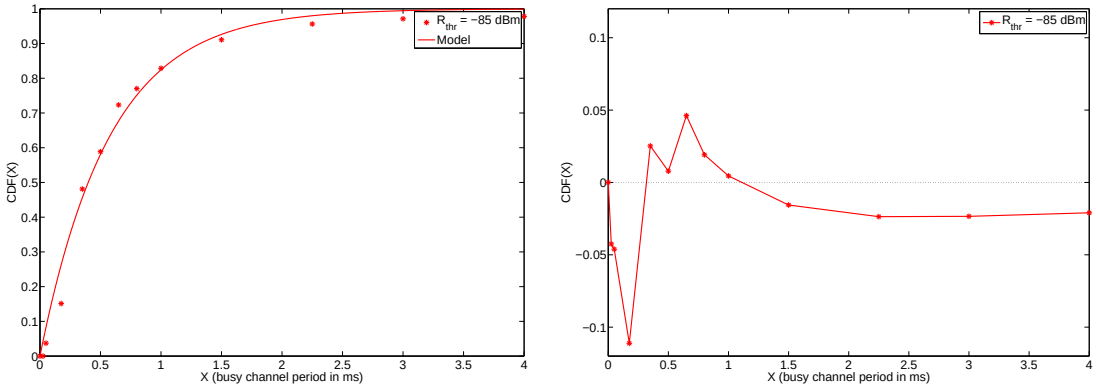
upper or lower bounds may be of importance. Here $P = 80\%$ was used, which results in $\delta_u = 1.06$ or $\delta_l = 0.535$ for the busy period CDF and $\delta_u = 1$ or $\delta_l = 0.035$ for the idle CDF.

Using bounds is particularly useful when aggregating interference measurements from different measurement locations to instantiate an interference model for a particular network region. This approach is described in the next subsection.

4.3.3 Aggregation

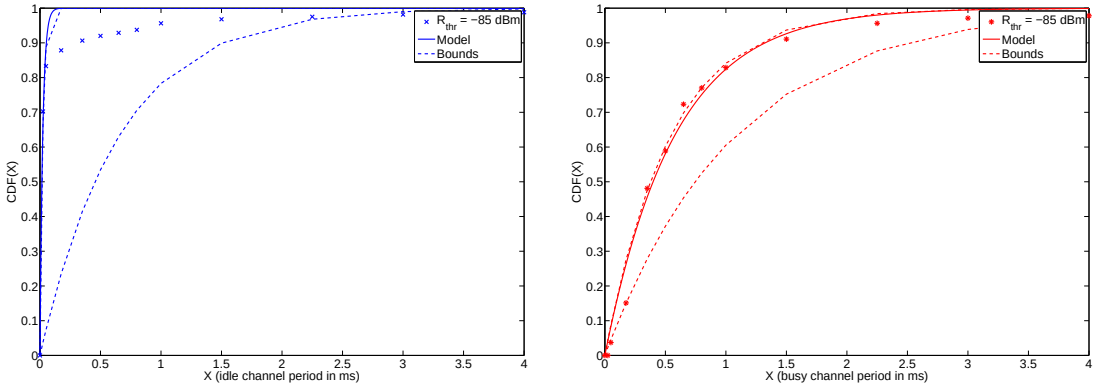
In most situations an interference model for multiple locations is required (a number of nodes are deployed). Thus, it would be beneficial if a single model instantiation can be used to cover a number of locations (i.e., nodes in the deployment). If interference patterns are similar at a set of locations, it is possible to provide an aggregated model.

Figure 4.8(a) shows an experimental setup in a lab environment. Figure 4.8(b) shows the



(a) Model fitting (b) Residuals

Figure 4.5: Model fitting for the busy period CDF with $R_{thr} = -85dBm$ and residuals.



(a) Model fitting (b) Residuals

Figure 4.6: Bounds for idle ($\delta_u = 1$, $\delta_l = 0.035$) and busy ($\delta_u = 1.06$, $\delta_l = 0.535$) periods with $P = 80\%$ ($R_{thr} = -85dBm$).

measured busy periods. Clearly, it can be seen that the measured interference for some nodes is similar (similar interference patterns are coded using the same color). Now it is possible to use all measurement data from all nodes (or some nodes) within a group of similar interference patterns to fit the interference model.

In a first step we use data from nodes 202, 207, 209, and 214 to instantiate the interference model for busy periods. The resulting fit for this data is shown in red in Figure 4.7. In addition, the red dashed lines show the upper and lower bound (with $\delta_u = 1.345$, $\delta_l = 0.615$ for $P = 80\%$) for this aggregated model instantiation.

We now reduce the number of measurement points considered for model instantiation. We only use nodes 214 and 207 to instantiate the model (which is shown as green curve in Figure 4.7). As we can observe, there is not much benefit in considering the additional two nodes

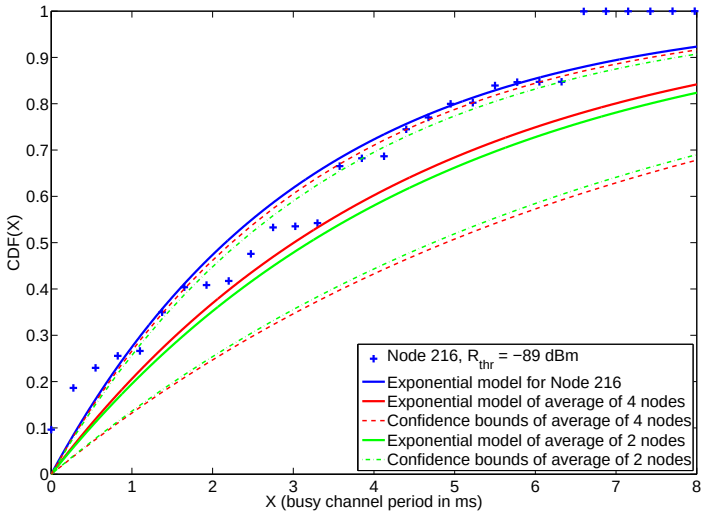
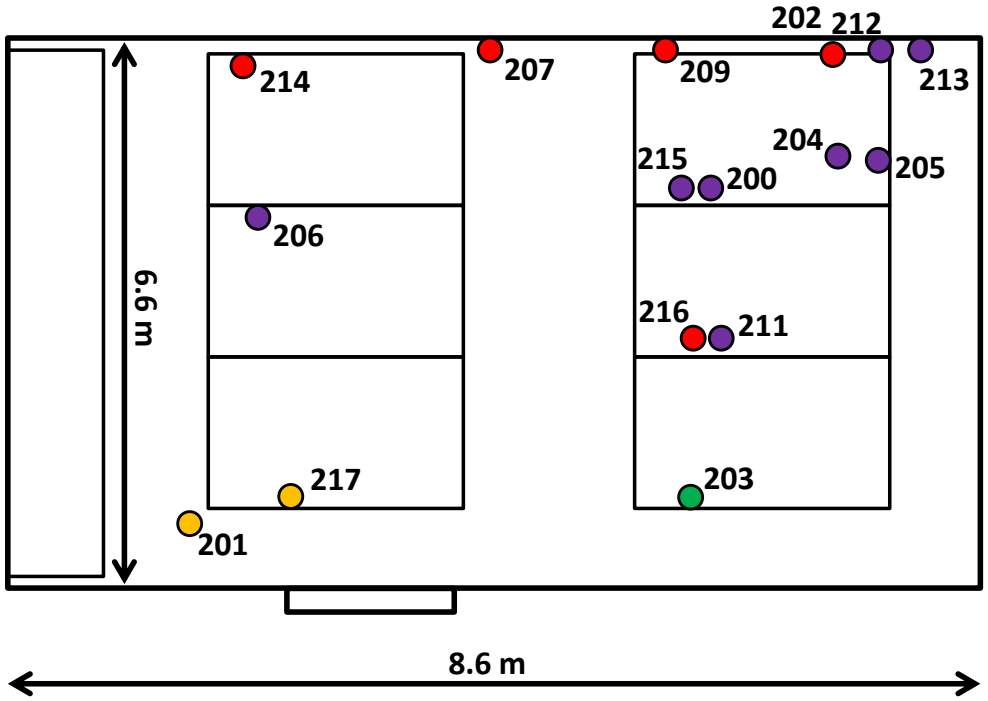


Figure 4.7: Aggregated fitting of 4 nodes (202, 207, 209, 214) and 2 nodes (207, 214), and node 216's busy periods when using $R_{thr} = -89dBm$.

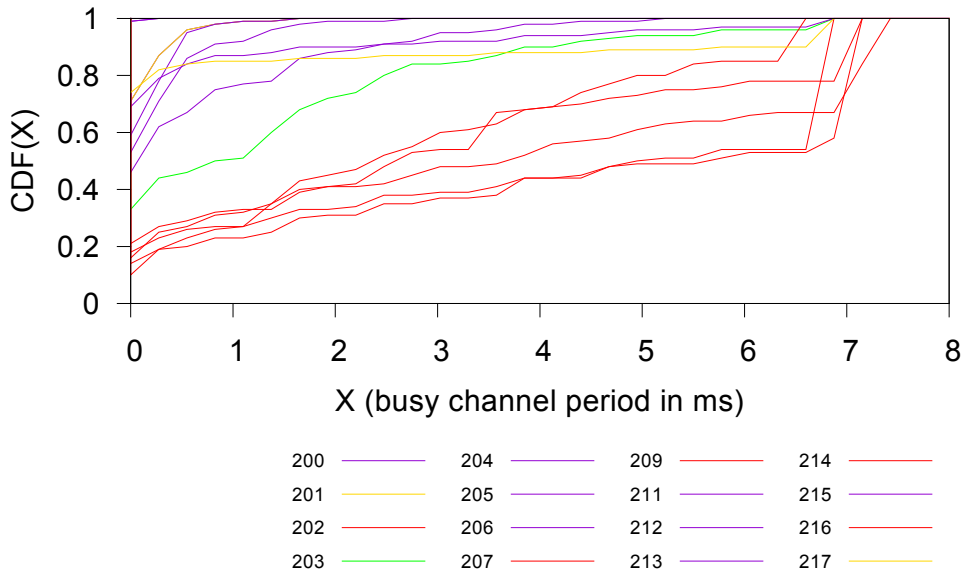
when instantiating the model as both fitted curves are nearly identical.

In a next step we consider node 216 (which is shown in blue in Figure 4.7). This node has not been included in the instantiation of the two models (shown in green and red), but, as we can observe, both models still cover reasonably well this node's busy period observations when considering the provided upper bound.

We have discussed here the aggregation of busy periods in an example deployment. The same approach can be used as well to aggregate idle periods. Aggregation can only be performed after interference patterns have been collected at each node. It has to be determined first, which nodes can be grouped together for an aggregated model. Thus, the effort for initial data collection cannot be reduced. However, using aggregated models has still a number of benefits. First, only one model is necessary which can be employed on a number of nodes which simplifies system configuration. Second, if interference patterns have to be monitored and verified it is sufficient to dedicate only a few nodes of an interference group to this particular task (if the same interference sources are present). This is particularly beneficial for implementation of runtime assurance.



(a) Location of the nodes



(b) CDF of busy periods

Figure 4.8: Cumulative distribution function (CDF) of busy periods among several Maxfor MTM-CM5000MSP nodes located inside a 56 m^2 office. The colors identify the nodes in which similar interference patterns were measured.

5 Platform: Receiver Signal Strength Indicator

Temperature is known to have a significant effect on the performance of radio transceivers, the higher the temperature, the lower the quality of the link. A model capturing the relationship between temperature and RSSI was presented in Deliverable D-1.1 [16] and summarised in Section 2.2.1. In this model, the effect of temperature on RSSI is modelled by looking at temperature effects on transmission power, receiver power, and noise floor. The effects are quantified in model parameters α, β and γ, where α represents the temperature effects on transmitted power, β on the receiver power, whilst γ on the noise floor. For this model to be instantiated, the value of each of these three parameters has to be known.

To calculate these values, we need tools that can collect RSSI measurements on the target platform. These measurements must not only be taken whilst the platform is exposed to different temperatures, but also in specific configurations. For example, to parametrise α, the effect of temperature on the transmitter must be isolated, and therefore we need RSSI measurements where only the temperature of the sender is varied whilst the temperature of the receiver remains constant. Likewise, for β (and γ), we need RSSI (noise floor) measurements where only the temperature of the receiver varies.

Whilst the model can be instantiated by parametrising α, β and γ with data collected using the approach outlined above, it is unknown whether these values are specific to the nodes (platform instances) used to collect the RSSI data. Analysis is required where data collected from multiple pairs of nodes are evaluated to see how similar the produced models are and if a single model can be used to represent all instances of a particular platform. Similarly, to produce an aggregated model for all platform instances, an understanding of how α, β and γ vary when different transmission powers are used, is also needed. Our aim through this analysis is to provide a single model instance for a given platform which would represent all instances of that platform at all transmission powers.

In the next section, we present a tool that runs on the platform being evaluated and collects RSSI and temperature data. We also describe how the tool can be used to capture a dataset. In Section 5.2 the collected dataset will be used to instantiate the model presented in Section 2.2.1. Next, we investigate whether a single platform model instantiation is sufficient or whether instantiations for individual platform instances are necessary. Similarly, an investigation on the validity of a single model instance covering all available transmission power is also given.

5.1 Collecting RSSI Data

For the RSSI platform model to be instantiated temperature and RSSI data must be collected. RSSI measurements must be recorded in multiple deployment configurations where the temperature of either the sender or receiver has been varied whilst the temperature of the other is

fixed. At the point of each transmission the temperature of both the sink and source system must be recorded along with the transmission power used. In this section a data collection application will be presented which can collect such data.

For the necessary data collection, we write an application for the Contiki operating system measuring RSSI, temperature, and transmission power settings. The application utilises packet transmissions between a pair of nodes to collect RSSI data. The source system transmits a packet containing a measurement of its on-board temperature and the transmission power selected. The receiver logs the measured RSSI, the temperature of the sender and receiver, and the transmission power used for transmission for every received packet. Alongside collecting RSSI data of received packets, the receiving system also measures the noise floor.

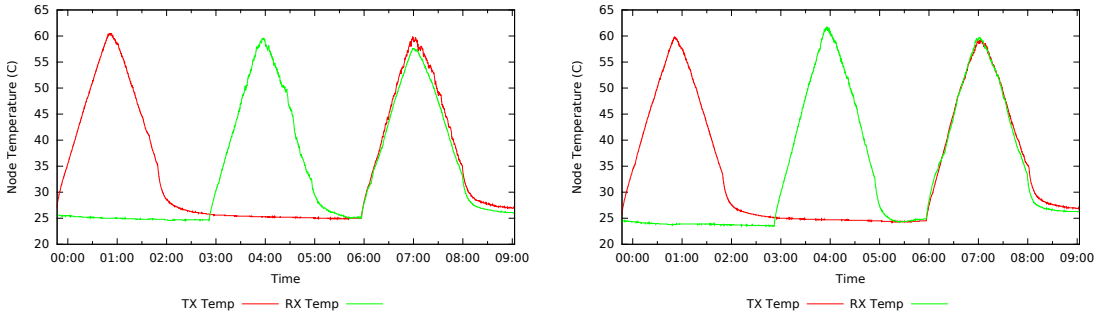
Source Firmware. The source application has a setup and an active phase. During the setup phase the application performs the general configuration of the platform including setting up its communication stack. The network stack is configured to use NULLMAC and NULL-RDC. These are MAC and Radio Duty Cycling (RDC) protocols without duty cycling. In this way, we can maximise the probability of packet reception and minimise the duration of an experiment.

During the active phase, the temperature is first measured and the transmission power is selected before entering the packet transmissions loop. There is an additional wait of $100ms$ after reading temperature to enable the sink to measure the noise floor. During each iteration of the packet transmission loop, the platform creates a packet containing a sequence number, the measured temperature and the transmission power. The packet is then transmitted to the receiving system. The system transmits 10 packets during the packet transmission loop, which should provide enough transmissions for each transmission power whilst executing quickly enough such that the temperature recorded at the beginning of the active period is still valid.

The active phase is continuously repeated, so to obtain sufficient sample points to obtain the necessary data. With each active phase, the application cycles through the available transmission power settings on the platform. This enables all transmission setting to be evaluated at each visited temperature point with only one run of the application. This cuts down on the number of application runs and the necessary executing time.

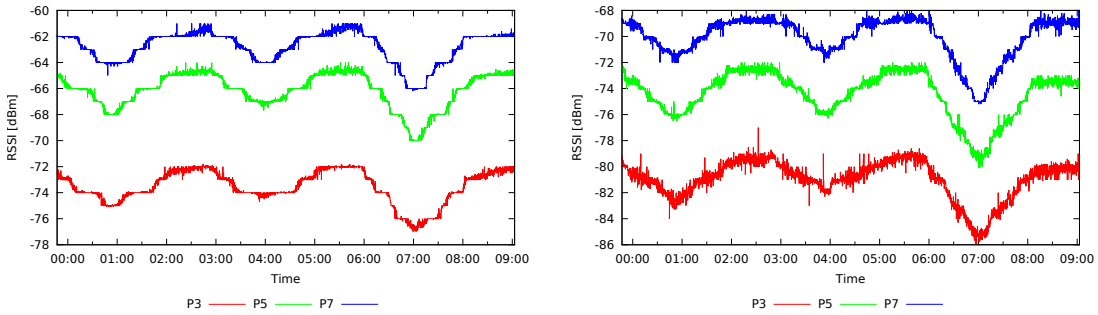
Receiving Firmware. Similarly, the receiving application has two phases: setup and active. During the setup phase the necessary system and network stack configuration are applied. In active mode, the device listens for packet transmissions measuring the RSSI for each given packet. In sync with the source system, using the sequence number, the node measures its on-board temperature every 10 packets. This ensures that both systems measure the temperature at the same point in time to minimise the chances of lost packets. After reading its temperature, the receiving system also takes 100 noise floor samples, recording the most frequent. The node outputs the measured temperature and noise floor at the start of each active phase using RS232. With each packet the measured RSSI, the local on-board temperature measurement, the on-board temperature of the sender and the transmission power selected is outputted.

Usage. The above application is deployed in a temperature controlled facility where the temperature of nodes can be controlled. This minimises the time required to collect the necessary data needed to parametrise the model. Nodes are used in pairs with each node heated individually. In one application run, the sender is heated whilst the temperature of the receiver



(a) Node 2 (b) Node 6

Figure 5.1: Temperature cycle during the RSSI data collection application.



(a) Node 2 (b) Node 6

Figure 5.2: RSSI Mesasurements for two nodes measured by the RSSI data collection tool.

is fixed to enable the effect of temperature on the sender to be measured. In a second application run, the temperature of only the receiver is changed to measure the effect of temperature change on the receiver. Multiple pairs of sender and receivers are used to increase measurement accuracy.

Results. We show results for collected data on two pairs of nodes. We used Maxfor MTM-CM5000MSP Telos-B clones, and placed the source and receiver nodes at approximately 3 meters distance. The temperature of each node was cycled from room temperature (25°C) to approximately 60°C over a time span of 2 hours. The temperature cycle followed a Gaussian curve. During the first cycle only the temperature of the sender was varied, whilst the temperature of the receiver was fixed. During the second cycle only the temperature of the receiver was varied. Finally, during the third cycle, both nodes were heated to allow us to see the overall effects. Heating each node separately and together isolates the effects of temperature on RSSI to either the receiver or sender. Figure 5.1 illustrates this heating in action, all three temperature cycles can be seen.

Figure 5.1 presents the RSSI measurement for the two sets of nodes. Whilst data was

collected for all transmission power levels, the figure only shows transmission powers 3, 5, and 7. Results for the other power levels follow the same curves offset by transmission power. For each plot the results show the expected strong correlation between changing temperature and RSSI. Regardless to whether the receiver or sender is heated, RSSI falls as heat is applied. The figure show that heating of both nodes has a cumulative effect on RSSI where the effect on the receiver can be added to the effect on the sender. Data from a second set of nodes was also collected which showed the same results. These two datasets will be used in the following section to instantiate the RSSI model, investigating aggregating platform instances and transmission power.

5.2 Model Instantiation

In this section the data collected by the RSSI collection application will be used to instantiate the RSSI model presented in Section 2.2.1. Initially the parameters α, β, and γ will be parametrised using data from a single pair of nodes evaluating how effective the model is to represent a platform instance. Next aggregation will be investigated, using data from multiple pairs of nodes to evaluate how well a single platform model can represents all instances of the platform. Finally, how well a single model instance represents all available power levels will be evaluated.

5.2.1 Parameterising the RSSI Model

We first derive values for the parameters α, β, and γ that are needed to model the effects of temperature on SNR given in Equation 2.16. Measurements collected for a single power level and node pair are used. The power level chosen is power level 3 (-25 dBm). Transmitter and receiver nodes are node 3 and node 2, respectively.

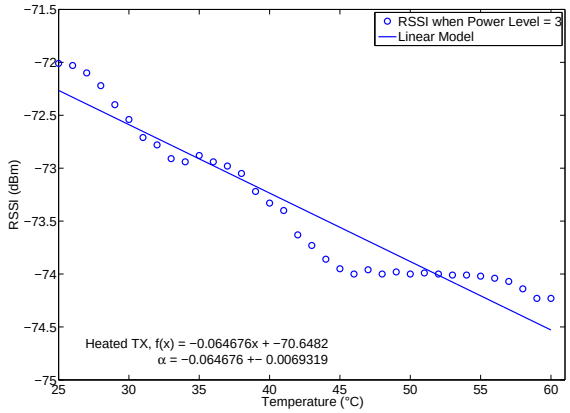


Figure 5.3: RSSI values with power level 3 when only transmitter is heated. (TX = Node 3, RX = Node 2)

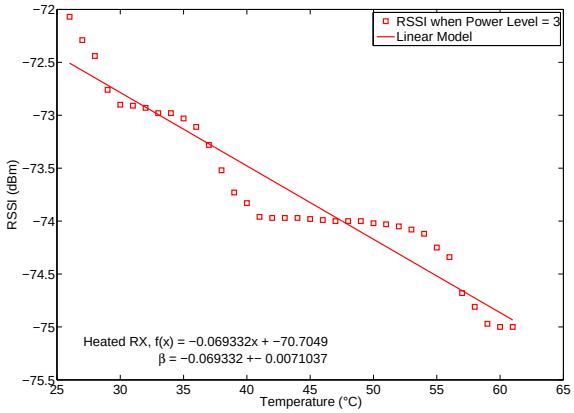


Figure 5.4: RSSI values with power level 3 when only receiver is heated. (TX = Node 3, RX = Node 2)

Figure 5.3 shows the relationship between RSSI and temperature obtained when only the transmitter is heated. The slope of the linear model of the RSSI values gives us the parameter

$\alpha = 0.065$. Figure 5.4 shows the relationship between the RSSI and temperature obtained when only receiver is heated. The slope of the linear model of the RSSI values gives us the parameter $\beta = 0.069$.

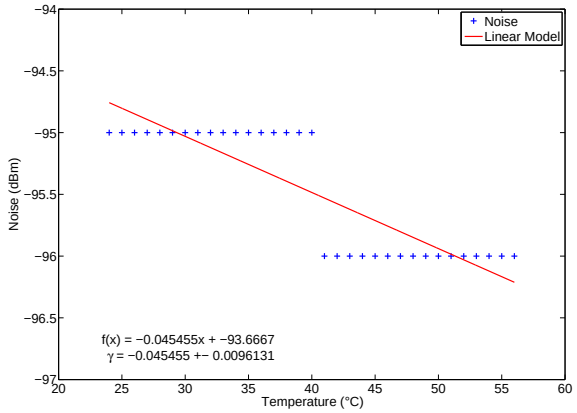


Figure 5.5: Observed changes in noise floor.

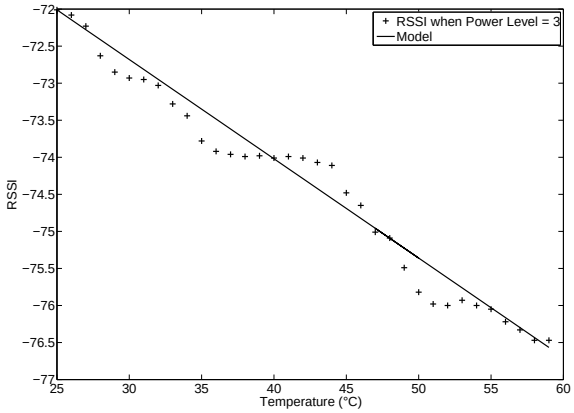


Figure 5.6: RSSI values with power level 3 when both transmitter and receiver are heated. (TX = node 3, RX = node 2).

Figure 5.2.1 shows the relationship between the noise floor and temperature. The slope of the linear model of the noise floor values gives us the parameter $\gamma = 0.045$. For completeness, Figure 5.2.1 shows the relationship between the RSSI and temperature obtained when both transmitter and receiver are heated. We used the model and the derived values above of $\alpha = 0.065$ and $\beta = 0.069$ to fit a line to the dataset. We can compare the model to a fitted linear trend, the slope of this line would be 0.132 whereas the slope derived by the model is $(\alpha + \beta) = 0.134$.

For the deployed nodes, the minimum and maximum temperature measured was 25°C and 55°C respectively. Over this temperature range, an attenuation in SNR of 3.04 dBm was observed when both transmitter and receiver were heated. Using the same temperature configuration, the model predicted that attenuation should equal to $(\alpha + \beta - \gamma)\Delta T = 2.67$ dBm, a deviation of 12% or 0.37 dBm.

5.2.2 Influence of Power Levels

In the previous section it was shown how values can be derived for the model parameters of a single node with a single transmission power level. Typically, platforms can adjust the selected transmission power to increase range, decrease interference or conserve energy. In this section we want to see the effect of temperature on SNR at different power levels. For this investigation, we first examine RSSI measurements of the link between node 3 and node 2 with transmission power levels 3 (-25 dBm), 16 (-7 dBm), and 31 (0 dBm).

Figure 5.2.2 shows the relationship between the RSSI and temperature obtained with the three different transmission power levels when only the transmitter is heated. α parameters

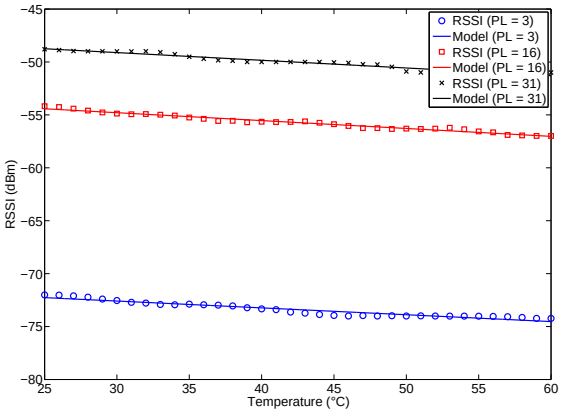


Figure 5.7: RSSI values with power levels 3, 16, and 31 when only transmitter is heated. (TX = node 3, RX = node 2).

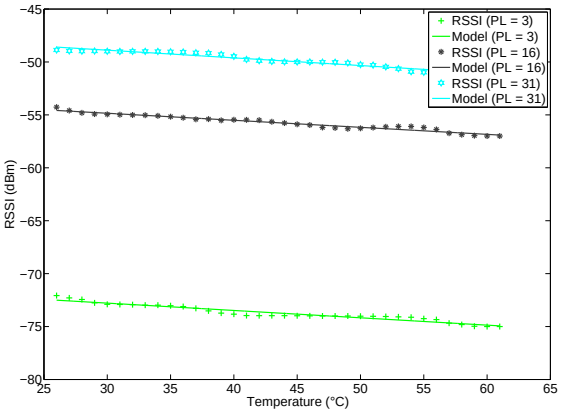


Figure 5.8: RSSI values with power levels 3, 16, and 31 when only receiver is heated. (TX = node 3, RX = node 2).

are 0.065, 0.074, and 0.073 with power levels 3, 16 and 31, respectively. Figure 5.2.2 shows the relationship between RSSI and temperature obtained with the three different transmission power levels when only the receiver node is heated. β parameters are 0.069, 0.066, and 0.073 with power levels 3, 16, and 31, respectively. There is a maximum deviation of 0.009 in α and 0.007 in β. The value of γ is not influenced by transmission power and thus is equal to the value derived in the previous section.

Whilst individual models for each power level can be instantiated, it would be beneficial to have one model for all transmission powers. Towards this, we calculate average, maximum and minimum α and β values for all power levels 3 to 31 inclusive (RSSI values are invalid for power levels below 3). We remove outliers from the α and β data sets. A value is decided as an outlier if it satisfies one of these two conditions: the change between the value and the previous value is greater than 1000% of the mean of 9 successive changes up to the previous value; and the change between the value and the next value is greater than 1000% of the mean of 9 successive changes after the next value. Table 5.1 presents the results. The table shows that the best (i.e., with minimum temperature effect) α and β values are achieved when the power level is 28, the worst (i.e., the highest temperature effect) α value is when the power level is 8, and the worst β value is when the power level is 24. We also find average, maximum and minimum $(\alpha + \beta - \gamma)$ calculation. It is shown in Table 5.2 that maximum SNR attenuation is observed when the power level is 8, and the minimum SNR attenuation is observed when the power level is 28. When instantiating the model, the difference in the minimum and maximum values over a temperature range of 30°C equals 1.77 dBm.

5.2.3 Aggregating Platform Instances

In this section we want to see the effect of temperature on SNR with the aggregation of multiple platform instances. The aim of aggregation is to allow a single platform model instance to be

	Average	Maximum	Minimum
α	0.075	0.092 (Power Level - 8)	0.064 (Power Level - 28)
β	0.069	0.081 (Power Level - 24)	0.048 (Power Level - 28)

Table 5.1: Average, maximum and minimum α and β values for all power levels 3 to 31 inclusive.

	Average	Maximum	Minimum
$(\alpha + \beta - \gamma)$	0.099	0.126 (Power Level - 8)	0.067 (Power Level - 28)

Table 5.2: Average, maximum and minimum $(\alpha + \beta - \gamma)$ calculation for all power levels 3 to 31 inclusive.

used to represents all instances of a platform. Whilst this will reduce the precision of the model when compared to measurements taken at a single link, it will decrease the computation of applying the model and necessity of collecting data on every platform instance.

We take the average RSSI values of the link between node 3 and node 2 and the link between node 5 and node 6. Power level is chosen as 3. We removed the outliers from the aggregated data set. A value is an outlier if it satisfies one of these two conditions: the change between the value and the previous value is greater than 2000% of the mean of 9 successive changes up to the previous value; and the change between the value and the next value is greater than 2000% of the mean of 9 successive changes after the next value. The value of 2000% is high enough to offer high confidence in that the samples removed are outliers.

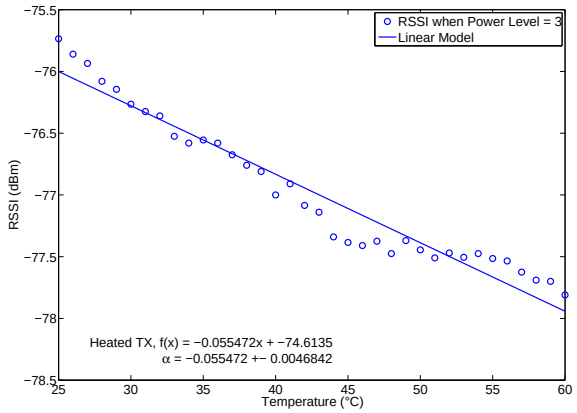


Figure 5.9: Average RSSI values of two links with power level 3 when only transmitter is heated. (TX 1 = Node 3, RX 1 = node 2, TX 2 = node 5, RX 2 = node 6).

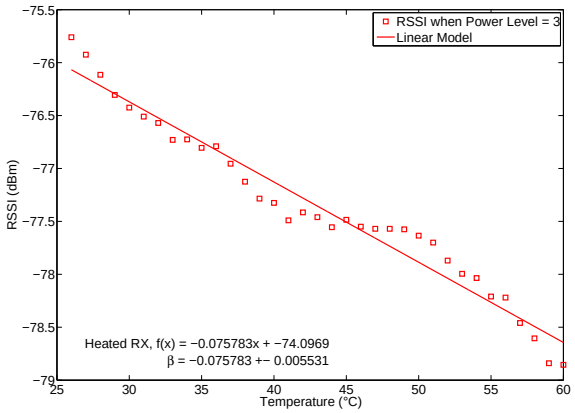


Figure 5.10: Average RSSI values of two links with power level 3 when only receiver is heated. (TX 1 = Node 3, RX 1 = node 2, TX 2 = node 5, RX 2 = node 6).

Figure 5.2.3 shows the relationship between the RSSI values of two links and temperature

obtained when only transmitters are heated. The slope of the linear model of the RSSI values gives us the parameter $\alpha = 0.055$. Figure 5.2.3 shows the relationship between the RSSI values of two links and temperature obtained when only receivers are heated. The slope of the linear model of the RSSI values gives us the parameter $\beta = 0.076$. Although the individual parameters change in comparison to only a single node, $(\alpha + \beta - \gamma)$ for this aggregated instance is 0.89 which compares to 0.9. This would equate to a difference of 0.03 dBm between the two models over a 30°C temperature range.

5.2.4 Bounding

In the previous sections it was shown how to fit the RSSI model to a single node, all power level and aggregation of multiple nodes. This model instantiation aims to provide the best fit to the average RSSI change with varied temperature. This average case has limited utility. Bounding of this model is required to increase its usefulness. A worst bound would enable the worst case change in SNR for a link between two nodes to be calculated with regards to varying temperature.

For RSSI, model bounding will be achieved by applying a single negative offset to the model to provide a worst case. This bound should be high enough to provide an adequate safety margin such that the majority of RSSI measurements fall above this bound. Whilst there are various methods that can be employed to calculate the offset used for this bound, we calculate this offset by summing the separate offsets to enable X% of samples to be included when only the transmitter is heated and when only the receiver is heated. We choose 80% for value of X.

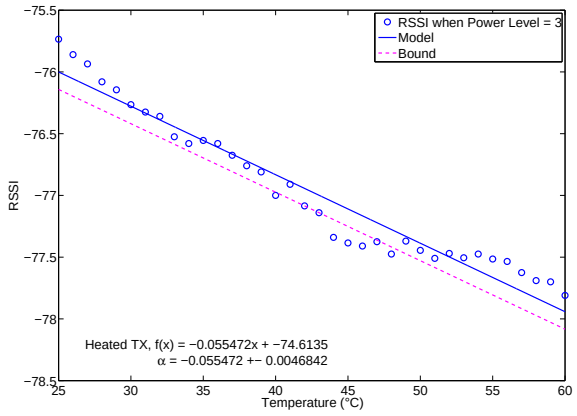


Figure 5.11: Average RSSI values of two links with power level 3 when only transmitter is heated with an 80% bound. (TX 1 = node 3, RX 1 = node 2, TX 2 = node 5, RX 2 = node 6).

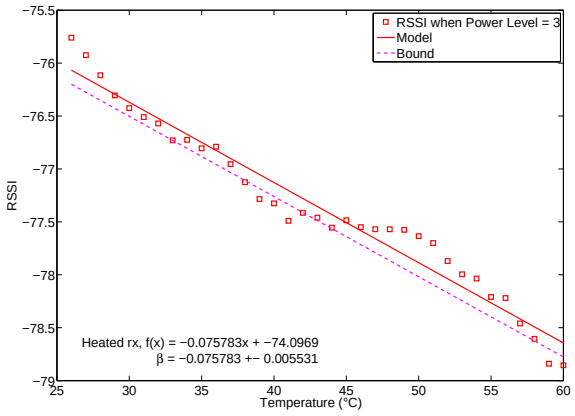


Figure 5.12: Average RSSI values of two links with power level 3 when only receiver is heated with an 80% bound. (TX 1 = node 3, RX 1 = node 2, TX 2 = node 5, RX 2 = node 6).

To investigate bounding, we use the aggregated dataset used in the previous section. Figure 5.2.4 shows the relationship between RSSI and temperature for this aggregated dataset

were only the transmitter is heated. The figure includes a bound where 80% of sample points fall above. The offset for this bound is calculated as 0.1415 dBm. Similarly, in Figure 5.2.4 the bound is shown in the case in which only the receiver is heated. The offset for this bound is calculated as 0.1325 dBm.

We take these two calculated offsets and sum them to determine the overall bound for the complete model. The sum of these offset is 0.274 dBm. Figure 5.13 presents the relationship between RSSI and temperature in which both receiver and transmitter are heated. The instantiated model for this dataset uses values of $\alpha = 0.055$ and $\beta = 0.076$ (calculated in the previous section) with an offset set to the value of the first sample. A bound line with the offset 0.275 dBm calculated above is also shown. To determine how effective this bound is, we calculated the percentage of sample points that lie above this bound as 40%. This differs by 40% to the 80% used on the two individual datasets above to calculate this offset. This percentage is relatively low and can be explained to the offset used to fit the model to the dataset. An offset which enabled the model to better fit the samples would lead to a bound that included more sample points. The same bounding approach could be used to improve the utility of the model fitted to the aggregation of power levels.

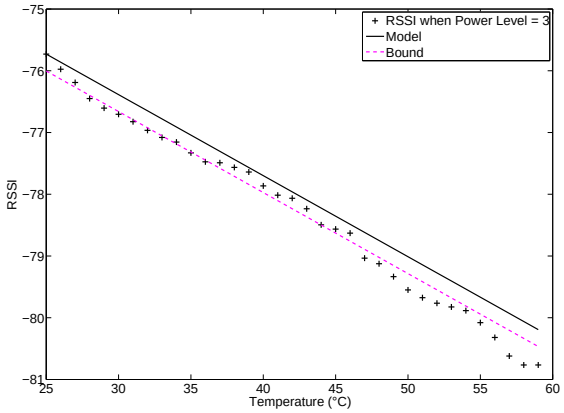


Figure 5.13: Average RSSI values of two links with power level 3 when all nodes are heated with an 80% bound. (TX 1 = node 3, RX 1 = node 2, TX 2 = node 5, RX 2 = node 6).

6 Platform: Timing

In Section 2.2.2 two models were presented which describe the relationship of the RT and RC clock to temperature. These models enable protocol designers to understand the changes in clock rate to mitigate their effects. For either model to be used, a number of measurements of the clock rate for the clock being modelled at various temperature points are required.

This chapter will first examine how to construct a tool to collect this timing data which can be used to instantiate the timing model. This tool is then used to collect an example dataset of timing/temperature measurements for three nodes. In Section 6.2, the results collected are used to instantiate the model. Next an examination of the number of samples required to instantiate both models will be presented with the aim of reducing the number of samples required and thus collection time. Following this, methods to determine bounds and to aggregate measurements are discussed.

6.1 Measuring Timing Characteristics

There are two basic approaches to measuring the timing characteristic of a mote. The first makes use of sophisticated lab equipment for example, a calibrated oscilloscope, to measure the clock frequency. The second makes use of the clock of a secondary system (for example, PC or mote) to measure the clock frequency. Either of these approaches can be used to measure the clock directly by attaching to the clock source or indirectly by timing an event such as the toggling of a GPIO pin or time between two packets controlled by the clock being measured.

In this section tools for the collection of timing data will be examined, the first is lab-based and the second is deployment-based. During lab-based collections, the platform would be examined before the deployment off-line, aided with lab equipment. For deployment-based collections the platform would be examined at the point of deployment during the setup phase prior to the system/application going live. This section will examine these two methods.

6.1.1 Lab-Based Approach

Naturally, measuring clock rate and temperature in the lab should yield the most accurate results. Clock rate can be measured with tools such as a calibrated oscilloscope whilst at the same time the on-board temperature of the platform can be tightly controlled enabling measurements to be taken at exactly the sample points required. This, however, comes at the cost of requiring expensive lab time for each platform or even each platform instance.

For lab-based data collection, a simple application was written for the Contiki operating system which would toggle a GPIO pin at fixed rate dependent on the clock being examined. The software has a setup phase where the system is configured ready for sampling and then an active phase where in a loop the temperature is measured and outputted before toggling the GPIO pin at the configured rate.

During the setup phase, the software configures two GPIO pins for output, configures the clock source being measured, and disables any activity that may interrupt the sampling process (e.g., interrupts/watchdog).

During the active phase, the node enters a temperature measurement period where temperature is measured and outputted via RS232. Next the system enters the clock sampling period where a GPIO pin, the trigger pin, is raised to mark the start of sampling. The second GPIO pin, the clock output pin, will then be toggled at a rate depending upon the clock being observed. After 100 output cycles, the trigger pin is lowered to mark the end of the clock sampling period. These two periods then repeat until the application is terminated.

The rate at which the clock output pin is toggled will depend on the clock source being measured. For a peripheral clock source, the software would toggle the pin every n clock ticks of the source clock. Whilst for the processor clock, the processor would perform n NOP operations between each pin toggle.

The value of n must be sufficiently large to ensure the clock period is large enough to remove any variance induced by the measurement software. The software uses a simple loop and if statement to determine when toggling is necessary. This simple set of instructions can add a small amount of variance to the time length of the clock period depending at the point of execution when the clock source state changes.

The platform is plugged into a PC to record the temperature output log, time stamping each temperature output using the PC system time. Both the trigger GPIO pin and the clock output GPIO pin are connected to a calibrated oscilloscope. The oscilloscope is configured to measure the period that the clock output pin is high whilst the trigger pin is set. These measurements are recorded by the oscilloscope to a flash drive where they are also time-stamped to enable correlation with the recorded temperature. Both the PC log and the oscilloscope log are later parsed using a simple script which outputs time, temperature and clock rate which can then be used in the model.

As the platform would normally sit at the ambient temperature of the lab, the temperature of the on-board platform must be controlled to enable the measurements at the sample points required. A TempLab facility can be used to control the temperature of the node. Using the heating lamp or the Peltier enclosures (LO nodes or PE nodes [5]), the temperature of the device can be finely tuned. Typically a Gaussian function would be used to produce a temperature curve, from room temperature up to the maximum lamp output temperature with a temperature step time sufficient to enable sampling.

This tool was used to collect the necessary data used to derive and validate the models in Deliverable D-1.1. Whilst collecting clock data in the lab is a valid approach, it can be a costly exercise. Another approach would be to measure the variances in clock rate at the deployment during setup: this would be both cheaper and would utilise the actual nodes being deployed to collect the data. This should produce a better fitted model tailored specifically to the platform instances used at the deployment.

6.1.2 Deployment-Based Approach

For the deployment-based approach, we have written an application for the Contiki operating system, which uses the timing of packet transmissions to determine the clock rate of the sending system. The source system transmits a packet containing a measurement of its on-board

temperature at a fixed interval controlled by the clock source being measured. The receiver receiving these packets will log the temperature of the sender and the inter-arrival time between packets. Using the inter-arrival time, the clock rate of the source system can be calculated to provide the necessary data (temperature/clock rate) required for the model.

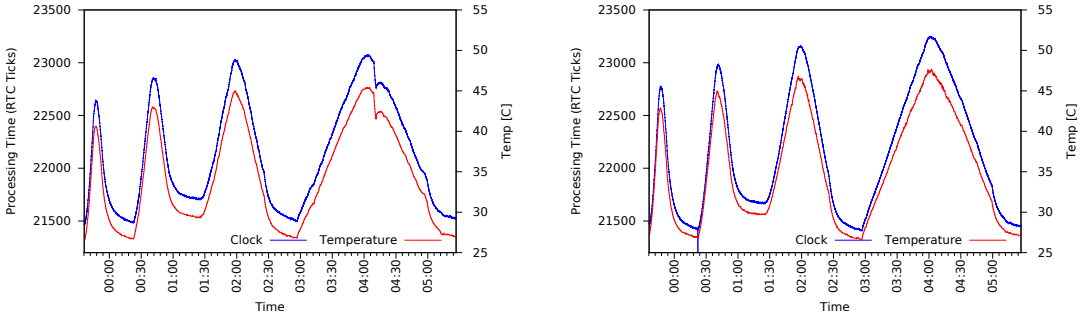
Source Firmware. The client application has a setup and an active phase. During the setup phase the application performs the general configuration of the platform including setting up its communication stack and the clock sources that the application samples. As stable packet transmission timing is essential, variances caused by MAC duty cycling or re-transmissions must be eliminated such that there is no effect on the clock rate sampling. ContikiOS NULL-MAC/RDC was used which performs no duty cycling or retransmission and 802.15.4 back-off mechanism was also disabled. This should enable the timing between the request for packet transmission and the start of actual transmission to be fixed. Interrupts and other sub-systems cannot be disabled as they are required for system and network operations and as such may lead to some variance in measurements. These variations are overcome by collecting a large number of samples and removing outliers during model instantiation.

During the active phase, the temperature is first measured before entering the packet transmissions loop. During each iteration of the packet transmission loop, the platform uses the clock being measured to perform a fixed period wait before constructing and sending a packet. The packet contains a sequence number, the measured temperature and a field identifying the clock source being measured. The number of packets sent has to be small enough such that the temperature recorded at the beginning of the active period is still valid. However, it is important that a large number of samples for a given temperature can be recorded to remove errors caused by fluctuations in processing time during packet transmission and the effects of any interrupts.

The active phase is continuously repeated to provide sufficient sample points to provide the necessary data. With each active phase, the application cycles through each of the available clocks on the system with each active cycle. This enables all clocks on the system to be evaluated at each visited temperature point with one run of the application. This should cut down on the necessary time needed during deployment setup to model timing.

Sink Firmware. Similarly, the server application has two phases: setup and active. During the setup phase the necessary system and network stack configuration are applied. In active mode, the device listens for packet transmissions, calculating their inter-arrival time. This calculation is only performed for consecutive packets (confirmed by sequence number) and packets that pertain to the same clock source (identified in the packet). Between the reception of each packet, the sink measures the local temperature to confirm its temperature stability. It is important that the receiver remains temperature stable, fluctuations in temperature could lead to changes in its own clock, which would lead to invalid results. The sink for each packet outputs the clock ID, the sink temperature, the source temperature and the measured inter-arrival time.

Usage. The application should be deployed on the nodes during the setup phase of a deployment. It is envisaged that nodes would be programmed one at a time with the source firmware. A laptop with connected mote would run the sink firmware collecting data from each of the motes. It is required to collect timing samples at different temperature settings.



(a) Node 11 (b) Node 13

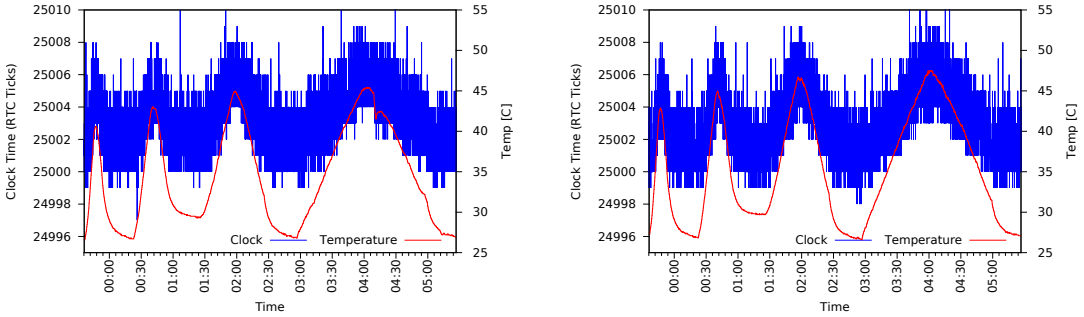
Figure 6.1: Processor RC clock from two nodes.

Results. We show results for three pairs of nodes. Using the TempLab facility enabled the application to be more thoroughly tested as temperature can be controlled. The platform selected for these tests was the Maxfor MTM-CM5000MSP Telos-B clones. Nodes where placed at approximately 2 meters from sensor to receiver. The temperature of each node was cycled from room temperature 27°C to approximately 50°C over a time span of 15*mins*. This was repeated three times doubling the duration of the temperature cycle to 30*mins* to 60*mins* and 120*mins* with a rest period of 30 mins between each cycle. The test was repeated three times with three different nodes acting as source to generate data from different instances of the platform.

Figure 6.1 presents the results for the processor clock. It confirms the results seen in D-1.1 with a correlation between the time taken by the process and therefore clock rate and temperature. As temperature increases, the time taken increases and, similarly, as temperature falls the time taken falls. The plot shows data for two nodes, the third node showed the same correlation. Figure 6.2 presents the results for the RTC clock. The same correlation between temperature and clock speed can be seen. The data collected and shown illustrated in these plots will be used in the following section to instantiate the two clock models.

6.2 Timing Model Instantiation

In this section, the example results collected in the previous section will be used to validate the collection tool by instantiating the model. Next an examination of the number of samples required to instantiate both models will be presented with the aim of reducing the number of samples required and thus collection time. Following this method to bound the model with the necessary upper and lower bounds required for WP3 will be examined. Finally, an examination of aggregation will be presented, evaluating the possibility of aggregating the models of multiple nodes to form a single platform specific model.



(a) Node 11 (b) Node 13

Figure 6.2: RTC Clock from two nodes.

6.2.1 Primary System Clock

The model shown in Equation 2.17 was used to describe how the RC clock changes with temperature. To instantiate this model, the data collected using the Maxfor MTM-CM5000MSP platform in the previous section was used. Room temperature, T_0, is generally assumed to be 25°C, thus we use this value for our calculations.

During data collection, multiple processing time samples were recorded at each temperature point. Some of these recorded processing times are significantly different to others. This large difference in values will be down to variances in the running system and these should be ignored. Thus, before applying the model we take the average of processing time values for each temperature value, whilst also removing outliers. We decide if a value is an outlier if it satisfies these two conditions: the change between the value and the previous value is greater than 500% of the mean of 4 successive changes up to the previous value; and the change between the value and the next value is greater than 500% of the mean of 4 successive changes after the next value. We use a large value when removing outliers to give a large confidence that values removed are outliers. After removing outliers, we calculate processing time change in percentage from the temperature when the experiment started, the processing time for the first observed temperature is chosen as the reference speed.

The processor clock was first examined. Several outliers observed at temperature points 30.4, 47.9 and 49 were removed. Figure 6.3 shows the relationship between the averaged processing time and temperature with the derived model for one example node (node 7). The same model process could be applied to the data collected for the other nodes. The Y-axis shows processing time change in percentage whilst the X-axis shows temperature. The determined temperature coefficient is 3.89E-01.

The residuals are used to evaluate how well the model fits the data measured by node 7. Each residual shows the difference between the model and actual data point. Figure 6.4 shows the residuals for processor clock from fitting the model to node 7. The measured data fits the model with an average residual of $2.98E-01\%$. Maximum residual of $6.05E-01\%$ is observed at 26.7 °C. These residuals show that the model is slightly biased.

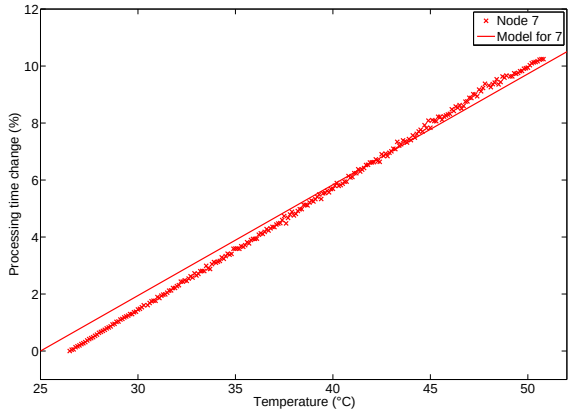


Figure 6.3: Model fitting for RC clocks.

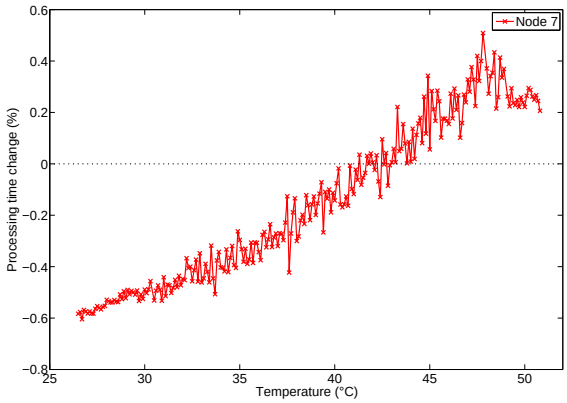


Figure 6.4: RC clock model residuals.

Node	$A_1[\%/°C]$	$A_1[\%/°C^2]$	$A_1[\%/°C^2]$
Node 7	-1.51E-04	8.93E-05	-2.10E-06

Table 6.1: Determined RT temperature coefficients for Node 7.

6.2.2 Real-Time Clock

For the Real-time clock, we repeat the same steps undertaken in the previous section. We use data from node 7 to instantiate the RT clock platform model. Similarly to the RC Clock a number of outliers where recorded at temperatures 28.6, 29.4, 34.2, 36.9 and 44.8, these where removed from the dataset. Figure 6.5 shows the relationship between the RTC clock time and temperature as well as the derived model. The x-axis shows temperatures and the y-axis shows change in the RTC clock expressed as a percentage. The determined temperature coefficient is shown in Table 6.1.

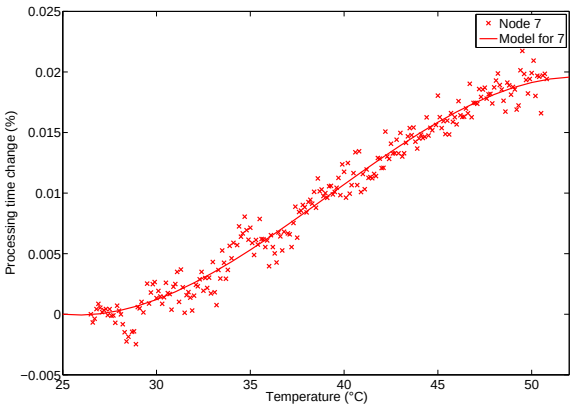


Figure 6.5: Model fitting for RT clocks.

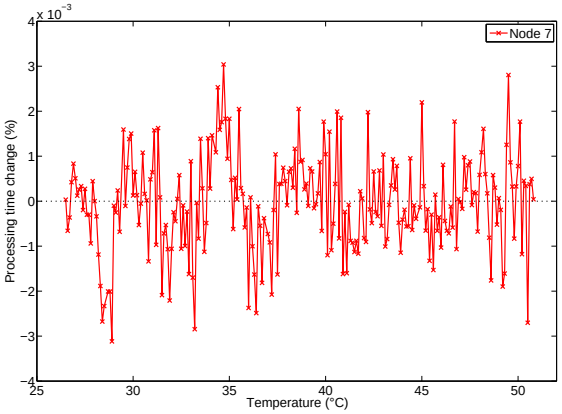


Figure 6.6: RT clock model residuals.

Figure 6.6 shows the residuals of fitting the RT clock model. The measured data fits the

	Node 7 - RC clock	Node 7 - RT clock
Experiment 1	1.17E-01	1.15E-06
Experiment 2	1.46E-01	1.24E-06
Experiment 3	1.72E-01	4.58E-06

Table 6.2: Average RSS values between the first experiment's data and the model curves for each experiment.

	Node 7 - RC clock	Node 7 - RT clock
Experiment 2	25.00%	7.6%
Experiment 3	47.64%	298.57%

Table 6.3: Average RSS increases in percentage of Experiments 2 and 3 relative to average RSS of Experiment 1.

model with an average residual of $8.25E-04\%$. Maximum residual of $3.10E-03\%$ is observed at 28.9 °C. These residuals are unbiased and show that the model is a good fit for the provided data.

6.2.3 Reducing the Sample Size

Instantiating the RC and RT model we used data collected with the deployment-based application. The application collected a significant number of timing samples at each temperature point. This large number of data samples may not be necessary to accurately instantiate the model. Reducing the number of samples necessary will reduce the time taken, the storage and the required processing. This section will repeat instantiating the model with a reduced data set to evaluate the effect of reduced sample sizes.

During data collection we recorded over 500 samples for each 5 °C interval, this will be regarded as the best case. We compare this configuration with two others which contain a reduced sample size. In the first configuration we use 100 samples for every 5 °C interval which is a 5-fold saving and for the second we use 11 samples for each 5 °C interval which is approximately a 50-times reduction. A meaningful model could not be instantiated when a sample sizes of less than 11 was used. As the comparison metric, we use average residual sum of squares (RSS). Average RSS values between the best case and the model curves for each of the reduced sample configurations are shown in Table 6.2. Error between the best case data and the model curves increases as the sample size decreases. Table 6.3 shows RSS increases in percentage of configuration 1 and 2 relative to RSS the best case. The percentage increase in RSS values for both reduced sample points are significantly higher with the RC clock and therefore more samples would be required to accurately instantiate the model. For the RT clock they are 7% higher with a sample size of 100 which may be acceptable, but nearly 300% higher when only 11 samples are considered.

6.2.4 Model Aggregation

So far in this section, the model for a single node, node 7 has been instantiated and evaluated. Whilst instantiating a single model for each node is possible, it can be time-consuming and unpractical. Instead it would be beneficial to have a single timing model instantiation for each of the RC and RT clocks which can be used to represent all instances of the platform. In this section, the aggregation of models is examined.

Continuing the examination of node 7, we aggregate the recorded samples from node 7 and node 11. As when instantiating a model for a single node, outliers must first be removed. We take the average calculated clock rate values of two nodes' for each temperature point. We calculate the changes in clock rate as a percentage and fit the models given in Equation 2.17 and Equation 2.18 for RC and RT clocks, respectively.

Figure 6.7 and Figure 6.8 show the model fittings for the average of two nodes for RC and RT clocks, respectively. Average RSS increases in percentage of RC clock and RT clock relative to average RSS of only considering node 7 are 25.00% and 7.66%, respectively.

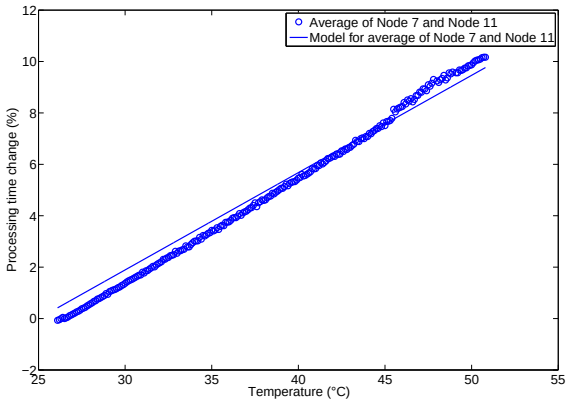


Figure 6.7: Model fittings of average of two nodes for RC clock.

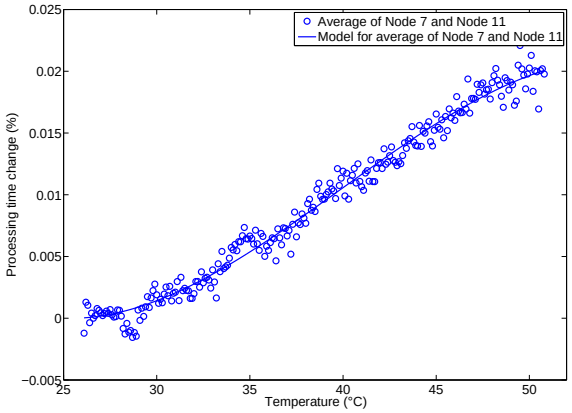


Figure 6.8: Model fittings of average of two nodes for RT clock.

To further examine aggregation, we aggregate three nodes (7, 11 and 13). Again we first remove outliers of nodes individually, and take the average clock time values of the three nodes' data for each temperature. We then calculate processing time changes in percent and fit the models.

Figure 6.9 and Figure 6.10 show the model fittings of the average of the three nodes for RC and RT clocks, respectively. Average RSS increases in percentage of RC clock and RT clock relative to average RSS of node 7 are 88.05% and 127.50%, respectively.

6.2.5 Model Bounding

The model provides limited advantages in helping protocol design/selection. Instead upper and lower bounds around the model would be more beneficial. Bounding is necessary to calculate protocol elements such as guard times. We calculate the bounds around the model by selecting a positive offset for the upper bound and a negative offset for the lower bound. These individual

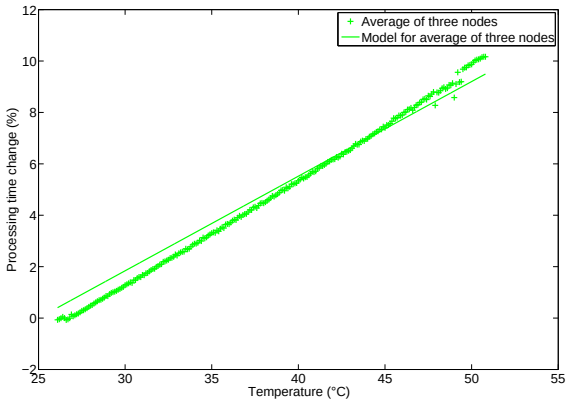


Figure 6.9: Aggregated model fittings of three nodes for RC clock.

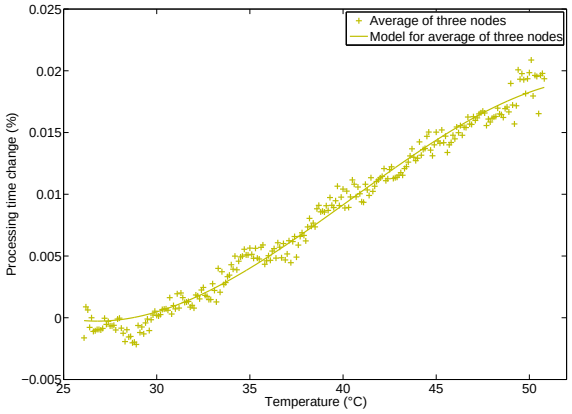


Figure 6.10: Aggregated model fittings of three nodes for RT clock.

offsets will be calculated to include X percent of measurement points. We selected 80% for the value of X.

To investigate bounding, we take the aggregated model for the two nodes used in the previous section. For both models we calculate the upper and lower bounds which would include 80% of samples. The offset to the model for upper bound for the RC-Clock was calculated as 0.3015 and the lower bound offset was calculated as 0.5. For the RT-Clock the offset for both the upper and lower bound was calculated as 0.001.

Figure 6.11 and Figure 6.12 illustrates these bounds around the respective models. To evaluate how effective these bounds are, we take data collected from a different node (node 13) and measure how well the model with bounds fits the measurements for this node. In Figure 6.11, 36% of measured points lie within the calculated bounds of the model. In Figure 6.12, 45% of measured point lie within the calculated model. For both models, the percentage of measured sample points of node 13 that fit within the model is low. Whilst this can be increased by increasing the bound interval of 80% for higher accuracy, nodes such as 13 whose data is found to be significantly different to other nodes should be included in the initial aggregation.

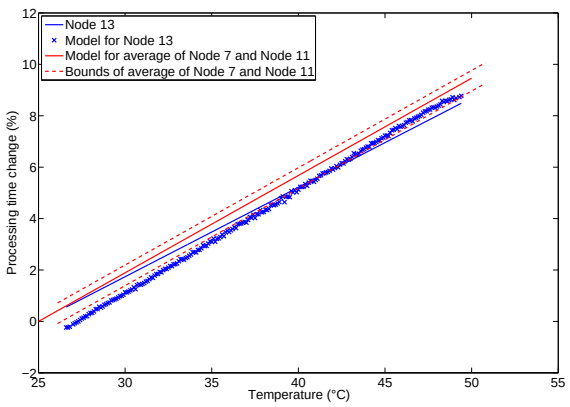


Figure 6.11: Aggregated model fittings to node 13 with bounds for RC clock.

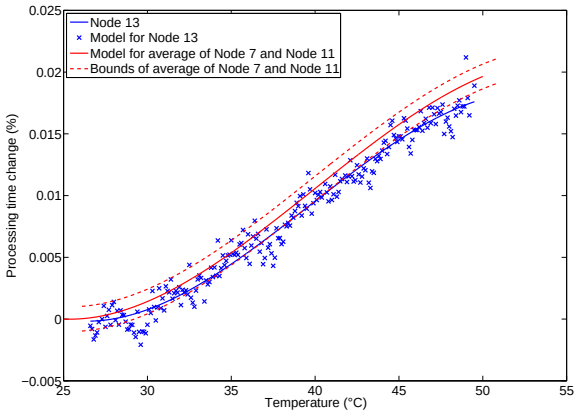


Figure 6.12: Aggregated model fittings to node 13 with bounds for RT clock.

7 Conclusions

In Deliverable D-1.1 we developed generic environmental and platform models which can be used to estimate the performance of sensornet deployments and identify optimal operating configuration. Focus has been on environmental aspects of temperature and interference and the platform aspects of timing and receiver signal strength. Over the last five months these earlier models have seen further work which has led to a number of refinements.

For these models to be effective, they must be fitted to specific deployment environments and platforms. For this fitting to be possible, tools are required to collect the necessary data to parametrise these models. This document has presented tools for collecting temperature, interference, signal strength and timing data from the deployment environment and platforms used for deployment. It was shown how example datasets collected via these tools can be used to parametrise the developed models. Aggregation and bounding methods for each model were evaluated to minimise computational requirements of using these to select optimal operating configurations in WP3.

For our next steps, we plan to use these parametrised models to replicate deployment conditions to develop and optimise communication protocols. Tools for runtime assurance must also be developed which will be initially based upon the tools described in this deliverable but will have to work with a much limited resource set to run alongside deployed applications.

Bibliography

[1] K. Bannister, "Impacts of thermal reduction in transceiver performance on outdoor sensing networks," Master's thesis, Arizona State University, Phoenix, AZ, USA, 2009.

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

[3] 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} *IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, apr 2011, pp. 175–186.

[4] 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)*, dec 2012, pp. 315–326.

[5] C. A. Boano, F. J. Oppermann, K. Römer, J. Brown, U. Roedig, C. Keppitiyagama, and T. Voigt, "D-4.2 - prototype of testbeds with realistic environmental effects," http://www.relyonit.eu/, RELYonIT: Research by Experimentation for Dependability on the Internet of Things, Grant Agreement no: 317826, Tech. Rep., Oct. 2013.

[6] 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.

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

[8] 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)*, Nov. 2004.

[9] 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)*, Cork, Ireland, Feb. 2009.

[10] 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)*, Coimbra, Portugal, Feb. 2010.

[11] F. Hocaoğlu, O. Gerek, and M. Kurban, "Hourly solar radiation forecasting using optimal coefficient 2-D linear filters and feed-forward neural networks," *Solar Energy*, vol. 82, no. 8, 2008.

[12] A. Kamerman and N. Erkocevic, "Microwave oven interference on wireless LANs operating in the 2.4 ghz ISM band," in *Proceedings of the* 8^{th} *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIRMC)*, vol. 3, sep 1997, pp. 1221–1227.

[13] D. Lane, K. Goris, and R. Somerville, "Radiative transfer through broken clouds: Observations and model validation," *Journal of Climate*, vol. 15, no. 20, 2002.

[14] H. Wennerström, F. Hermans, O. Rensfelt, C. Rohner, and L.-A. Nordén, "A long-term study of correlations between meteorological conditions and 802.15.4 link performance," in *Proceedings of the* 10^{th} *IEEE International Conference on Sensing, Communication, and Networking (SECON)*, Jun. 2013.

[15] Y. Liu et al., "IEEE 802.11 WLANs WG Group Information doc. no. 802.11-10/1079r0," Sep. 2010.

[16] M. A. Zúñiga, C. A. Boano, J. Brown, C. Keppitiyagama, F. J. 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.