RELYonIT
dependability for the internet of things

Research by Experimentation
for Dependability on the
Internet of Things

SEVENTH FRAMEWORK
PROGRAMME

D-4.3 – First Integrated Prototype and Experiment

Grant Agreement no: 317826
www.relyonit.eu

Date: May 8, 2014

Author(s) and affiliation: Nicolas Tsiftes, Thiemo Voigt (SICS), Faisal Aslam, Ioannis Protonotarios, Marco Antonio Zúñiga, Koen Langendoen (TUD), Carlo Alberto Boano, Felix Jonathan Oppermann, Kay Römer, Marcel Baunach (TUG), James Brown, Utz Roedig (ULANC), Patricio Moreno Montero, Rafael Socorro Hernández (ACCIONA), Màrius Montón, José Carlos Pacho (WOS)

Work package/task: WP4
Document status: Final
Dissemination level: Public
Keywords: Internet of Things, sensor networks, interference, temperature

Abstract This document presents the work carried out in Task 4.3—First Integrated Experiment. We first describe the design of the first integrated prototype, which includes selected protocols from Task 2.2—Protocol Design. The protocols are designed to improve their performance by using the learning of environmental and platform models developed in WP1. We then present an experimental evaluation of the integrated prototype, which we conduct both in a set of testbeds and in a real-world deployment in Barcelona. The testbed evaluation is carried out with the help of tools developed in Task 4.1—Testbeds with Realistic Environmental Effects. These tools support experiments in which harsh environmental conditions—both with respect to interference and temperature—can be generated in a repeatable manner with predetermined patterns of variation. We relate the outcome of these results to the use case requirements of Deliverable D-4.2, and present the next steps to be taken in the project based on what we learned from the first integrated experiment.

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

This deliverable, created in the context of Task 4.3 *First Integrated Experiment*, presents the design and implementation of our integrated prototype. The integrated prototype combines the components developed in WP1-3: including 1) learning of environmental models and parameters, 2) run-time assurance of the environmental models, and 3) newly developed and optimized protocols that make use of the environmental models to mitigate the adverse effects of temperature variation and interference.

A key objective of the integrated experiment is to evaluate the work conducted hitherto in WP1-3 with respect to the use case requirements specified in Deliverable D-4.1. Specifically, the integrated prototype focusses on the Outdoor Parking Management use case (SmartParking) that has been selected for implementation in Task 4.2. We further use several different testbeds augmented with the tools developed in Task 4.1 *Testbeds with Realistic Environmental Effects* to test the integrated prototype in a wide variety of scenarios. These tools can generate high interference and large temperature variations that test the protocols' dependability in extreme conditions. To better understand the environments in which the use cases are expected to be deployed, we collect real-world traces of temperature and interference in the Barcelona facility and in SmartSantander—a FIRE testbed facility in Santander, Spain. We use these traces to augment the TWIST FIRE testbed with interference generation using the JamLab tool stemming from Task 4.1. In summary, the two main points of the work carried out as part of the deliverable are the following.

Using controllable and repeatable patterns of temperature variations and interference, we evaluate four protocols as part of the integrated prototype: MiCMAC, Evergreen, RPL++, and Temperature-Aware MAC. The first two strive to attain dependable performance under interference, whereas the last two do so when challenged by temperature variations. These protocols are specified in Deliverable D-2.1, and are in this deliverable integrated into a system that is deployed in different testbeds, including the FIRE testbed TWIST, hosted by Technical University Berlin.

The results of the integrated experiment show that the new protocols developed in RELYonIT sustain a much higher performance in the presence of environmental impact than the state of the art. Based on the outcome of the first integrated experiment, we summarize the results with respect to the use cases. We show that MiCMAC achieves over 90% packet delivery rate under heavy interference, while maintaining a latency below 10 seconds and a radio duty cycle below 2%. These results fulfil the MUST requirements of the selected use case. When testing the temperature-aware MAC protocol, we have shown that the network sustains up to 42% lower energy consumption and 87% higher packet reception rate in the presence of temperature variations commonly found in outdoor deployments. In the final part of this deliverable, we discuss lessons learnt and what the impact of this will be on future work within RELYonIT.

1 Introduction

This deliverable describes our work on Task 4.3 (*First Integrated Experiment*). In particular, we describe the first integrated prototype, which builds on components developed in WP1 and WP2, and we evaluate it within the use case selected in Task 4.2: WOS' outdoor parking management (SmartParking). This use case was identified as the most suitable one for the integrated experiment (for details, see D-4.1) for two main reasons. First, it has an immediate time to market and can potentially grow the company's business to new countries with extreme weather conditions. Second, WOS has been working on parking management solutions for years, with easy access to two deployment sites in the city of Barcelona, Spain, that significantly speed-up experimentation.

The *SmartParking* use case (as well as the *Civil Infrastructure Monitoring*, *Condition-based Maintenance*, and *Ventilation on Demand* ones used by the industrial partners) requires that the integrated system provides dependable performance despite operating in environments where a multitude of different adverse conditions may occur, particularly due to interference and temperature variability. Whilst the integrated prototype is implemented to mimic predominantly the traffic patterns and data packet contents generated by a parking management application, we have designed a general system that can handle a wide variety of traffic patterns, including those we envision from the other use cases proposed by the industrial partners. The main contributions of the work conducted in this task are summarized as follows.

Integrated Prototype We describe the design and implementation of the integrated prototype, which is a fully deployable system consisting of the protocols developed or adapted in Task 2.1, the learning of environmental and platform models from Task 1.1-1.3, and run-time assurance of such models from Task 1.4. The protocols under consideration in this document are MiCMAC, a multi-channel MAC protocol design to be robust to interference; RPL++, a temperature-aware version of RPL; a temperature-aware MAC protocol; and Evergreen, a new routing protocol for data collection that improves the robustness to interference.

Integrated Experiment We conduct an experimental evaluation of the integrated prototype in four different testbeds. The integrated experiment uses traces collected in the FIRE testbed SmartSantander, and in the SmartParking facility in Barcelona. We use tools developed in Task 4.2 *Testbeds with Realistic Environmental Effects* to study the effects of interference and temperature in a repeatable manner. We analyze the results of this experiment with respect to the selected use case, and draw conclusions that will provide the foundation for the next steps in the project.

The primary goal of the experimental evaluation is to test the hypothesis that it is possible to *reduce the impact of the environment on network performance* by i) modelling the environment of the network, ii) including the model in newly designed or optimized protocols, and iii)

enabling the protocols either to adapt to environmental variations or to take precautionary measures against them. Our experimental results have proven that this hypothesis holds true; we have successfully mitigated the adverse effects of temperature variations and interference both at the network layer and the link layer. We have analyzed the results with respect to the selected SmartParking use case, and found that the MUST requirements are largely fulfilled, but also identified areas that can be improved. The lessons learnt in the first integrated experiment will serve as input to our next iteration on the work in WP1-3.

This deliverable is structured as follows. Chapter 2 describes the design and implementation of the first integrated prototype. Chapter 3 describes the first integrated experiment and presents our results. Chapter 4 analyzes the results of the integrated experiment in relation to the use cases provided by the industry partners. Chapter 5 describes the lessons learnt from this work, and derives strategies for refining our earlier approaches in the remainder of the project. Chapter 6 concludes this deliverable.

2 Integrated Prototype

The integrated prototype is an implementation of the protocols, environmental and platform models, and run-time assurance developed in WP1-3. The integrated prototype is predominantly built using the Contiki operating system's low-power IPv6 architecture. By using this architecture as the foundation of our work, we leverage upon a standards-based, open-source environment that is well-tested within the Internet of Things and sensor networking communities. The protocols developed in Task 2.2 *Protocol Design* replace or augment existing components at various layers of the IPv6 communication stack.

At the network layer, we modify RPL to be temperature-aware by disseminating temperature and platform models, and using this information to calculate the worst-case link attenuation and blacklist poor links. As we will show in the integrated experiment, however, we were not able to gain conclusive evidence that this method improves the network performance. We have analyzed why our initial hypothesis of this protocol was incorrect, and instead leverage this work in the design and implementation of the Temperature-Aware MAC protocol (TempMAC). We present early experiments of the Evergreen data collection protocol, which is a separate component implemented on top of TinyOS that mitigates the effects of interference.

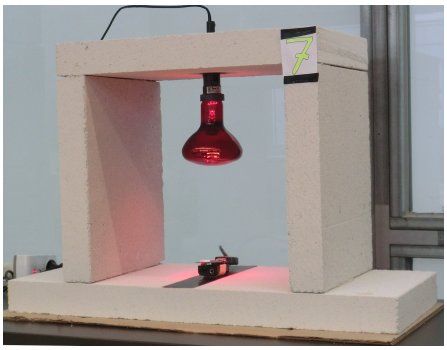
At the link-layer, we use MiCMAC, a multi-channel MAC protocol that effectively mitigates interference. MiCMAC is based on ContikiMAC, Contiki's main sender-initiated MAC protocol. We also improve upon the energy detection used in CSMA-based protocols with a Temperature-Aware MAC protocol. TempMAC improves the network performance when the network is challenged by temperature variations. This mechanism operates in conjunction with ContikiMAC in our integrated experiment, but it can just as well be applied to MiCMAC.

The aforementioned protocols are integrated with run-time assurance and the learning of environmental models and parameters developed in WP1. We collect traces from the FIRE testbed facility SmartSantander and the SmartParking testbed in Barcelona. These traces will be used in the first integrated experiment. The two tools developed in Task 4.1, JamLab and TempLab, can replay interference and temperature traces in a controllable and repeatable manner. We will augment another FIRE testbed, TWIST, with JamLab nodes to test the protocols' performance under interference. The temperature traces will be replayed at testbeds hosted by project partners TUG and ULANC. The learning of models and parameters is based on data extracted from such traces, and our run-time assurance component warns during run-time if the models and parameters do not hold true based on the latest sensor samples.

2.1 Testbeds with Realistic Environmental Effects

To understand how the environment affects the performance and the operation of IoT protocols it is fundamental to be able to rerun experiments under identical environmental conditions. Within RELYonIT, we consider as primary environmental factors temperature and radio interference. In Deliverable 4.2 (*Prototype of Testbeds with Realistic Environmental Effects*), we

(a) Overview of our testbed infrastructure



(b) IR heating lamp on top of a sensor node

Figure 2.1: TempLab uses infra-red heating lamps on top of each sensor node to accurately control their on-board temperature [9].

have presented the design and implementation of two IoT testbed extensions that enable the repeatable playback of environmental conditions: TempLab and JamLab. In the first integrated experiment, we use these tools to study the effects of interference and temperature in a repeatable manner on the protocols we have developed in the project. TempLab and JamLab can replay temperature and interference traces that we collect from experimental outdoors facilities, including the FIRE testbed SmartSantander.

2.1.1 Replay of Real-World Temperature Traces

To better study the impact of temperature variations on low-power wireless communications and protocols, we have designed TempLab [9], a testbed infrastructure with the ability of varying the on-board temperature of sensor nodes and reproducing the temperature fluctuations that can be normally found in outdoor deployments. We have built two versions of TempLab within the Consortium: one at TU Graz and one at Lancaster University for experimenting with temperature. Indeed, we could not simply use existing testbed facilities, but it was instead necessary to have physical access to an indoor testbed with hardware extensions to playback environmental conditions. Figure 2.1 shows an overview of the TU Graz facility, with infra-red heating lamps on top of each sensor node to control their on-board temperature.

Reproduction of temperature profiles. In order to support a wide range of experimentation techniques, TempLab can generate temperature profiles using three different approaches. Firstly, one can re-play temperature traces collected in-situ at a given deployment site. Such *trace-based* temperature profile instantiation can accurately reflect the temperature variations over time with fine granularity if long-term measurements from one or more nodes are available. Given that traces are not always at one's disposal, a second possibility is to use a *model-based* temperature profile to have an estimation about the temperature dynamics at a certain location without the need of traces collected in-situ. A model-based approach uses models to estimate the temperature profile of objects using basic environmental information such as the maximum solar radiation and the minimum temperature during a day (that is readily available

from satellites and meteorological stations). A third possibility is to use TempLab to vary the temperature of sensor nodes using specific *test patterns*. For example, a user may not be interested in recreating a specific profile and needs instead only to verify whether a high temperature variation has an impact on the operation of a given protocol. In this case, TempLab can be fed with on-off patterns (e.g., a series of cold and warm periods) or jig-saw patterns that vary temperature with a specified frequency, allowing a quick debugging of protocols' behaviour.

Time-lapsing of traces. TempLab also offers the possibility of time-lapsing an experiment: indeed it is often desirable to compress the time scale of an experiment to save evaluation time (one may want to time-lapse the recreation of real-world traces and playback, for instance, in a few hours the profile of a full day).

Replay of real-world traces

As trace-based temperature accurately reflect the temperature variations over time at a given location and hence enhance the realism of experiments, we have collected and replayed a number of traces from real-world deployments and FIRE testbeds. First, we have parsed one year of traces in Wennerström et al.'s outdoor deployment in Uppsala, Sweden [22]. In the latter, 16 TelosB nodes equipped with the CC2420 radio were placed within each other's transmission range, and exchanged packets and recorded statistics for more than one year.

Figure 2.2 shows the ability of TempLab to accurately reproduce the original trace (full results available in [9]). It is important to highlight that the higher the time-lapsing factor T_F, the higher will be the inaccuracy of the replay. In this specific case, we have replayed a trace taken during August at the original speed, and with different compression factors up to $T_F = 10$. We can notice that the average replay error remains below 0.5°C when $T_F < 5$, showing a good capability of our infrastructure to quickly cool-down and heat-up the sensor nodes.

We also collect several traces taken from the FIRE testbed facility in Santander, Spain. In particular, we connect to `www.smartsantander.eu/map` and collect temperature readings from nodes on the main road in Santander on the waterfront (near the parking spaces, see Figure 2.3)[1].

We first take traces for different seasons and time-lapse these traces by a factor $T_F = 20$, i.e., we reproduce the temperature fluctuations that would occur in reality within 48 hours in 150 minutes only. This puts the testbed capabilities of quickly heating up and cooling down nodes through the wringer, but the average error below 0.75°C hints that the duration of the experiments can be significantly shortened without compromising accuracy. Figure 2.4a replays traces collected during winter and spring. The top figure shows the replay of a trace collected in Santander between December, 7 and 8. These temperature fluctuations were observed across several nodes during winter-time. The bottom figure shows the replay of a trace collected between April, 12 and 13, with spring temperatures. The average error of the replays is $\approx$ 0.5°C.

[1] Although the large number of nodes deployed, the main limitation in the data available from Santander is that many of the sensors have periodic outages where no data is collected/recorded, as well as the time reference are often not coherent. This makes it hard to extract a long trace of multiple distributed sensors where all sensors continuously provide data. Furthermore, a description of the location of the sensors (indoor, outdoors) and its packaging is missing, making it hard to give a proper meaning to the available traces.

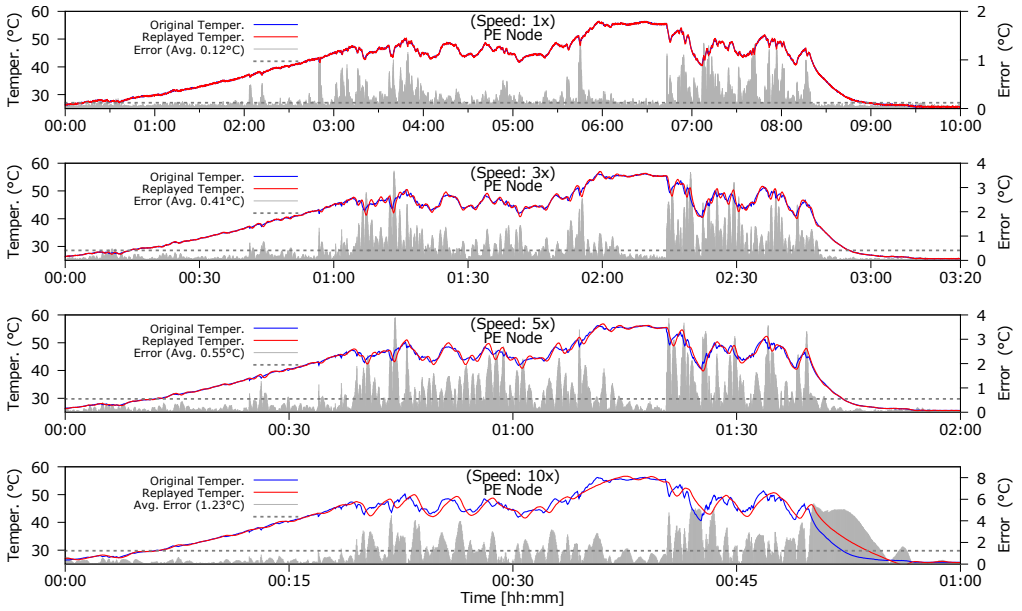
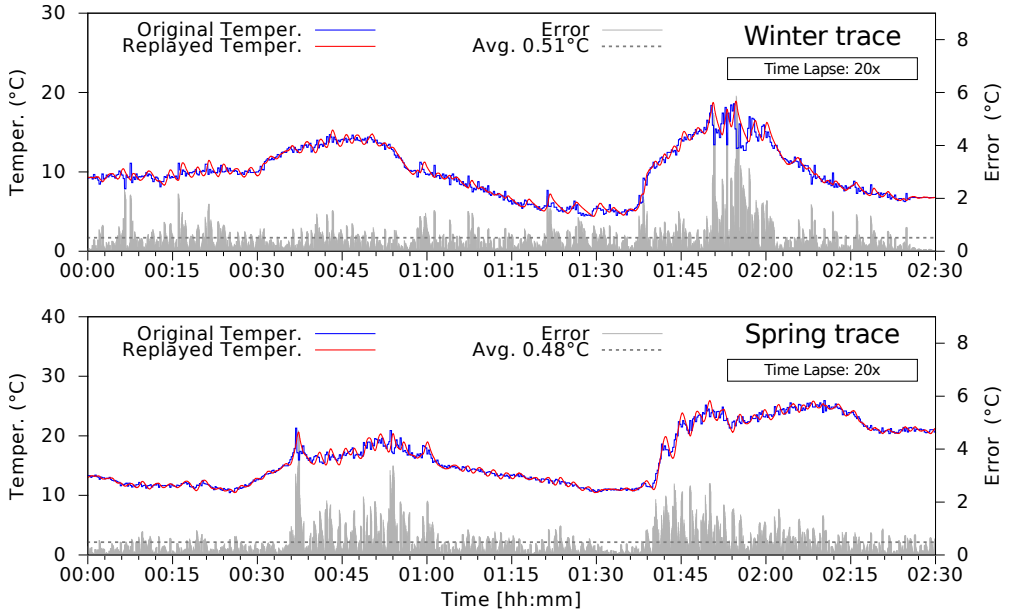


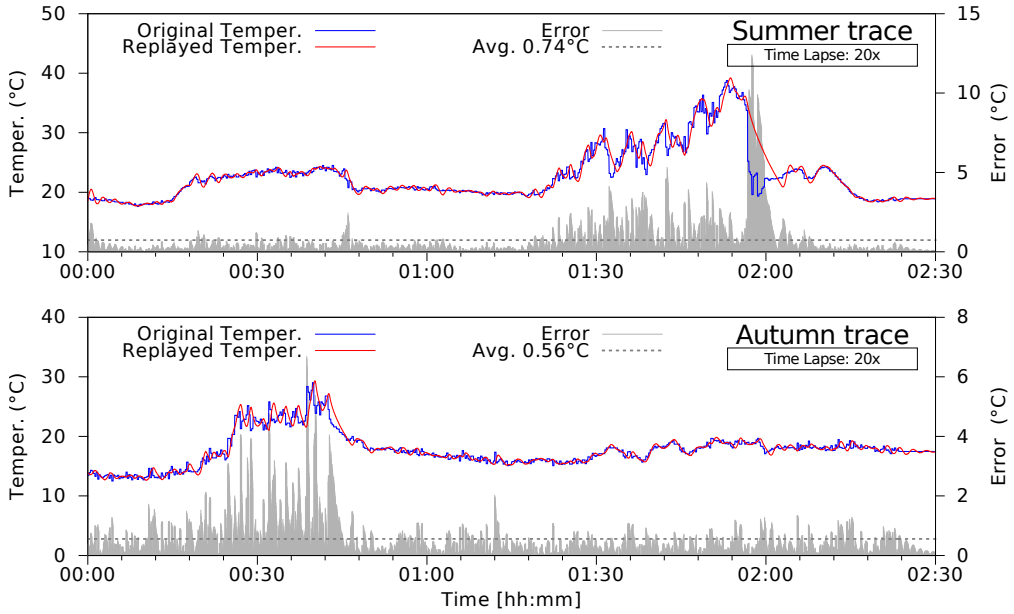
Figure 2.2: Replay of real-world outdoor temperature traces using TempLab [9].

Figure 2.3: Map of the city of Santander, with the location of the nodes from which we retrieved the temperature data to be replayed (`www.smartsantander.eu/map`).

Figure 2.4b replays traces collected during summer and autumn. The top figure shows the replay of a trace collected in Santander between June, 26 and 27. These temperature fluctuations were observed across most of the nodes during summer-time. The bottom figure shows the replay of a trace collected between October, 15 and 16, with colder temperatures. The average error of the replay varies between ≈ 0.56 and 0.74°C. One of the main contributions



(a) Replay of winter and spring traces



(b) Replay of summer and autumn traces

Figure 2.4: Replay of temperature traces collected in Santander using TempLab [9] with a time-lapse factor of 20.

to the inaccuracy of the replay is that the granularity of the traces collected in Santander is one temperature value every 5 minutes. As temperature can significantly vary in this timeframe, the testbed controller has to suddenly trigger a temperature variation that may be up to 5-6°C:

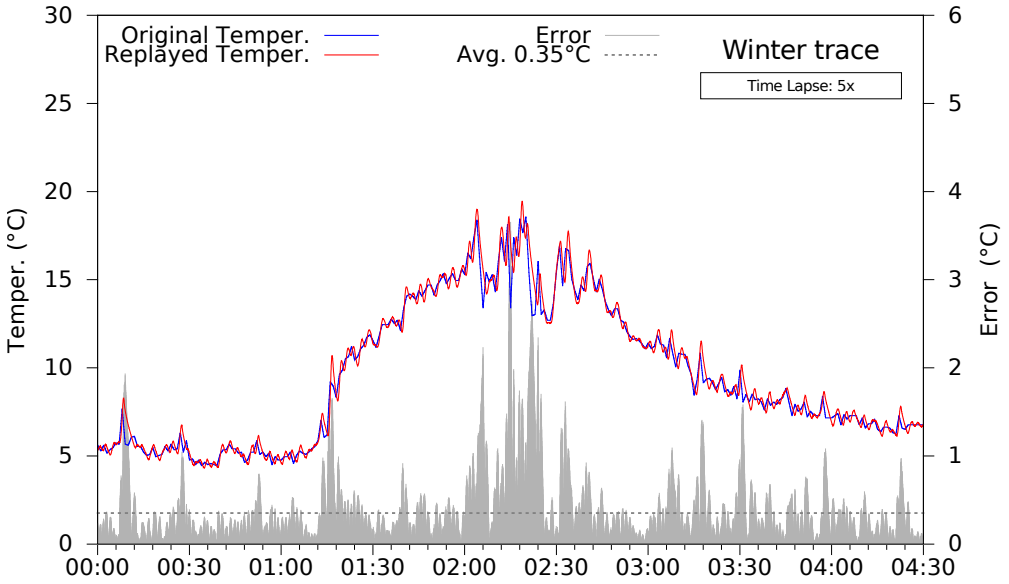


Figure 2.5: Replay of temperature traces collected during winter in Santander using TempLab [9]. In this experiment, we have used a time-lapse factor of 5 only, leading to a higher accuracy of the replay.

in a time-lapsed trace with a $T_F = 20$, this leads to a situation in which the actuators cannot warm-up or cool-down the nodes that fast.

However, when lowering T_F, the accuracy can be increased. Figure 2.5 shows that the error drops down to 0.35°C when decreasing T_F to 5.

The role of the enclosure. Please note that the traces taken from Santander show significantly lower temperature fluctuations over the day compared to the ones from the outdoor deployment in Uppsala. This is counter-intuitive, as the temperatures in Sweden across the year should be colder than the ones in Spain. The reason is that in Uppsala nodes are enclosed in casing and are exposed to direct sunlight, and the measured temperature is the *on-board* temperature. In Santander, the temperature sensor does not refer to the actual on-board temperature, but rather measures the air temperature and may be unsuitable for assessing the impact of temperature on communication protocols, as it does not map the actual temperature of the radio chip. Real-world deployments have shown that the on-board temperature of wireless sensor nodes deployed outdoors can indeed be significantly higher than air temperatures measured by traditional weather stations [21]. Sensor nodes are indeed often exposed to direct sunlight and embedded into airtight packaging absorbing IR-radiation [4], causing the inner temperature in the casing to reach values as high as 70°C [5]. In a long-term outdoor deployment, Wennerström et al. [22] have observed that the on-board temperature of a sensor node enclosed into an airtight packaging can experience variations up to 83°C across different seasons, and 56°C within 24-hours [7], with large heterogeneity across the network [9].

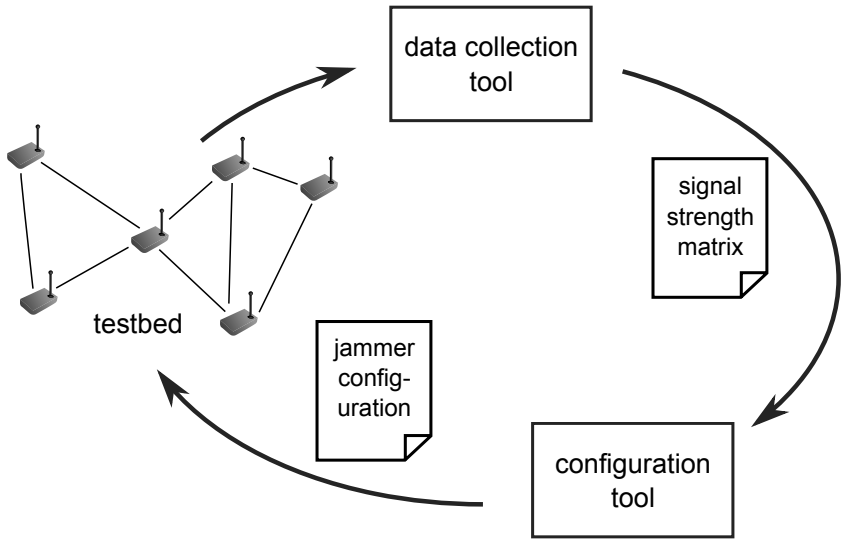


Figure 2.6: Tool-chain to support jammer configuration process.

2.1.2 Emulation of Interference

Another environment effect that significantly affects IoT systems is radio interference. This is especially true for indoor deployments and urban environments. Consequently, we also need to be able to emulate interference effects in testbeds in a similar manner as it is possible for temperature. Current IoT testbeds do not possess such capabilities. To augment existing testbeds with an emulation of radio interference effects, we employ an extended version of JamLab [6], a flexible low-cost testbed infrastructure that allows the repeatable generation of a wide range of interference patterns. The system has been integrated with existing testbeds—such as the FIRE facility TKN Wireless Indoor Sensor network Testbed (TWIST) at TU Berlin—without a need for additional hardware or significant modifications. Instead of adding additional devices to recreate interference effects, a subset of the already deployed nodes is used. These nodes use a special software that allows them to act as jammers and interfere with the communication of other nodes. As assessed in earlier work by Carlo Boano [6], this is sufficient to emulate typical interference sources, like WLAN, Bluetooth and microwave ovens.

To make the system work reliably, the nodes that operate as jammer need to be carefully selected, such that each remaining node is covered by at least one jammer. The signal from this jammer needs to be stronger than any other signal in order to ensure an effective jamming. Of course, we want to reduce the number of jammers as much as possible, to have a sufficient set of nodes available for the actual experimentation.

In the original work, a manual process was employed to select jammers [6][2]. For the RELYonIT project, we developed a tool-chain to semi-automatically derive a suitable configuration for a testbed as illustrated in Figure 2.6. Such a configuration is generated in a three step process:

[2]Please note, that in contrast to our terminology, the original paper refers to interference generating nodes as "HandyMotes".

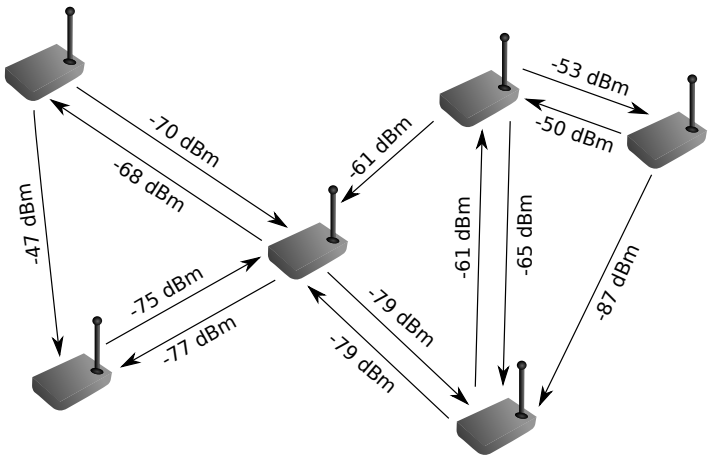


Figure 2.7: Signal strengths of an exemplary testbed.

1. The signal strength of all potential links at all output power levels is recorded.
2. This signal strength data is employed to determine an optimized selection of jammers and suitable output power level for the jammers.
3. The jamming software is deployed on the previously selected nodes.

Afterwards, the remaining nodes can be used for experimentation.

The first tool builds a connectivity matrix with signal strength readings for each possible link between a pair of nodes (see Figure 2.7). The tool consists of a program that is deployed on the nodes of the testbed and a central data collection application. During data collection, the nodes sequentially send a broadcast message, including the employed power level and node ID, at each available output power level. All other nodes monitor the radio and record the signal strength and the transmitted information for each message they receive. The collected data is integrated into a single signal strength matrix by the detached data collection application. The node component of this tool was implemented for Contiki and TinyOS to support a wide range of different environments.

The generated signal strength matrix is in turn used by a separate Python program running on a more powerful machine to derive a suitable configuration, by employing an optimization strategy. This tool generates a selection of jammers and determines a suitable output power setting for each of these jammers. This configuration is written to a file and can be subsequently employed to deploy the actual jamming software on the selected nodes. The latter step is currently not fully automated due to heterogeneous programming interfaces at the different FIRE facilities. The JamLab jamming software itself could be efficiently used in the assessed testbeds and did not require any modifications.

The full jammer configuration tool-chain was employed as part of the first integrated prototype to support the testbed experiments described in Chapter 3. Most notably, it was used to assist with the generation of suitable jammer selections for the local testbed at the Graz University of Technology and in the FIRE facility TWIST at TU Berlin.

2.2 Selected Use Case

The integrated prototype is a system designed to adhere to the requirements of the candidate use cases detailed in Deliverable D-4.1. The four use cases that we consider within the RELYonIT project are the following:

1. Outdoor Parking Management,
2. Civil Infrastructure Monitoring,
3. Condition Based Maintenance,
4. Ventilation on Demand.

The first three use cases are inherently based on sense-only data collection, whereas the last one requires a more complex feedback loop in which the sensor nodes must act as actuators as well. The integrated prototype embodies protocols that are designed to handle both sense-only and control-loop use cases. These protocols are designed to provide dependable performance when challenged by interference and harsh environmental conditions.

We perform the integrated experiment within the context of the Parking Management use case which been selected for implementation in Task 2.4 as reported in D-4.1. Project partner Worldsensing has provided access to a test deployment in the streets of Barcelona, and a graphical user interface to view parking slot occupation. The integrated prototype is designed so that it can emulate the traffic pattern of a parking management application, which is similar to a typical data collection sensor network. Each node sends its latest sensor samples as needed by the application—either periodically or just when a sensor event occurs, such as a parking slot occupancy change is detected. In addition, these data packets that are relayed to the sink will include network statistics kept by the node, which helps to understand the performance of the network at little extra cost.

In the integrated experiment, we investigate how our protocols perform in a variety of testbeds. We test the performance of our protocols when subjected to severe interference and drastic temperature changes, using tools developed in Task 4.1 *Testbeds with Realistic Environmental Effects*. In Chapter 4, the outcome of this experiment is analyzed within the context of the use case requirements.

2.3 Learning of Environmental and Platform Models

Generating environmental and platform models is a key component of the integrated prototype. Understanding and modelling how the deployment environment changes over time and its effect on the deployed platform provides much of the necessary information to develop dependable IoT solutions. Two environmental aspects that were found during our work in task 1.1 to have the most prolific effect on IoT systems were temperature and radio interference. The temperature of an enclosed system exposed to direct sun light can vary by as much as 56°C over the course of a single day [7]. This changing temperature can have significant effects on a platforms electronics particular communication components. Furthermore, IoT systems are typically deployed in environments with radio interference, a problem which is ever increasing

as more systems a deployed. Radio interference can cause packet corruption which leads to unreliable service and increased energy consumption as transmissions are necessary.

Within Tasks 1.1 and 1.2 and presented in D-1.1, environmental and platform models targeting these two aspects where developed and will be instantiated during this first integrated experiment. For temperature we will model using minimum and maximum readings and for interference we will examine idle and busy periods through the use of idle and busy CDF model with both models described in D-1.1. As described in D-1.2, to realise these models, data needs to be collected from the deployed environment and analysed to parameterize the models for the specific environments and platforms. For temperature, a trace of regular temperature samples is required over the course of a cycle/day in the deployment site. For interference, the number and length of idle and busy periods are needed, measured at a sufficient frequency and duration to accurately characterise the environment.

A single tool, an extension of those presented in deliverable D-1.2, was developed to collect the necessary data to parameterise both models as a prototype to be used for the first integrated experiment. The tool was evaluated at the selected use case which consisted of a parking lot in the @22Barcelona innovation district. Unlike protocols that can be tested in environments that simulate that of the user case, it was necessary to evaluate the data collection tool in the real deployment to collect real-world data. The results collected by this tool are presented and analysed in section 3.2.

The developed tool which is in the form of a Contiki OS application was designed to run prior to a deployment to collect data to enable the instantiation of both the temperature and interference models. It is designed to run on the deployed application platform and as such have no specialist hardware requirements. It should run for a sufficient period to accurately capture all variances. For temperature this should be at least one full cycle (usually a day) which should capture the peak temperature during the day and minimum during the night. For interference this should be long enough to capture the most intense periods of activity (usual during the working day). The tool should also collect data at a sufficient number of locations to cover the entire deployment site. For the first integrated experiment, the application ran on the Maxfor MTM- CM5000MSP Telos-B clones at four locations for a period of 48 hours in the middle of the working week.

During its execution, the application kept time synchronisation between each node to ensure measurements were taken at the same time. A single mote was devoted to this task and regularly sent out time synchronisation beacons that were received and flooded by the other motes on an ad-hoc basis when the mote was not recording measurements. Time synchronisation messages were sent at the maximum transmission power on channel 11. This and channel 12 was then excluded from any measurement to reduce the chance of interference being caused by the tool.

During each hour, each mote would measure the distribution of idle and busy periods on a series of channels as well as recording the on-board temperature. With regards to interference, for each channel a three minute measurement was taken which consisted of sampling the RSSI of the radio every 24 µs and comparing it with a specific threshold (-80 dBm) to then determine if the channel is idle or busy. The length of each idle and busy period (the number of consecutive samples) was then recorded in the form of an idle and busy period distribution. These distributions were time-stamped and written to flash memory. For temperature, the on-board temperature of the node was sampled four times per hour and the current, minimum and maximum temperature was written to flash with time-stamp. Sufficient time was reserved

at the end of each hour to ensure a high probability that a time synchronisation packet would be received and then forwarded to other nodes.

A the end of the measurement campaign the flash of each node was read and the data was analysed to enable the temperature and radio interference models presented in D-1.1 to be parameterised. The results of the data collected at Barcelona for the first integrated experiment is presented in section 3.2.

2.4 Run-time Assurance of Environmental Models

In the previous section we discussed how data will be collected in the integrated experiment to instantiate environmental and platform models. Such models will later be used with protocol models in WP3 to select the appropriate configuration to give the desired dependable performance. This performance can only be assured whilst the environmental aspects stay within the predictions of the created models. If an environmental aspect such as temperature or radio interference changes beyond the predictions of the environmental model then the application may fail without reassessment of the model and reconfiguration of the selected protocols. It is important to continually monitor the environment for violations of the model during application operation to trigger such reassessment, which is the task of run-time assurance.

Within Task 1.3 methods for monitoring the environment during application operations are being developed which will be documented in deliverable D-1.3. These methods will concentrate on measuring the two selected environmental aspects of temperature and radio interference. For the first integrated experiment, the run-time assurance module will only examine temperature. The module is in the form of a software component that will run alongside the application. It will monitor temperature which has been sampled and recorded as part of the core application for violations to the minimum and maximum temperature model derived from the pre-deployment data. When violations are detected, a fault will be raised which later in the project will be used to potentially trigger a reconfiguration.

2.5 Protocols

The integrated prototype includes four newly designed or adapted protocols: MiCMAC, Evergreen, RPL++, and TempMAC. These protocols are predominantly built by using the Contiki operating system's low-power IPv6 stack as the base. An exception is the Evergreen data collection protocol, which is implemented in TinyOS. Based on the requirements of the use-case scenarios, we designed these protocols to mitigate environmental conditions, such as temperature variations and interference, to provide more dependable performance. In the case of RPL++ and TempMAC, we use the environmental models and parameters detailed in D-1.2. We modify the baseline network stack at both the link layer and the network layer. The design and implementation of these protocols are presented in D-2.1. In the integrated experiment, we will validate the suitability of our new protocols, following a two-step approach. First, we run a trial *in situ* in the parking deployment to obtain a performance baseline for an unmodified network stack and to obtain an estimate of the environmental conditions. Second, we design testbed experiments that emulate the conditions expected in this and other real deployments. The protocols operate in conjunction with the separate run-time assurance module,

which monitors whether the used models and parameters are valid during operation.

Overcoming temperature effects with RPL++ and TempMAC. The car parking scenario has the salient feature of being deployed outdoors and to be located inside the tarmac. Under these conditions, nodes can be exposed to high temperatures. To overcome potentially pernicious temperature effects, we first tried to optimize the standard RPL protocol—we called this protocol RPL++. Our optimizations were focused on the Routing Layer and we found little improvement. As a result of our findings, we then developed TempMAC, a new protocol that adjusts the CCA according to temperature effects.

Using a multi-channel MAC protocol to mitigate interference. In the use case of the Smart Parking application where the system is deployed in an urban environment, channel interference is a given fact. To assess this we performed measurements in Barcelona using nodes in the tarmac and overground, during different times of the day, where we detected 38 networks across the 2.4 GHz band. We saw that both location and time affect the level of interference greatly. In order to overcome the effects of interference, we designed the MiCMAC protocol, that is a channel-hopping variant of the ContikiMAC.

The Evergreen approach to route packets under interference and temperature variations. The aforementioned work provides very good results but it is suitable for either interference or temperature variations. An application, such as smart parking, could encounter interference and temperature variations at the same time. Hence, we designed a new protocol, Evergreen, that tolerates both phenomena.

3 Integrated Experiment

The integrated experiment evaluates the protocols of the integrated prototype in several testbeds, which are augmented with the tools developed in Task 4.1 *Testbeds with Realistic Environmental Effects*. One of these testbeds is *TempLab*, provided by project partner TU Graz. Another of the testbeds is the FIRE facility *TWIST*, which is hosted by TU Berlin, and is open to the research community for experimentation. Furthermore, we collect interference and temperature traces from an outdoor SmartParking testbed in Barcelona, which is hosted by project partner Worldsensing, and from the FIRE facility SmartSantander. These traces can be replayed in other testbeds by using the JamLab and TempLab tools developed in Task 4.1. The Barcelona deployment also serves as a baseline for assessing how an unmodified RPL performs in the SmartParking scenario.

Our experimental evaluation of the integrated prototype is conducted in these testbeds to allow us to improve upon the design and implementation of our protocols from WP1-3. The lessons learnt will be highly useful for the section iteration of our work, and the final system will be tested in Task 4.4 *Second Integrated Experiment*.

We begin this evaluation by describing the relevant metrics for dependable performance that we study, and the hypothesis that forms the foundation of our work. We then set out to experimentally evaluate the protocols and modules comprising the integrated prototype in the aforementioned testbeds. In the next chapter, the results of this experiment will be analyzed within the context of the selected use case provided by the industrial partners of the consortium.

3.1 Methodology

The integrated experiment is conducted in a set of testbed facilities, where we test each protocol individually. All the resulting statistics are collected through periodic transmissions from each node to a designated sink node. Energy measurements are carried out using software-based power profiling, in which the time that a device is in a certain state is recorded, and multiplied by the current consumption for that state [13].

The objective of the integrated experiment is to test the hypothesis that one can *reduce the impact of the environment on network performance* by

1. Modelling the environment of the network.
2. Including the model in newly designed or optimized protocols.
3. Enabling the protocols either to adapt to environmental variations or to take precautionary measures against them.

We quantify the impact of the environment by seeing how a set of key performance metrics are affected as the environment changes. Our goal is to improve the dependability of the

protocols by avoiding considerable performance degradation when challenged by harsh environments. Beyond directly improving the operation of deployed industrial applications, this type of protocol improvements makes the network easier to model as well—both for the purpose of doing optimizations and to give probabilistic bounds for worst-case performance.

3.1.1 Metrics for Dependable Performance

The four main metrics considered in this experiment are latency, energy consumption, packet loss rate, and network stability. We give a detailed description of each metric below.

Latency The end-to-end latency is measured between the point a packet transmission is initiated at a sensor node and the point where it has been received at a data sink. Since we do not include time synchronization in the integrated prototype, our measurement of latency is typically limited to time-stamping of messages on the serial bus of the devices, and is therefore not highly accurate at a millisecond level. The errors that can be expected are, however, insignificant compared to the latency requirements of the use cases, which are on the scale of seconds or higher.

Radio duty cycle The radio duty cycle is defined as the percentage of time that the radio is in listen mode. It is well-established in the literature that the radio is by far the largest energy consumer in a sensor network, so this metric is a proxy for the energy consumption of the integrated prototype. In all our test scenarios, other system components such as the sensors and the micro-controller use a negligible part of the total energy.

Packet loss rate The packet loss rate measures the packet loss observed for end-to-end, network-layer traffic. Hence, packet losses that occur at the link layer, are not counted if a retransmission of the packet succeeds. Link-layer packet losses are accounted for indirectly through increasing link metrics (causing network instability) and added energy consumption.

Network stability The network stability is typically measured by looking at the link churn, or the number of routing parent switches observed over time. For modeling purposes, and to attain more dependable performance, it is desirable to keep the network highly stable, while still preserving some agility to react to dynamic conditions that occur within the network due to stochastic external factors such as interference.

3.2 Barcelona Deployment

The purpose of the Barcelona deployment is to establish a baseline for our testbed experiments, i.e., to measure the environmental conditions found there and to examine the performance of an unmodified network stack. The Barcelona testbed relies on an existing setup in the well-known Smart City testbed in the 22@Barcelona innovation district [3]. The current testbed contains about 30 sensor spots in different locations, among them one street and three street corners with two of the corners being short-time parking zones for loading/unloading allowing a high rotation (i.e. parking state changes) for the tests. The battery-powered sensor motes are buried in the road's tarmac and they are easily accessible and interchangeable for testing purposes.

For the tests in the RELYonIT project, one of the load/unload zone corners has been selected. This corner has 5 sensor spots and they are close to a base station to provide full coverage in each test.

The motes used in the Barcelona deployment are T-Motes with a 2.4 GHz radio chip (TI CC2420) [1] running the Contiki operating system. The motes are battery-powered and encased inside a FastPrk standard box as shown in Figure 3.1 and Figure 3.2. We also can use USB-powered motes in the same box as seen in Figure 3.3. Once buried in the tarmac with the box closed, the entire system looks like Figure 3.4. All the experiments have been done with the box completely closed as depicted in the figure.

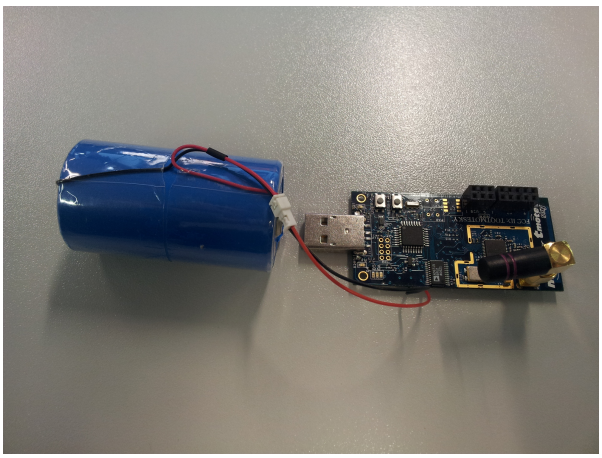


Figure 3.1: A T-mote equipped with a battery.

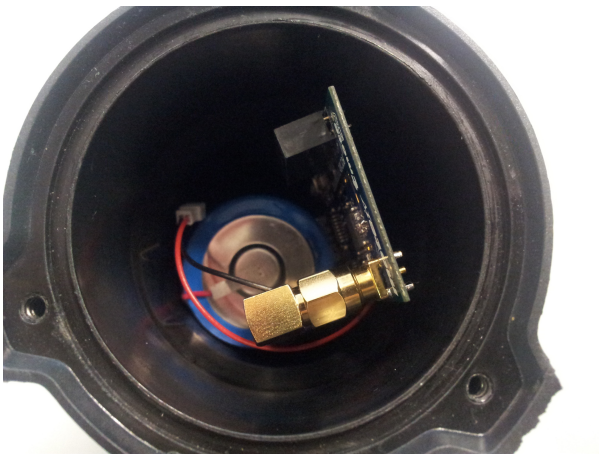


Figure 3.2: A T-mote placed inside a standard FastPrk box.

Figure 3.3: A T-mote buried in the tarmac with a debug USB.

Figure 3.4: A closed box buried in the tarmac.

3.2.1 Temperature

The on-board temperature of each of the motes in the Barcelona deployment was recorded over a period of 45 hours. Figure 3.5 presents the hourly averaged temperature recorded at mote 203 and mote 253. The two motes were located within the tarmac of two car parking spaces. The temperature recorded at both motes follow the same trends, with mote 203 recording slightly more extreme temperatures than temperatures recorded at mote 253. During the experiment the minimum non-averaged temperature observed by mote 203 was 15℃ whilst the maximum temperature was 30℃, giving a temperature variance of 15℃. For mote 253, the minimum observed temperature was 16℃ and the maximum was 27℃, giving a variance of 11℃. Over the 45 hour period we see the expected cycling of temperature between night and day. Hence, even across such short time period a significant temperature variation has been observed. Even higher variations are to be expected over a whole year. This proves that there is a need to make protocols aware of such temperature variations, and motivates the development of the RPL++

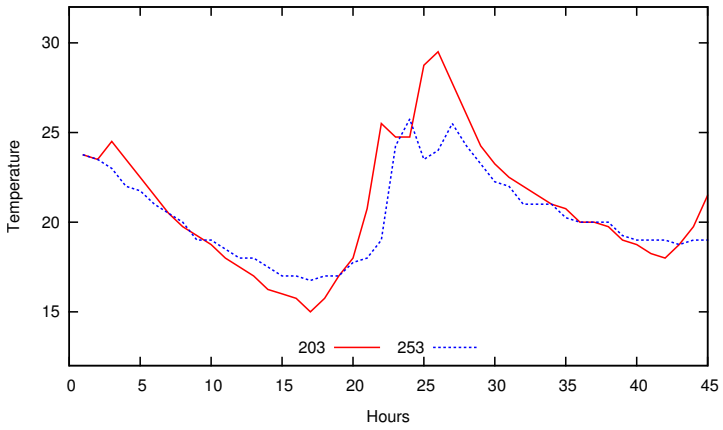


Figure 3.5: Hourly averaged temperature for motes 203 and 253 over 45 hours.

and Temperature-Aware MAC protocols.

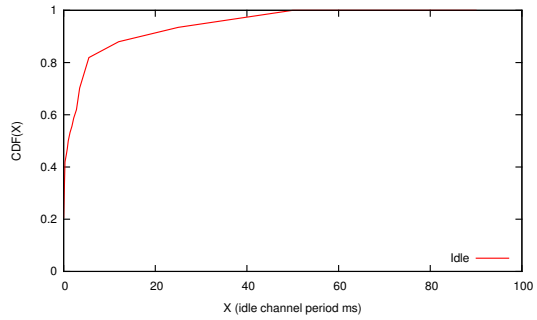
3.2.2 Interference

Within the Barcelona deployment four motes were deployed to collect the necessary pre-deployment interference data using the collection tool (described in section 2.3) to instantiate the environmental model. Mote 187, 253 and 203 were positioned in the tarmac in three car parking spaces whilst mote 161 was placed four meters above the ground on a pole at the base station. Mote 187 was assigned the duties of time synchronisation whilst the other three motes performed data collection.

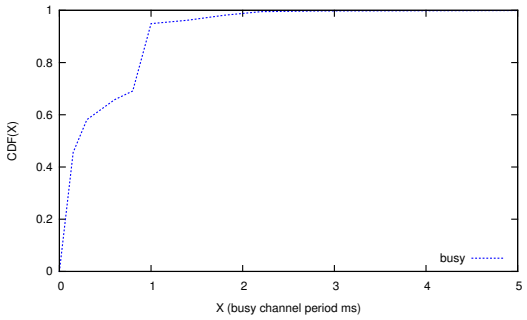
Before the experiment, the Wi-Fi spectrum was surveyed using an off-the-shelve wireless adapter and 38 networks were found across the available 2.4 GHz frequencies. With this level of Wi-Fi saturation, interference is unavoidable, and each of the sampled channels shared frequencies with at least four Wi-Fi networks. IEEE 802.15.4 channel 16 was found to have the highest number of coexisting Wi-Fi networks with sixteen.

Figure 3.6 presents the idle and busy CDF recorded by mote 161 on channel 19. The data represented in this figure consists of a single 3 minute measurement window during which 34491 idle/busy periods were detected. The median of the idle CDF is 1 ms, which implies significant interference was present. Comparing the idle and busy periods, the idle periods are longer. 95% of busy periods were found to be less than 1 ms, whereas 50% of idle periods are below 1 ms with 13% being above 12 ms. For the remainder of this analysis we will focus on examining the idle CDF only of each measurement only; it is assumed that nodes carry out a Clear Channel Assessment (CCA) prior to transmission, and thus transmit only in an idle period, which makes the idle periods of greater significance.

Next we examine how the idle CDF varies from one hour to the next within the Barcelona deployment. Figure 3.7 presents the idle CDF recorded from three separate hours by mote 161 on channel 19. In the first hour at 22:00, this represented what can be considered a typical CDF for that channel found during the experiment. In this hour 50% of the idle periods are

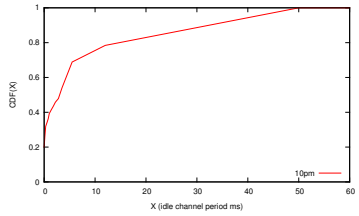


(a) Idle periods.

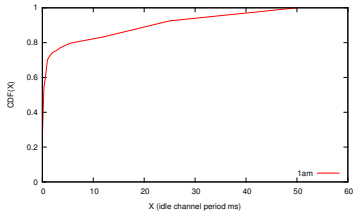


(b) Busy periods.

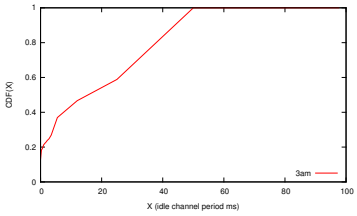
Figure 3.6: CDF of idle and busy periods for channel 19 at midnight.



(a) 22:00



(b) 01:00



(c) 03:00

Figure 3.7: CDF of idle periods for three different hours on channel 19.

less than 2.8 ms. In the second CDF, which was recorded at 01:00, 50% of the idle periods are less than 0.3 ms. This CDF was found to contain the worst-case interference observed during the experiment on channel 19. No analysis was performed to ascertain the reason to why the worst-case interference was observed at 01:00. In the final hour shown 03:00, 50% of the idle periods were found to be less than approximately 20 ms. In this CDF, idle periods are significantly larger than what was found at 01:00, and this CDF was found to have the least amount of interference for channel 19 during the experiment.

Next we examine how the idle CDF differ between different channels due to diverse levels of interference at different frequencies. Figure 3.8 shows an average of the idle CDFs recorded by mote 161 on four separate channels. The measurement of interference on each channel was taken in the same hour back to back. The CDF for each channel are quite different, with channel 13 generally having smaller idle periods overall than the other three channels. However, channel 19 has the higher number of idle periods below 1 ms with 70% compared to 33% for channel 13. It could be argued that channel 19 has the most destructive interference of the two channels as the majority of idle periods are smaller than the time required for the smallest 802.15.4 transmission. Although in channel 19 some idle periods are larger than are seen in channel 13 with 17% of the idle periods being longer than 12 ms compared to just 1%, significantly more idle periods are less than 1 ms. Surprisingly, channel 16, which shared frequencies with the

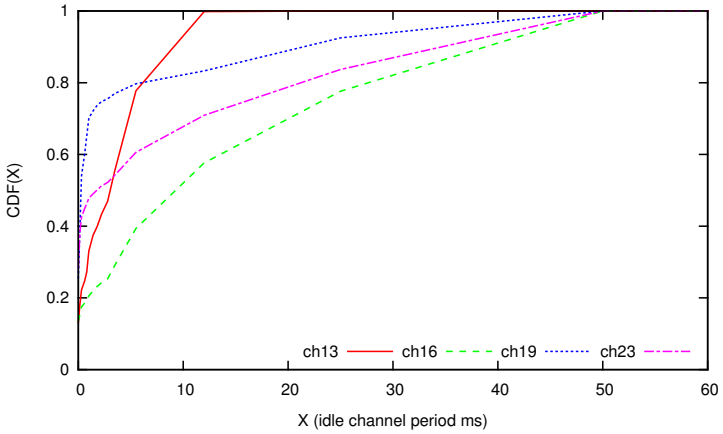


Figure 3.8: Average CDF of idle periods for channels 13, 16, 19, and 23.

highest number of Wi-Fi networks, had the least amount of interference. This could be due higher contention in the Wi-Fi network causing stations to back off, which would create longer than average idle periods.

Next we examine the interference data recorded by the motes within the tarmac. The recorded interference by the two motes was found to be very erratic, which was also seen in each generated idle CDF. This was due to small numbers of detected idle/busy periods in each measurement window. For many of the three minute windows sampled, the busy and idle periods totalled less than 200, which is two orders of magnitude less than what was seen by mote 161 mounted on the pole. With such little data, a statistical analysis using idle CDF model and comparing these to those recorded by mote 161 is not possible. We believe that the tarmac and cars parked above these nodes is attenuating the signal of the interference sources. The nodes are therefore essentially seeing a clear channel with little to no interference.

The vast difference in detected interference between that of the base station mounted in clear air on a pole and that of the tarmac can cause considerable communication issues. The majority of communication in the use case application will be directed to the base station and as such the link involving the base station will be the most heavily saturated. This is also the one that was found to have significant interference. A node transmitting from the tarmac to the base station will detect the channel as clear because no interference is seen and thus will transmit its messages. These messages will have a high probability of suffering corruption due to interference at the base station, which will cause packet loss. On the other hand, messages originating at the base station will be significantly delayed due to high numbers of failed CCA checks and thus message postponement. In this scenario, careful consideration by protocols will be necessary to provide dependable operations.

In summary, this analysis shows that interference recorded in the CDF models can change significantly from one hour to another, between different channels and at different locations. Furthermore, it also shows that there can be considerably different numbers of periods recorded in each sampling window, which can affect the validity of the calculated CDF. These issues need to be taken into account when selecting a CDF—whether an individual CDF or the fusion of

multiple CDF—to best represent the interference of an environment for the purpose of using it in Work Package 3 to help select and configure protocols. This experiment has also revealed that mechanisms to mitigate interference are needed, and this motivates our work on MiCMAC and Evergreen.

3.2.3 RPL Test

We deployed a data collection application in the Barcelona testbed to investigate the baseline performance of an unmodified low-power IPv6 stack, and to find out whether the dependability of the system must be improved to satisfy the use case requirements. To this end, this test collects characteristics of the network topology, and the basic statistics from the routing layer. In order to get a full trace over a day, in which the environmental conditions can vary considerably between day and night, we ran the experiment with 6 nodes for 26 hours. This test provides a baseline as to how an unmodified system operates in an environment that emulates the Outdoor Parking Management use case. In this case, however, the nodes send periodic traffic instead of event-based traffic in order for us to collect fine-grained statistics of how the network performs. The data packets are sent using UDP, and contain sensor values and node-local network statistics.

Table 3.1: Network statistics from the RPL deployment in Barcelona.

Metric	Mean	Min	Max	St.dev
Packets received	2707.8	2584	2888	128.9
Packets lost	166.7	53	278	113.7
Packet loss rate (%)	6.2	1.8	10.4	4.27
RX duty cycle (%)	0.61	0.56	0.71	0.060
TX duty cycle (%)	0.16	0.03	0.51	0.195
Routing metric	583	512	2884	277
Neighbor count	3.9	1	4	0.095
Beacon interval	2001	8	2097	366

Table 3.1 shows the experimental results. The duty cycles for all nodes are below 1%, which is acceptable. However, the maximum packet loss rate was above the MUST requirement for packet loss rate in the Outdoor Parking Management use case: 10.4% versus the required maximum of 10%. The mean packet loss rate was 6.2%, which is slightly over the SHOULD requirement of 5%. This amount of packet loss is a significant issue that we strive to address through our newly developed and optimized protocols.

It remains to be seen whether this is a flaw of the actual placement of nodes (adding more nodes as routing options may solve this problem), or whether it is a malfunction of the link-layer; e.g., the CCA threshold is unsuitable for the node, and this may results in a large number of false wake-ups that increase the duty cycle. Our current work on Temperature-Aware MAC may provide a solution to this problem. The other metrics—i.e., the routing metric, neighbour count, and beacon interval—do not reveal any inherent performance problems of the deployment. The routing metric is the average RPL rank of each node (low is better), and the beacon interval is the average routing beacon interval (high is better). Since the test infrastructure consisted

of only six nodes, these metrics did not vary much, as expected. Topological stabilities are instead more likely to occur in larger networks, as can be seen in our evaluation of RPL++ in Section 3.4.1.

3.3 Testbed Experiments - Interference

3.3.1 MiCMAC

MiCMAC [2] is a multi-channel MAC protocol designed with the objective to be able to mitigate experience, and thereby make the network performance more dependable. We describe the protocol design in detail in Deliverable D-2.1, whereas in this deliverable, we present an experimental evaluation of MiCMAC in two large testbeds: Indriya [11] and TWIST [18], a FIRE facility hosted by Technishe Universität Berlin. This experiment makes use of the JamLab tool for interference generation. The traffic pattern used in these experiments is similar to that of a typical SmartParking application. We send UDP packets consisting of sensor samples and network statistics over the multi-hop testbed networks. In a deployed SmartParking application, the parking slot occupation status will be reported in one of the sensor value fields of the UDP packet.

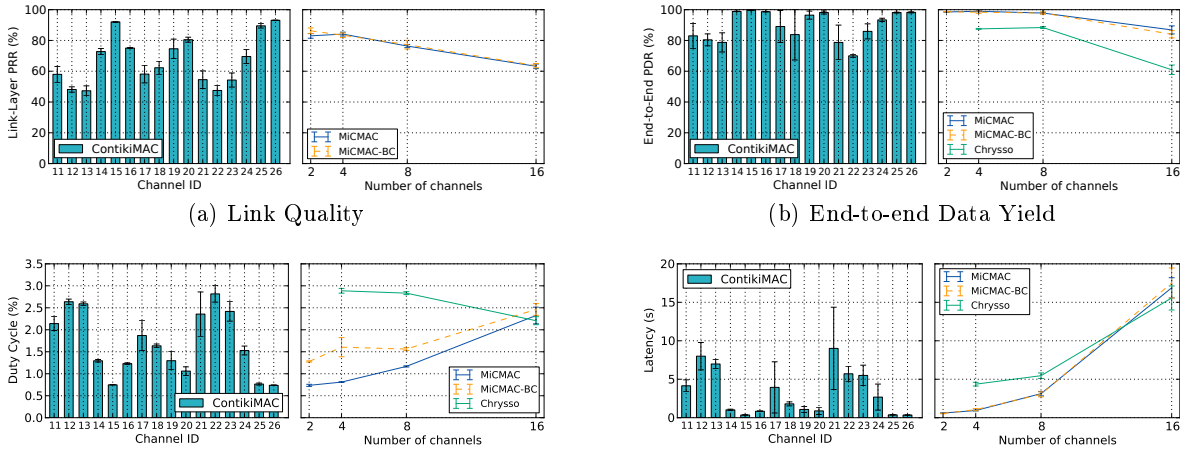
Experimental Results from the Indriya Testbed

We validate MiCMAC experimentally in a 97-node testbed and compare it against the state-of-the-art Chrysso [19] protocol. We run a full low-power IPv6 stack on top of MiCMAC, performing data collection over the standard RPL and 6LoWPAN protocols. Finally, we inject controlled interference to study how the different layers of the communication stack react, and to measure the benefits of multi-channel operation.

Methodology We implement MiCMAC in Contiki, based on ContikiMAC. We run all our experiments in Indriya testbed [11], which at the time of our experiments features 97 TelosB nodes spanning a three-floor office building. We use node #1, in the middle of the top floor, as network root, so that we have nodes up to two floors away from the destination. Our application scenario is a periodic data collection where each node transmits a 64-byte payload datagram to the root at an average interval of 1 min (transmissions are jittered). The network stack is a complete low-power IPv6 stack, with UDP at the transport layer, RPL [23] in charge of routing, and 6LoWPAN as IPv6-to-802.15.4 adaptation layer. It is worth mentioning that running MiCMAC did not require any change in RPL routing nor other layers – we use the out-of-the-box Contiki-2.7 network stack. At the MAC layer, we set the MAC wakeup frequency to 8Hz (ContikiMAC's default).

We run RPL for upwards traffic only (as the scenario is a data collection), with ETX as a metric and the MRHOF objective function. In this setting, RPL boils down to a gradient collection protocol similar to CTP [17][1]. The link estimator is used *as is* even with MiCMAC: the link ETX between two nodes is updated at every transmission attempt, independent of the channel, resulting in an aggregated estimate over all channels in use.

[1] For more details, we refer the reader to the RPL RFC [23].



(a) Link Quality

(b) End-to-end Data Yield

(c) Energy

(d) Latency

Figure 3.9: **Performance of MiCMAC, ContikiMAC and Chrysso with Different Channel Settings.** The performance of MiCMAC with 2 to 4 channels is similar to that of ContikiMAC running on the best available channels (26, 15, 25, or 20). As the number of channels increases (to 8 or 16), worse channels are being used, and MiCMAC results in a compromise between the channels in use. Chrysso exhibits low PDR overall, but also shows better scalability with the number of channels than MiCMAC (since MiCMAC has CSMA backoff and broadcast strobe time proportional to the number of channels).

We compare three different protocol stacks:

RPL/ContikiMAC Using Contiki's default power-saving MAC and RPL implementation. This is our baseline, operating over a single radio channel (unless explicitly mentioned, we use channel 26, which yields the best results).

RPL/MiCMAC Our MiCMAC implementation running below RPL. We use it in different settings, with the number of channels ranging from 2 to 16.

Chrysso We compare the RPL-based solutions to Chrysso [19], a multi-channel collection protocol where MAC and routing are integrated.

We focus on the following key metrics:

Link-Layer Packet Reception Rate (PRR) Represents the transmission success rate for packets, at the MAC layer. Maximizing this metric is not an end goal for the application, but rather an indicator of *the quality of the radio medium* during a given experiment.

End-to-End Packet Delivery Ratio (PDR) Represents the transmission success rate for datagrams, computed end-to-end, from the initial sender to the network root over multiple hops. It tells *how reliable the protocol is*.

Duty Cycle We use duty cycle, the portion of time where the radio is turned on, as a platform-independent metric for power. It tells *how energy-efficient the protocol is*. We measure the duty cycle inline using Contiki's energy profiler [14].

Latency We measure latency as the time difference between the reception of datagrams at the root and its initial transmission time from the originator. We base the measurement on testbed

timestamps of the serial output from the sender and receiver nodes. For some applications (*e.g.*, alarm, live monitoring), minimizing end-to-end latency is a key goal.

We run each experiment for a duration of 60 minutes and extract our results from the last 30 minutes, where the topology is most stable. Note that we observe an initial network setup phase of about 10 minutes in general, after which RPL keeps doing minor topology adjustments but the overall performance has converged. We set the transmission power to 0 dBm. We repeat each experiment at least 3 times. Data points are averaged over all iterations, error bars represent standard deviation across the iterations.

Effect of Multi-channel on Performance We first run ContikiMAC on all individual 16 channels of 802.15.4 to get a picture of each channel's quality, and to measure how RPL/ContikiMAC operate in different channel conditions. From this experiment, we sort the channels by decreasing average PRR. We then run the multi-channel protocols (MiCMAC, MiCMAC-BC, Chrysso), with 2, 4, 8 or 16 channels (we always pick the N best channels according to the aforementioned single-channel PRR measurements). It should be mentioned that these per-channel measurements are not strictly required for MiCMAC to operate, but we do them for the sake of fair comparison.

Figure 3.9a shows the average link PRR obtained in different experiments. It shows that the testbed is subject to WiFi interference, with lower PRR at the most common WiFi channels, and with the best PRR at the 4 WiFi-free channels: 15, 20, 25, and 26. Those are the 4 channels we use in further 4-channel experiments.

In reliability (Figure 3.9b) and duty cycle (Figure 3.9c), MiCMAC keeps the overhead over the best ContikiMAC results at a reasonable level, in spite of the increased cost for broadcast (for instance, channel 15 yields a 99.7% PDR and 0.75% duty cycle vs. 99%, 0.81% duty cycle for MiCMAC with 4 channels). MiCMAC suffers from a latency increase from 0.35s (ContikiMAC, channel 15) to .91s (MiCMAC, 4 channels). This is explained by the longer CSMA back-off that MiCMAC uses, multiple of the number of channels in use. When using 16 channels, the performance degrades due to using all (including bad) channels and due to increased cost of broadcast and channel-lock operations. MiCMAC-BC achieves performance similar to MiCMAC, except in duty cycle, where the extra wakeup on a broadcast channel increases the baseline consumption (the trade-offs of using a dedicated broadcast channels are evaluated in more details in §3.3.1).

In contrast, Chrysso suffers from a reduced data yield (about 88% for 4 and 8 channels, and close to 60% for the case of 16 channels), and results in higher duty cycle than MiCMAC. The reduced data yield is attributed to the occurrence of asymmetric links between child nodes and their parents on the testbed. Especially, when a child node does not receive acknowledgments for its data packets on account of link asymmetry, it eventually executes the channel scanning routine to find a new neighbor. As the decision to perform channel scanning is deferred until the control loops fail to re-connect the child to the routing tree, the child node incurs a significant delay that directly affects data yield. Likewise, the higher duty cycle achieved by Chrysso is attributed to the frequent use of channel scanning on account of asymmetric links. Overall, we find that MiCMAC outperforms Chrysso on all the three metrics.

Our experiments show that the set of channels used has tremendous impact on performance. Although MiCMAC would still have a good chance of communication due to channel hopping,

we would recommend to carefully profile every individual channel in pre-deployment tests. In our results for example, where the 4 WiFi-free channels show much better performance than others, MiCMAC sees its performance degrade when using more than 4 channels. Performing inline channel blacklisting would be a possible extension of MiCMAC, but this would require some extra control traffic for nodes to notify their neighbors upon every blacklist update.

Overall, this series of experiments shows that MiCMAC operates over multi-channel with little overhead, with end performance similar to that of ContikiMAC experiments over the same set of channels.

Resilience to External Interference We evaluate the efficacy of MiCMAC when it comes to recovering from external interference. To experiment in controlled environment, we use WiFi-free channels only, *i.e.*, 15, 20, 25, and 26, but inject emulated WiFi interference using the JamLab tool [6] over a single channel (we pick the best channel, 26). We set 4 nodes (id #2, #4, #5, and #12) close to the root to generate WiFi interference following JamLab's implementation of the Garetto model [6], emulating an access point with 25 hosts (which results in a measured loss rate of about 81% for nodes next to the interference source). We use 4 nodes in order to widen the range of interference in a setting where all nodes in the testbed use the same transmission power of 0 dBm. We periodically turn the interferer nodes on and off at a 5 minute interval to observe how different protocols react to changes between bursty and noiseless environment.

Figure 3.10 shows how different metrics evolve during the course of the experiment, for ContikiMAC and for MiCMAC in 2-channel or 4-channel settings. A first observation is that MiCMAC, even when using no more than 2 channels, keeps its reliability high during interference periods (above 90%), while ContikiMAC drops down to around 40% PDR. This is explained by channel diversity: when losses occur on a channel, the next transmission attempt, on a different channel, does not necessarily suffer from the same interference. Consequently, losses are largely hidden from the routing layer, resulting in few RPL parent switches, and a more stable topology. In contrast, ContikiMAC compensates losses with link-layer retransmissions, increasing duty cycle and latency. Note that RPL routing protocol reacts accordingly: certain links are classified as bad (high ETX), forcing nodes to switch parent. As a result, better links are used, which explains the increase of PRR on channel 26 during the course of the experiment. A downside of this topology adaptation is increased hop count, which occurs during the first interference period and only in ContikiMAC case.

This experiment shows that unlike ContikiMAC, MiCMAC successfully recovers from interference by hiding link losses to upper layers, keeping the topology stable and application-layer metrics high.

Topology As found in the above experiments, channel conditions affect the routing topology and the resulting hop count. Figure 3.11 gives a closer look at the resulting topology in different scenarios.

Figure 3.11a shows a sample (and typical) topology obtained when running ContikiMAC on channel 13, *i.e.*, the worst observed channel. The resulting topology has up to 6 hops. In contrast, when running on the best channel (26), the topology is more compact, with only 4 hops, because the nodes are able to reach further (see Figure 3.11b). Interestingly, running

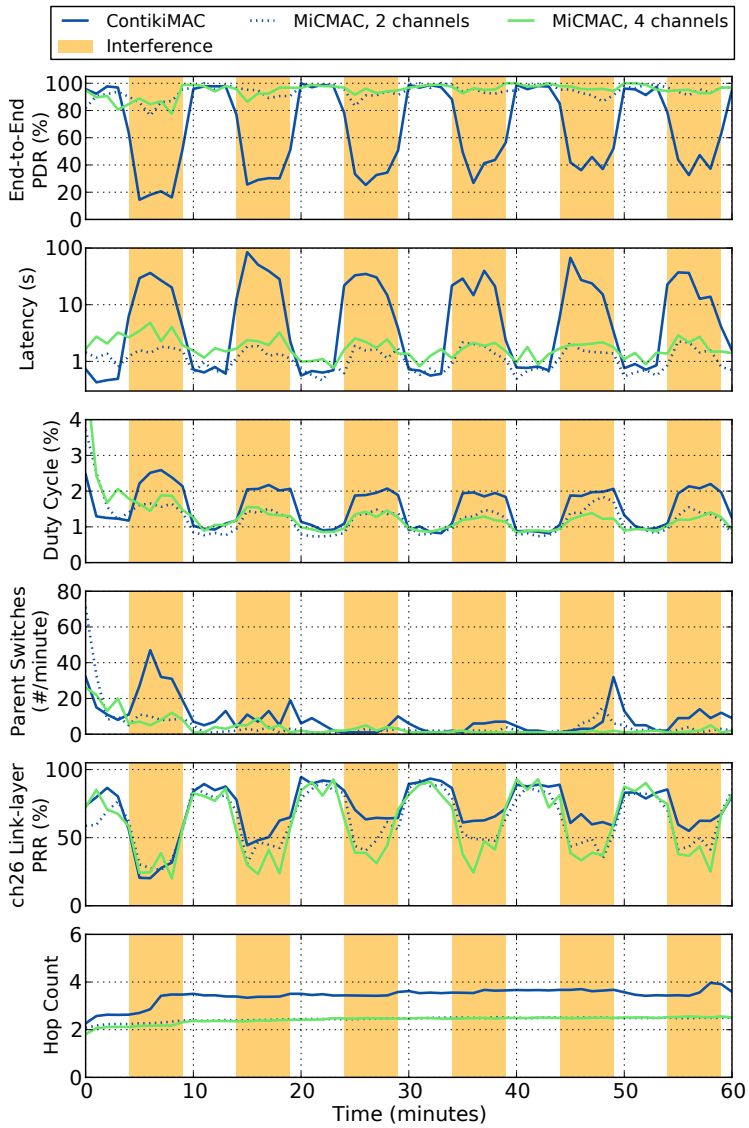
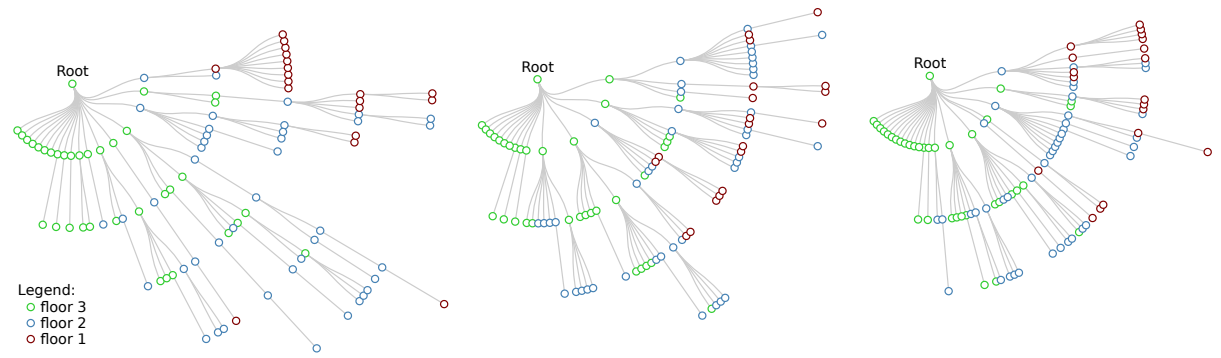


Figure 3.10: **Effect of External Interference on ContikiMAC and MiCMAC.** MiCMAC increases robustness to external interference through channel hopping, resulting in higher packet delivery ratio, lower latency and duty cycle than ContikiMAC. MiCMAC hides most of the link losses to the upper layer, and does not force RPL to react during interference (fewer parent switches and no change in hop count).

MiCMAC over 4 channels (see Figure 3.11c) results in an even more compact topology, with now only one node 4 hops away from the root. This is explained by channel diversity, which increases the number of usable links due to different signal propagation obtained when hopping to a new channel. Note that channel diversity also leads to a more stable topology, as reflected by the reduced number of parent switches. This behavior helps MiCMAC reaching high performance,



(a) RPL / ContikiMAC on Channel 13

# hops	2.42
# neighbors	13
Rank (ETX)	5.78
Parent switches / min	6

(b) RPL / ContikiMAC on Channel 26

# hops	2.48
# neighbors	17.1
Rank (ETX)	2.65
Parent switches / min	0.62

(c) RPL / MiCMAC on 4 Channels

# hops	2.16
# neighbors	17.4
Rank (ETX)	2.59
Parent switches / min	0.12

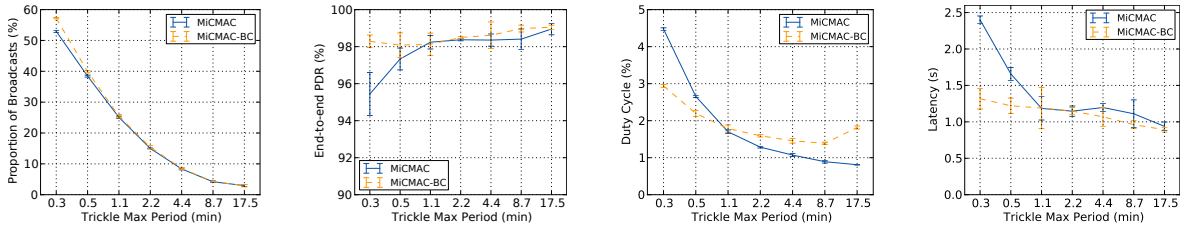
Figure 3.11: **RPL Topology Obtained with Different Channel Settings.** When running on top of ContikiMAC in bad channel conditions (channel 13, PRR of 42.8%), RPL builds a topology with up to 6 hops. On a good channel (channel 26, PRR of 93%), nodes can reach farther as no more than 4 hops are required to connect the network. MiCMAC, through channel diversity, increases the number of usable links, making it possible for RPL to build an even more compact topology, with most nodes in the [1-3]-hop range.

both under interference and in good channel conditions.

Optimizing for Unicast vs. Broadcast We finally look at the tradeoff of running MiCMAC with or without a dedicated broadcast channel, under both broadcast-intensive or unicast-intensive settings. To this end, we vary the maximum interval of the RPL beaconing (based on a so-called "Trickle" timer), within the range 2^{14}ms (0.3 min) to 2^{20}ms (17.5 min) (the latter being RPL's default). As Figure 3.12a shows, this results in broadcasts constituting from about 50% of the overall traffic (when the Trickle max period is 0.3 min) down to about 2.5% (with Trickle max period of 17.5 min).

In broadcast-intensive scenarios (Trickle max period between 0.3 min and 1.1 min), MiCMAC-BC performs best: its cheaper broadcast strobing length reduces contention and energy use. The crossing point between MiCMAC and MiCMAC-BC is at a Trickle max period of about 1 min, *i.e.*, in a setting where 25% of the overall traffic is broadcast. This holds for PDR (Figure 3.12b), Duty Cycle (Figure 3.12c) and Latency (Figure 3.12d). In unicast-intensive scenarios (Trickle max period above 1.1 min), MiCMAC-BC performs similarly to MiCMAC in PDR and latency but results in a higher duty cycle. Looking at where energy is spent in more details (Figure 3.13), we see that MiCMAC-BC have a more expensive wakeup as it has to check the broadcast channel periodically.

Another fact that is worth noting when looking at the Tx/Rx ratio in Figure 3.13 is MiCMAC-BC occupies the channel less than MiCMAC does. This is explained by shorter broadcast strobes and shorter channel-lock strobes as both of them happen on one channel only. This



(a) Broadcast Proportion of Total Traffic (b) End-to-end Data Yield (c) Energy (d) Latency

Figure 3.12: **MiCMAC with and without a Broadcast Channel in more or less Broadcast-intensive Scenarios.** The dedicated broadcast channel proves useful in broadcast-intensive cases, where it saves energy (cheaper strobing) and improves latency (less internal interference). With less frequent broadcasts (e.g. Trickle max period of 17.5 seconds), both protocols perform similarly except in energy, where the broadcast channel costs more than it saves.

could be exploited to minimize interference with nearby networks.

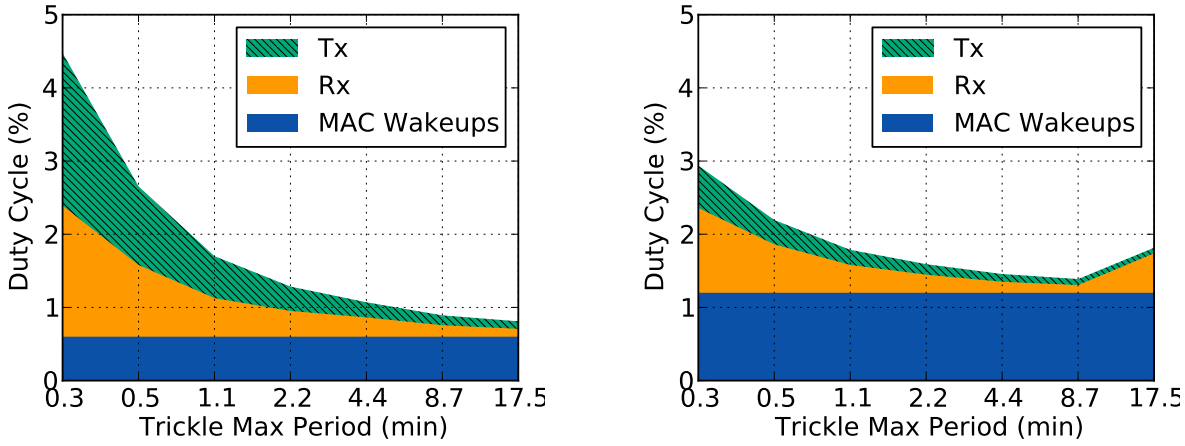
Experimental Results from the TWIST Testbed

In this experiment, we run a UDP-based data collection application on top of Contiki's IPv6 stack. Out of a total of 87 nodes used for the experiment, 83 nodes are data collection clients, with the responsibility of sending sensor samples, network statistics, and run-time assurance information to the sink node. To be able to collect fine-grained statistics of the network performance over time, we let the clients send data packets with an interval of 1 minute. The sink finally prints the statistics on the serial port, whence it is read and logged.

The interference is generated using the JamLab [6]. Three nodes placed randomly in the dense TWIST network run a jamming pattern by which they switch repeatedly between constant jamming for five minutes and no jamming for another five minutes. This pattern will show how MiCMAC operates when challenged by heavy interference. We compare between two settings of MiCMAC: single-channel and multi-channel. The former setting is the baseline, since it essentially is the same as the default MAC protocol in Contiki, ContikiMAC [12], when running on a single channel. When running in multi-channel mode, MiCMAC is able to carry out its measures against interference with only two channels being used.

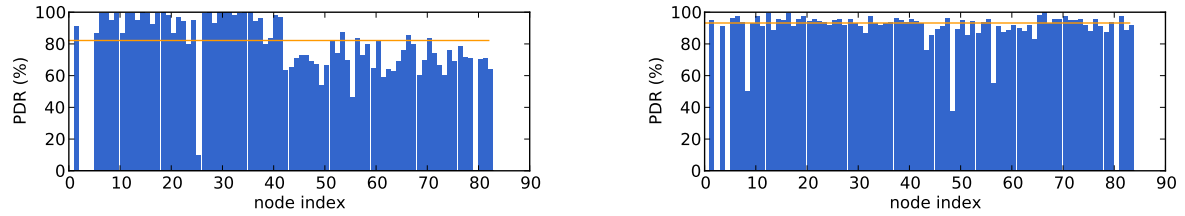
Results Figure 3.14 shows the end-to-end packet delivery ratio (PDR) for all data collection clients. In the baseline case with 1 channel, there is a large fraction of nodes that exhibit severely degraded PDR when the three JamLab nodes exert their intense interference on the network. When the nodes are unable to transmit the data packets, the internal queue of the nodes build up, and packets eventually get dropped from the queue. The multi-channel mode responds to the situation by simply jumping to a clear channel, and can thereby process packets in the queue in a stable manner.

In Figure 3.15, we see that lost packets have a major effect on the stability of the topology. ContikiRPL uses the estimated number of transmissions (ETX) as the main link metric, and this



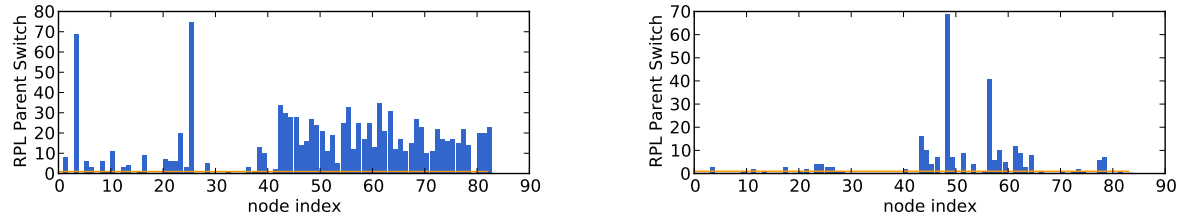
(a) MiCMAC (b) MiCMAC-BC

Figure 3.13: **Energy Profiles of MiCMAC with and without Broadcast Channel.** With a dedicated broadcast channel and in broadcast-intensive scenarios, the reduced cost for broadcast transmissions outweighs the overhead of checking an extra channel at every wakeup.



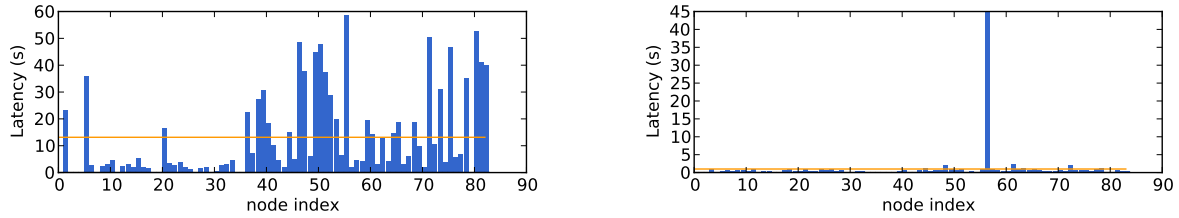
(a) Single-channel mode. (b) Multi-channel mode.

Figure 3.14: The end-to-end packet delivery ratio of different nodes.



(a) Single-channel mode. (b) Multi-channel mode.

Figure 3.15: The stability of the routing topology, as indicated by the number of parent switches made by RPL.



(a) Single-channel mode.

(b) Multi-channel mode.

Figure 3.16: End-to-end latency for data collection packets.

metric is updated after each packet transmission by using an exponentially-weighted moving average. During the interference in 5-minute intervals caused by the JamLab nodes, the ETX will have time to reach a high value, causing the nodes to exhaustively look for other parents in their neighbor set. The multi-channel MAC does not suffer from this instability in the topology, since the ETX metric is stable when switching to another channel.

3.3.2 Evergreen in the TWIST Testbed

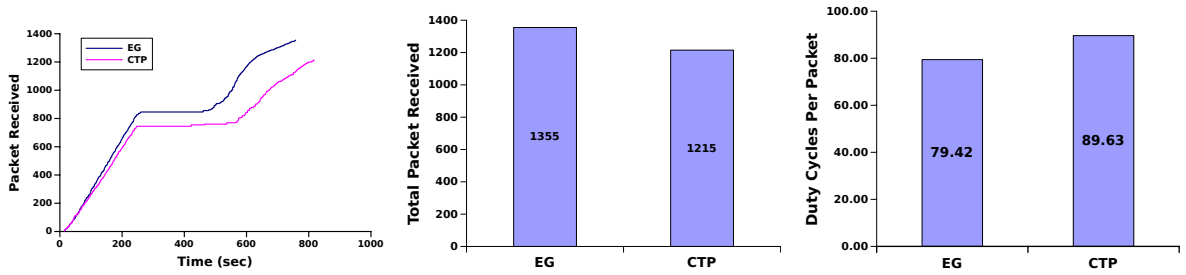
Existing collection protocols are typically created to work well under a set of specific assumptions. Consequently, the performance of these protocols deteriorate significantly when used in difficult conditions violating the assumptions of their design. The challenge now is to lift some of these constraining assumptions such that collection protocols can also operate in tomorrow's networks.

Evergreen is a new collection protocol designed to work—and not to break down—in extreme conditions including: high interference, large temperature variations, node mobility, and high data rates. In addition Evergreen also aims to perform well in stable, more favourable environments achieving at least the same performance as existing protocols designed for those conditions.

Main Contributions Evergreen has the following main characteristics. It is because of these characteristics that Evergreen is tolerant to changes in the network and environment.

1. *Link Quality Estimation (LQE)*: Evergreen's link quality estimation is designed to quickly recognize a lossy link. This helps in reducing the number of packets being dropped or delayed, when a previously usable link is suddenly disconnected due to interference or temperature instability. Another key contribution of our LQE is its ability to maintain link quality estimation in the absence of data packets. This comes handy if the application has a busty traffic pattern, as link estimation remains available for routing any sudden traffic flow.

2. *Alternative Forwarding Algorithm*: When the main algorithm of Evergreen cannot forward packets due to the changes in the network then an alternative algorithm kicks in. The main algorithm is useful to find near optimal paths in stable network conditions whereas



(a) Packets Received Per Sec (b) Total Packet Received (c) Duty Cycles Per Packet

Figure 3.17: Number of packets received using Evergreen over-time are shown in Figure 3.17a and total packet received in Fig. 3.17b. Average duty cycles consumed to transmit a single data packet are illustrated in Fig. 3.17c.

an alternative algorithm is used when the network is in a transitional phase and no route to a root node is known.

3. To maintain network state a node has to decide how frequently control messages are to be sent. CTP uses the Trickle timer to start sending control messages with high frequency when significantly better routes, to a root, become available. Evergreen employs the same strategy, but uses an additional set of comprehensive strategies for asserting when to increase the frequency of control messages. These strategies are critical in efficiently maintaining the network state in different situations.

4. Existing protocols use same-sized control messages for three different purposes: i) a root announcing its existence, ii) a node communicating path cost to a root and iii) to maintain a link quality estimation and link existence. Using a single, large message for multiple purposes increases the overall overhead of control messages. In contrast, Evergreen uses three different control messages depending on the state of the network. This leads to a reduction in overhead and allows running at lower duty cycles, leading to increased network lifetime.

5. Miscellaneous Optimizations: Evergreen also employs several other optimizations including packet aggregation, packet queue management, and short-circuit data forwarding.

Evaluation Setup We use Twist to compare Evergreen's performance against the Collection Tree Protocol (CTP) [15]. CTP is a well-known protocol and one of the most widely used in the sensor network community. At the time of writing, Twist had 96 active wireless sensor motes [16]. A node near the edge of the network was selected as the root (sink), while four nodes, that were the farthest away from the root, were selected as sources of data traffic. Due to these settings and Twist's network layout, most of the data had to traverse at least four hops before reaching the root node. Each source node sends a new data packet every second destined to the root, and each source sends a total of 500 packets.

Results The aim of this evaluation is to compare Evergreen with CTP under a high-interference environment. To emulate interference we use JamLab [6], a tool that is capable of producing different kinds of interference on a given radio channel. To resemble a particularly stringent setting in our Car Parking use-case, we emulated the interference caused by Wi-Fi video streaming. Three nodes in the neighbourhood of the root node are selected as interferer. These nodes generate Wi-Fi interference for a few minutes in the middle of experiment. When the Wi-Fi interference is active, the root node is disconnected from the rest of the network and data packets cannot reach it.

Thus, in order to maximize data delivery under high interference, a protocol should be able to quickly identify viable links to use them for data traffic. Furthermore, a protocol should also be able to quickly propagate any routes available to the other nodes in the network. The results of Fig. 3.17a show that Evergreen possess the ability to identify and propagate link state information in a fast manner. We observe that the period when there is no traffic towards the root (due to interference), is much shorter in Evergreen as compared to CTP. Evergreen is able to transmit 12% more data packets as compared to CTP. Furthermore, Evergreen consumes less energy. Since most of the energy is consumed by the radio, we measured radio duty cycles for both protocols. Fig. 3.17c shows that, with Evergreen, a data packet needs on average 11% less duty cycle to reach the root node. Overall, compared to CTP, Evergreen is able to deliver more data packets, in less time while conserving power.

3.4 Testbed Experiments - Temperature

3.4.1 RPL++

A hypothesis that we made in the early stage of Task 2.2 *Protocol Design*, was that we could use the environmental model and platform model to improve the routing layer of our integrated prototype. To this end, we extended the RPL protocol to include this information in its routing decisions. RPL is an IETF standard for IPv6 routing in low-power and lossy networks [23]. The Contiki operating system, which stems from project partner SICS, has a well-tested, open-source implementation of RPL called ContikiRPL. This implementation provides a strong baseline for comparison with environmentally-aware protocols.

RPL is a distance-vector protocol that selects routes based on the rank as calculated by an *objective function* (OF). A few objective functions have been specified as IETF RFC:s, including the Minimum-Rank with Hysteresis Objective Function (MRHOF), which is the default one in ContikiRPL. Whilst MRHOF provides some freedom as to which link metric implementations can use, ContikiRPL and most other implementations that we know of try to minimize the additive ETX link metric [10] for all links along a routing path. To avoid high churn of routes, MRHOF includes a hysteresis mechanism that allows the current parent of a node to vary in rank within a configurable range, as long as there is not another candidate parent that will give the node a lower rank than the minimum value of this range. The ETX itself is calculated by aggregating the statistics of sent unicast packets into an exponentially-weighted moving average value. Hence, it is dependent on recent traffic in order to give a reasonable assessment of current link conditions.

Temperature-Aware RPL (RPL++) is an extension of ContikiRPL developed with the objective to be able to avoid selecting routes that can be adversely affected by environmental

conditions. RPL++ makes predictions about which routing paths might degrade without relying on current traffic statistics as the baseline RPL does. RPL++ makes two extensions of ContikiRPL: 1) a complex objective function (TEMPOF) that uses the environmental model (Task 1.1) and the platform model (Task 1.2) to calculate the worst-case signal-to-noise (SNR) ratio. We describe this in more detail below. 2) A new routing metric container that contains the environmental and platform model of each node. This information is disseminated in routing control messages (RPL DIO messages), which are able to handle custom metric containers in an opaque manner.

The worst-case SNR for each neighbour N can be assessed by measuring the noise floor of the node, recording the received signal strength (RSS) of incoming packets from N, and using this information together with the environmental and platform model parameters obtained from the first received DIO from N. If TEMPOF has not yet recorded any model parameters for N, it calculates the rank of the node precisely as MRHOF does, using the ETX link metric. When the model parameters are available, TEMPOF will calculate the worst-case SNR information and penalize the ETX-based rank for node N according to the following formula.

$$N_{rank} \leftarrow N_{rank} + \frac{P_\alpha}{2+{N_{snr}}^2}$$

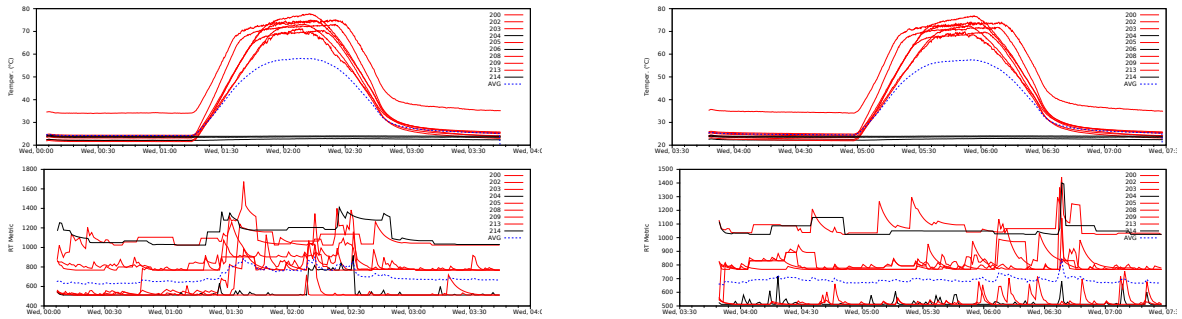
P_α is a penalization factor that we use to vary the weight of the worst-case SNR in relation to the basic ETX link metric. S_{worst} refers to the estimated worst-case SNR of node N. The formula is only used if the N_{snr} is below a threshold, P_t, where it makes sense to penalize. In the current version of RPL++, we set P_α to 8 and P_t to 5. A lower P_α will cause nodes to put more significance on the rank of nodes rather than the prediction of link attenuation. This will give a short average hop length on average throughout the network. If P_α is set higher, the routing decisions will be taken primarily with respect to the forecasted link attenuation, and secondarily with respect to the RPL rank.

Results

We have evaluated RPL++ in TUG's TempLab testbed, where we are able to generate controllable temperature variations that can affect link qualities considerably. The results from this evaluation, however, have not provided us with conclusive evidence that the routing layer optimizations carried out in RPL++ are beneficial for dependable performance.

Figure 3.18a and 3.18b show how the routing metric can be affected by temperature changes. These results stem from two experiments in which we subjected regular RPL and RPL++ to a major temperature change in the TempLab testbed. Both protocols show that the routing metric is more volatile during the heating period. In this case, RPL++ appears more stable but the results that we have collected over a large number of experiments do not show conclusive evidence that RPL++ provides a higher stability.

In some tests, we have observed that regular RPL is more stable, which leads us to make the hypothesis that the network performance, when being subjected to large temperature variations, is primarily affected by factors at the link layer. Furthermore, we believe that the density of the network can in two different ways counter-act the measures taken by the routing protocol. First, in a dense network such as TempLab, there is often an abundance of candidate parents to choose from. If the selected parent's link quality decreases, the routing protocol can quickly



(a) Regular RPL

(b) Temperature-Aware RPL

Figure 3.18: A timeline of how the routing metrics (bottom graphs) change in relation to the temperature curve (upper graphs).

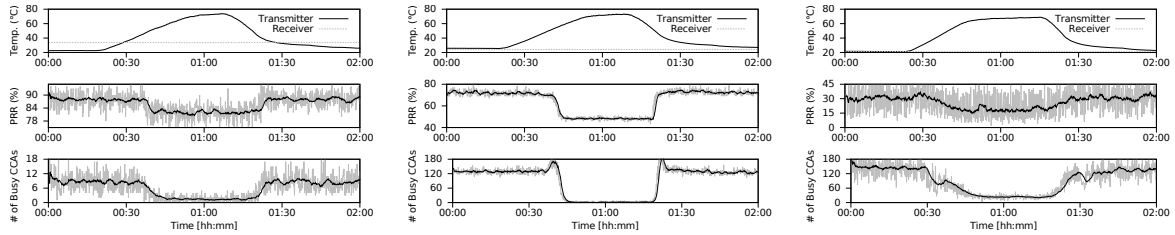
find another sufficiently good parent. Second, in a sparse network, the number of choices may be so small that a temperature-aware routing protocol cannot find a better choice than the currently selected parent, even if the protocol has forecasted that the link with that parent will degrade later.

Instead of optimizing for temperature at the routing layer, we have indeed found that the information is best used at the MAC layer, as we will show below in Section 3.4.2. Although the routing penalization has not yielded results that support our initial hypothesis of being able to create favourable routing topologies before environmental changes begin to affect the network negatively, we have been able to use the second part of the RPL++ extension to disseminate information about each node's environmental model and platform model, which will be used to improve the performance of the Temperature-aware MAC protocol described next.

3.4.2 Temperature-Aware MAC

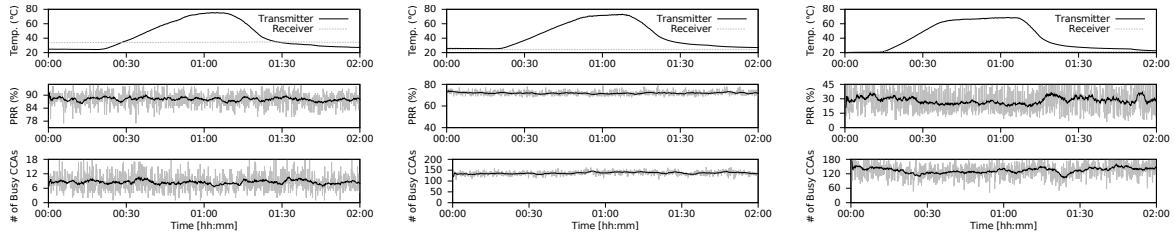
Our experiments with a temperature-aware routing protocol shown in Section 3.4.1 did not lead to the expected improvements in terms of dependability. Therefore, we now exploit the temperature information directly at the MAC layer in order to mitigate the impact of temperature fluctuations. Such fluctuations can indeed considerably reduce the efficiency of carrier sense multiple access (CSMA) data link layer protocols, leading to a substantially decreased packet reception rate and to increased energy consumption. We identified a reduced effectiveness of clear channel assessment (CCA) as the reason for such performance degradation, and showed that this reduced effectiveness compromises the ability of a node to avoid collisions and to successfully wake up from low-power mode. Based on these insights, we propose two mechanisms to mitigate the problem by dynamically adapting the CCA threshold to temperature changes: one based on the temperature measured locally, and one based on the highest temperature measured across all neighbouring nodes.

We now evaluate the performance of the proposed approaches using our TempLab testbed infrastructure. Because the parking management use case scenario is outdoor, we replay temperature traces that resemble the on-board temperature collected in real-world outdoor deployments, as well as in countries with extreme weather conditions, with daily temperature



(a) JamLab's emulated Wi-Fi video streaming (b) JamLab's emulated microwave oven (c) File transfer between two Wi-Fi devices

Figure 3.19: When using fixed CCA thresholds, temperature affects the efficiency of collision avoidance in CSMA protocols.



(a) JamLab's Wi-Fi Video Streaming (b) JamLab's Wi-Fi Active Microwave Oven (c) Real Wi-Fi File Transfer

Figure 3.20: When adapting the CCA threshold based on local temperature measurements, temperature does not affect the efficiency of collision avoidance in CSMA protocols. In contrast with the results shown in Fig. 3.19, the PRR remains fairly constant for all interference scenarios despite temperature variations.

fluctuations up to 50°C. Using this setup, we show that the proposed approaches alleviate the collision avoidance and wake-up problem in CSMA protocols. We then run a network of nodes, and show that when employing a MAC protocol with an adaptive threshold, the performance of the network significantly increases, with up to 42% lower energy consumption and 87% higher PRR in the presence of temperature variations commonly found in outdoor deployments. In the latter experiment, we use a data collection protocol with a traffic pattern that is similar to that of a typical parking management application, with parking sensor nodes periodically sending packets to a sink node.

Improved Collision Avoidance

With high temperature fluctuations, a transmitter employing a fixed CCA threhsold T_{CCA} can erroneously measure a weaker noise and generate wasteful transmissions [8]. We now analyse the performance of the transmitter-receiver pairs in our testbed when dynamically adapting T_{CCA} using local temperature information. We use TempLab [9] to vary the on-board temperature of the nodes between 25 and 75°C, and we carry out experiments consisting of several transmitter-receiver pairs running a basic Contiki application, in which the transmitter node periodically

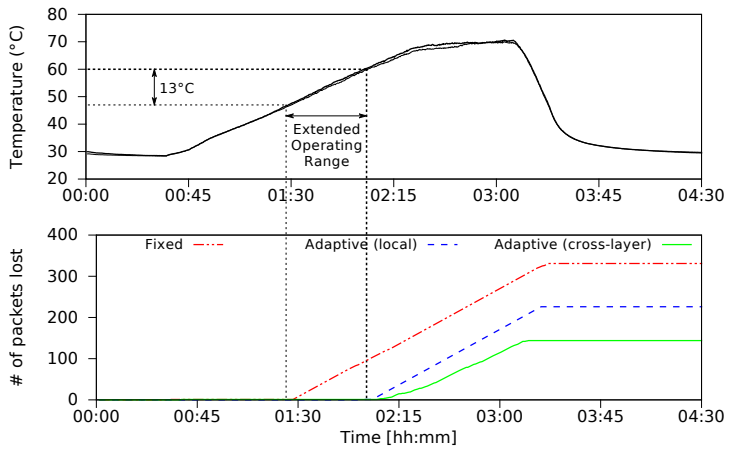


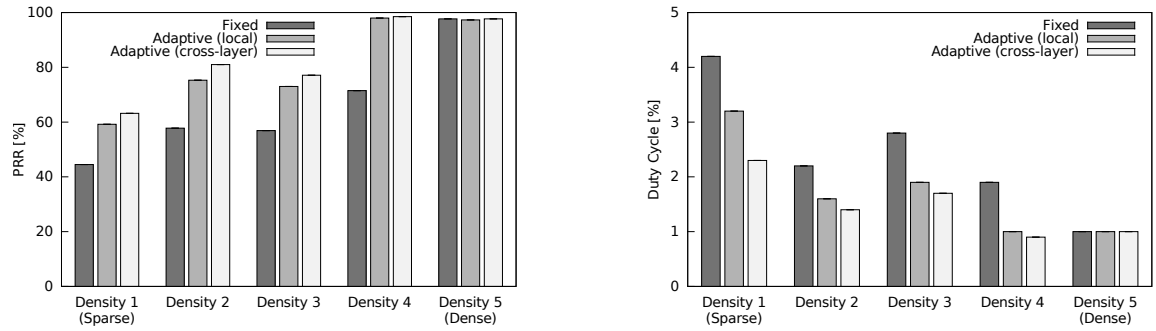
Figure 3.21: Adaptive CCA thresholds alleviate significantly the wake-up problem at high temperatures. By adapting T_{CCA}, we can extend the usability of a link at much higher temperatures.

sends packets to its intended receiver and collects statistics such as the energy expenditure at the link-layer and the RF ambient noise in the radio channel. The latter is computed as the maximum of 20 consecutive RSSI readings after a packet transmission. We further use JamLab [6] to produce repeatable interference in our testbed on different channels. We emulate on one channel the interference caused by a computer streaming videos from a Wi-Fi access point, and on another channel the one caused by an active microwave oven. We also let a computer transfer large files from a nearby Wi-Fi access point using a channel that is not affected by JamLab. We then analyse how this affects the PRR on the transmitter-receiver pairs in our testbed.

Fig. 3.19 shows the impact of erroneous clear channel assessments on protocols employing a fixed CCA threshold in the presence of different interference patterns. Fig. 3.20 shows the PRR experienced by the links when using an adaptive CCA threshold (the experiments were executed back-to-back). If we compare the two figures, we can immediately notice that when using a dynamic CCA threshold, the PRR does not depend on the on-board temperature of the nodes, but remains instead fairly constant throughout the experiment. This hints that the adapted protocol is able to avoid the intersection between the RSSI curve and the CCA threshold, mitigating the collision avoidance problem in CSMA protocols.

Improved Wake-Up Efficiency

Temperature can also affect the efficiency of the wake-up mechanism: a receiver node exposed to temperature variations may not receive a signal sufficiently strong to cause a wake-up of the radio, and constantly remains in low-power mode, causing the disruption of the link. We employ ContikiMAC with a $T'_{CCA} = n'_f + K$ with $K = 6$ dBm and use TempLab to warm-up and cool-down the on-board temperature of both transmitter and receiver, emulating the daily fluctuations that can be found in real-world deployments. We repeat the experiments several



(a) PRR as a function of network density

(b) IR heating lamp on top of a sensor node

Figure 3.22: **Network performance when using fixed and adaptive CCA thresholds as a function of network density.**

times and run (i) an unmodified ContikiMAC using a fixed CCA threshold, (ii) an adaptive threshold based on local temperature information, and (iii) an adaptive threshold based on the information inferred from the routing layer. Fig. 3.21 shows the packet loss rate on a representative link in the testbed (the same trend was observed across all links): the adaptation of the CCA threshold can significantly alleviate the wake-up problem. We can see that the use of a dynamic T_{CCA} essentially extends the usability of a link to a higher temperature (up to 60°C the link experiences 100% delivery rate when using an adaptive threshold based on local temperature information). As expected, the lowest packet loss is recorded when using the information inferred from the routing layer, as it also takes into account the attenuation of the transmitter. It is important to highlight that the adaptation of T_{CCA} does not mitigate completely the impact of temperature. The reason lies in the selection of T'_{CCA}: by selecting $K = 6$ dBm, the high temperature variation attenuates the signal strength by several dB, reaching the physical limit of the radio (i.e., we receive a signal strength that is too weak to be successfully demodulated). Hence, the higher is K, the higher can be the performance gain compared to a protocol using a fixed CCA threshold.

Performance on a Network Level

We now present results obtained running a data-collection protocol on several networks, and show the benefits of using dynamically adapted CCA thresholds in the presence of temperature variations. We use RPL in our testbed deployed in a 55 m^2 room: we select one node as a sink, and we create five different network densities by using only a portion of the nodes: 5, 7, 9, 11, and 13 nodes, respectively. By varying the density from one node every 11 m^2 to a node every 4 m^2, we can see largely different impacts on a network level. Using the same temperature profiles and setup as in the previous example, we carry out experiments with an unmodified ContikiMAC using: (i) a fixed CCA threshold, (ii) an adaptive threshold based on local temperature information, and (iii) an adaptive threshold using the information inferred from the network layer.

Our results indicate that temperature strongly affects network performance, especially in

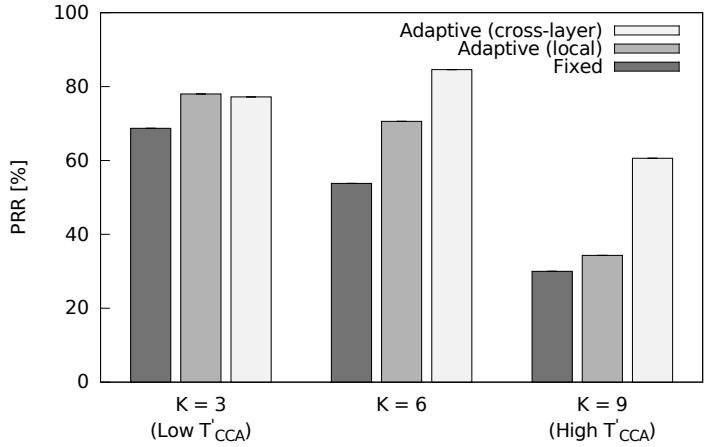


Figure 3.23: **Network performance when using fixed and adaptive CCA thresholds as a function of T'_{CCA}.**

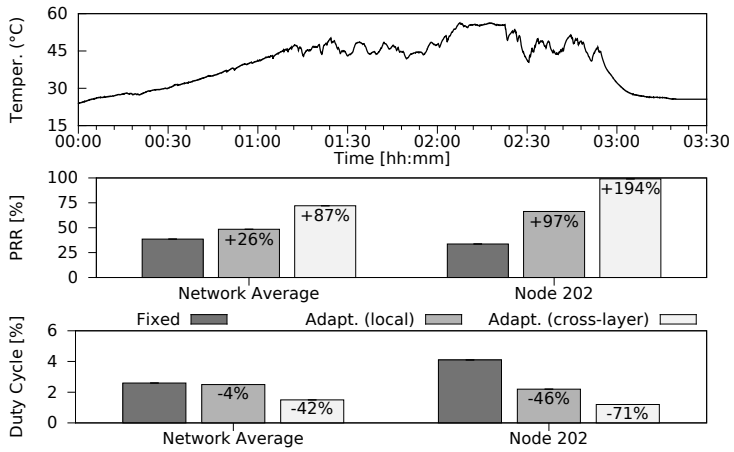
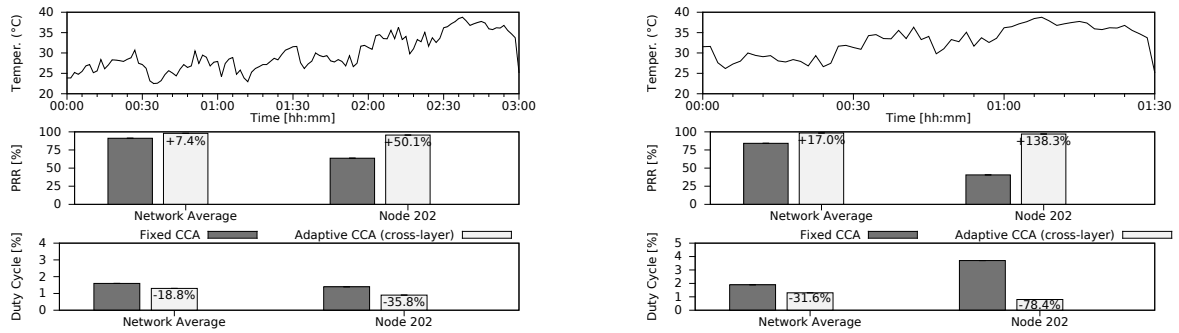


Figure 3.24: Regeneration of a real-world trace recorded in an outdoor deployment in Uppsala, Sweden [22], and impact on PRR and energy efficiency on a network level and on a single node.

sparse networks. Fig. 3.22a shows that if the network is dense, the routing layer can mitigate the impact of temperature and sustain a high PRR even with a MAC protocol employing a fixed CCA threshold. The less dense the network is, the higher becomes the impact of temperature on a protocol using a fixed threshold, with the average PRR in the network dropping below 50%. Instead, when using adaptive thresholds, the network sustains higher reception rates in sparse networks (from 44 to 63%, and from 57 to 81% in the two sparsest configurations), with the highest PRR recorded when using the information inferred from the routing layer in line with the experiments in Sect. 3.4.2.



(a) Average statistics during daytime

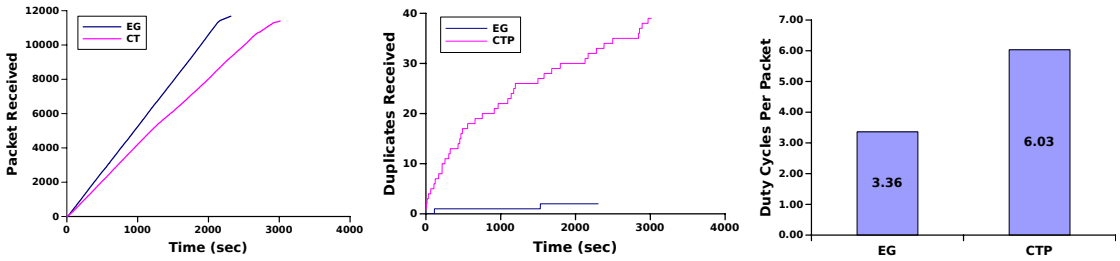
(b) Statistics during the hottest part of the day

Figure 3.25: **Regeneration of a real-world trace recorded in SmartSantander, and impact on PRR and energy efficiency on a network level and on a single node.**

We further analyse the energy-efficiency of the different approaches: Fig. 3.22b shows that significantly lower duty cycles can be achieved when using adaptive CCA thresholds. In the sparsest network configuration, the average duty cycle of the network drops from 4.2% to 3.2% in the case of local temperature information and to 2.3% if the information is inferred from the routing layer. The latter corresponds to a 55% higher energy-efficiency than when using a fixed threshold.
Fig. 3.23 shows the role of $T_{CCA}^{'}$ in a network with a density of one node every 8 m^2. We use set the initial CCA threshold $T_{CCA}^{'} = n_f^{'} + K$ using different K values, and show that the higher K is, the higher are the performance improvements introduced by the adaptive approaches. This is the result of the observation made in Sect. 3.4.2: the higher K is, the more the usability of a link can be extended at high temperatures. Please note that when using high K, the fixed threshold approach sustains a progressively lower PRR, as a result of a lower number of links in the network (only a few links are able to wake up a neighbour with a signal strength higher than $T_{CCA}^{'}$).

Regeneration of traces - Uppsala deployment. We use TempLab to time-lapse a 24-hours trace recorded in an outdoor deployment [22], and see what is the impact in a network with a density of one node every 8 m^2 when using $T_{CCA}^{'} = n_f^{'} + 6$. The results show that the adaptive approaches that we proposed significantly improve performance, both on a link basis and on a network level. Fig. 3.24 shows that the network sustains up to 42% lower energy consumption and 87% higher PRR in the presence of temperature variations commonly found in outdoor deployments, and that a single link may experience up to 71% lower energy consumption and 194% higher packet reception rate.

Regeneration of traces - SmartSantander deployment. We finally use TempLab to time-lapse a 10-hours trace recorded in the SmartSantander outdoor deployment, and see what is the impact in a network with a density of one node every 4 m^2 when using $T_{CCA}^{'} = n_f^{'} + 6$. We reuse a portion of the temperature traces recorded during summer shown in Section 2.1.1.



(a) Packets Received Per Sec (b) Duplicates Per Sec (c) Duty Cycles Per Packet

Figure 3.26: Temperature Variations: Number of data packets and duplicate data packets received per second are shown in Fig. 3.26a and 3.26b respectively. Duty cycles consumed on average to transmit a single data packet are shown in Fig. 3.26c.

Despite the temperature fluctuations recorded in SmartSantander are significantly smaller than in the deployment by Wennerström et al. in Uppsala [22], our experimental results show that also when replaying a trace with a smaller temperature variation, the adaptive approaches that we proposed significantly improve performance, both on a link basis and on a network level. Fig. 3.25 shows that the network sustains up to 31.6% lower energy consumption and 17% higher PRR in the presence of temperature variations commonly found in outdoor deployments, and that a single link may experience up to 78.4% lower energy consumption and 138.3% higher packet reception rate during the hottest part of the day.

3.4.3 Evergreen with Temperature Variations

Evergreen is a collection protocol designed to tolerate network and environment changes. It is summarized in Section 3.3.2 and described in more detail in D-2.1. This section presents Evergreen's performance under temperature variations. Interference on a link can be switched on or off within a few milliseconds, whilst heating or cooling down environment takes at least few minutes. Thus protocols have much more time to adapt to the changes in network dynamics due to variations caused by temperature than by interference. It is observed that collection protocols generally adapt better to temperature variations than sudden influxes of interference.

Evaluation Setup: We used our Templab testbed for the experiments, which has 17 nodes [9]. TempLab is equipped with infrared heating lamps to enable studies of the impact of temperature variations in a real-time controlled environment. For the experiments, a root node (node 214) and three source nodes (node 216, node 206, and node 207) are selected, such that the number of hops between the sources and the sink are maximized for the given testbed. Each source generates two 4-byte long data packets every second. The contents of the data packets are the source identifier and a serial number to uniquely identify the packet. This information is used at the end of an experiment to find duplicate data packets received at the root node by processing the trace files. Each source node sends 4000 packets, and hence each experiment lasts at least $4000 \times 0.5 = 2000$ seconds (or 33.33 minutes). Each node has a 500 ms long sleep cycle. Both Evergreen and CTP are MAC- and hardware-independent protocols, and therefore we used the default MAC protocol of TinyOS 2.x in our evaluations [20]. The infrared lamps available in TempLab heat the nodes until they reach a maximum temperature of 70°C. After waiting for

a short duration, these lamps cool down and temperature gradually returns to normal. This whole process last about 40 minutes.

Results: Fig. 3.26 shows the effects of temperature variations during the experiment. Evergreen performs better than CTP in all scenarios. The most prominent difference in the results, is the average duty cycles consumed to transmit a single data packet by CTP and Evergreen. Evergreen consumes 44% less duty cycles as compared to CTP while transmitting slightly more data packets in a significantly reduced time period. To be precise, Evergreen takes 33% less time to transmit a single data packet while delivering 2.5% more packets than CTP. There are multiple factors that contribute to Evergreen's performance gain. For instance, the reduction in duty cycles is mainly due to our smart control packet transmission strategies. Whereas reduction in data packet loss is due to the adaptive link quality estimation algorithm.

4 Analysis of the Results in Relation to the Selected Use Case

In this section, we analyze the experimental results presented in Chapter 3 with respect to the selected use case for the integrated experiment: Smart Parking. For each of the tested protocols in the integrated prototype—MiCMAC, RPL++, Evergreen, and Temperature-Aware MAC—we extract the key results from the evaluation, and compare them with the quantitative requirements of the selected Smart Parking use case.

First we summarize the dependability requirements reported in Deliverable D-4.1 *Report on Use Case Definition and Requirements*. As for this first set of experiments, we focus on delay and packet loss without taking into account energy consumption in absolute numbers. Then, a brief discussion is offered for every protocol tested.

ID	SP-1
Name	Latency $<$ 30 seconds
Description	The system has to have a time response in less than 30 seconds. Time response is considered as the time between a car change is detected by a mote and data is received by the Gateway.
Priority	M
Failure Effect	A driver could reach, thanks to the system, to a place that is already occupied losing confidence in system, if this situation recurs.

ID	SP-2
Name	Latency $<$ 10 seconds
Description	The system should have a time response in less than 10 seconds. Time response is considered as the time between a car change is detected by a mote and data is received by the Gateway.
Priority	S
Failure Effect	A driver could reach, thanks to the system, to a place that is already occupied losing confidence in system.

ID	SP-3
Name	Data loss $< 10\%$
Description	The system does not lose more than 10% of the events
Priority	M
Failure Effect	System could give wrong information too often.

ID	SP-4
Name	Data loss $< 5\%$
Description	The system should not lose more than 5% of the events
Priority	S
Failure Effect	System could give wrong information sometimes.

ID	SP-5
Name	Data loss $< 1\%$
Description	The system could not lose more than 1% of the events
Priority	C
Failure Effect	System could not have reduced quality of service.

4.1 Temperature and Interference Models

4.1.1 Temperature

As described in Section 3.2.1, we have been able to measure a temperature variation of 15°C (30°C maximum, 15°C minimum) in the major temperature amplitude case. The official air temperature registered on the days that the test was conducted were 12°C minimum to 19°C maximum. As seen, the mote's temperature is heated beyond the official air temperature due to the heating power of the sunlight even in the temperate days of spring. Across the year, much higher temperature variations are to be expected.

These measurements demonstrate that the variations inside the motes, hence in the radio transceivers can vary much more than the official air temperature, raising the necessity for temperature-aware communication protocols for the Smart Parking use case.

4.1.2 Interference

As explained in Section 3.2.2, interference was also recorded in the real deployment. Although the deployment area was an open area inside the city (wide corner street with few offices around), significant interference is observed. Due to the variation in duration and intensity of the interference, new models as being developed in WP3 are needed.

4.2 MiCMAC

As described in 3.3.1 the results for both testbeds used for experimentation with this MAC protocol accomplish the MUST requirements for both latency and packet loss with regard to the requirements for the Smart Parking application.

In both testbeds, with a topology up to 6 hops deep, the packets loss is less than 10% in cases with very strong interference (90% of PDR) while the latency is maintained below 10 seconds (much less than the 30 seconds requirement). Hence, these two requirements are accomplished with respect to the MUST priority.

For the energy consumption, although it has not been measured directly, changes in the duty cycle given an indication. In the case of MiCMAC, the duty cycle is kept below 2%, which is close to the duty cycle of ContikiMAC of about 1%. From that we can conclude that the energy consumption of MiCMAC does not come at a high overhead in return for its interference mitigation capability.

4.3 Evergreen

In Section 3.3.2 Evergreen is compared with the CTP protocol in the TWIST testbed. The experiments show that, in an environment with high interference, the performance of Evergreen is better in terms of throughput, delivery rate and energy consumption (Figure 3.17b and 3.17c). In TWIST, Evergreen delivers 12% more data packets, in 20% less time, while saving 11% of energy. We hence expect Evergreen to deliver a good performance in the Smart Parking use-case by increasing the robustness of the system against different kinds of interference and by extending the motes' batteries lifetime. These results reduces the data loss of the entire system, helping to accomplish the *Data Loss* dependability requirements SP3 - SP5.

4.4 RPL++

The test of an unmodified low-power IPv6 stack running the RPL protocol for routing was performed in the 22@ deployment. The results shows a high packet loss rate for some of the motes. Since this result is greater than the MUST requirement for the SmartParking use case, it is clear that some new algorithms and solutions are required to maintain the network functionality while the motes still are using the radio with low duty cycle.

As the RPL++ experiment does not show conclusive evidence of higher stability or better performance, this algorithm will not been tested in the 22@ deployment, as this first experiments in Barcelona are designed to obtain a baseline for a standard networking stack. The second integrated experiment will integrate the results from WP1 - WP3 developed during the second year of the project with the same conditions and prototype platform that were used this first set of experiments.

4.5 Temperature-Aware MAC

4.5.1 Improved Collision Avoidance

In this experiment, we find that the adaptive CCA threshold significantly enhances the temperature range, as shown in Figure 3.20. This improvement is based on a dynamic adaptation of the CCA threshold, and we observe that it makes the packet delivery more dependable. This result is linked directly to *Data Loss* dependability requirements SP-3 to SP-5 (MUST: Data loss $< 10\%$, COULD: Data loss $< 1\%$).

4.5.2 Improved Wake-Up Efficiency

The stability of the link depends on T_{CCA} value. The conclusion we make from the experiments modifying this threshold is that it can be customized to every particular network topology of each Smart Parking deployment.

4.5.3 Performance on a Network Level

As Smart Parking network topologies are usually sparse, it is important to focus on the density 1 topology from the evaluation. Also, due to the specific topology usually found in this use case, where there is a chain of motes connecting one to another until the last reaches the base station, losing a single link can be catastrophic.

In Figures 3.22a and 3.22b, it can be seen that the adaptive (cross-layer) algorithm performs better in terms of PRR (less packets lost, better link between nodes) and duty cycle (less time radio on) than the other algorithms for the density 1 case.

It can also be seen that stability of the link depends on T_{CCA} value as seen in previous section. Experiments modifying this value end with the conclusion that it can be customized to a particular network topology increasing the performance of the network links and lowering the duty cycle (energy consumption) (Figure 3.24).

4.6 Conclusion for Smart Parking Use Case

We have also observed that Wi-Fi interference in a typical Smart Parking scenario inside a city and high temperature variations are real and can affect the communication between the motes and the base station. While a state-of-the-art network stack fails to satisfy the MUST requirements for the selected use case, experimentation in testbeds makes us confident that the improved protocols will be able to meet these requirements. The components will now be further improved and integrated again into a final prototype towards the end of the project in Task 4.4. This prototype will then be deployed and tested in the parking scenario.

5 Lessons Learnt and Next Steps

The first integrated experiment has revealed several important insights that will be used to improve upon the work in WP1-3. The integrated prototype has effectively combined learning of environmental parameters and models with the newly developed and adapted protocols devised in Task 2.2 *Protocol Design*. The temperature model has proven useful in the Temperature-Aware MAC protocol, where we have shown that by adapting the CCA threshold based on the temperature, the communication performance increases considerably. By calculating the link attenuation caused by temperature increase, Temperature-Aware MAC knows how much to adjust the CCA threshold to compensate for the adverse effects of temperature. Our initial hypothesis that such effects can be mitigated through the routing layer did not hold, however. Through extensive experiments, we did not find a statistically significant improvement in the various metrics that we consider. These findings have led us to conclude that we should focus on the link layer with regards to improving the performance of applications used in environments where the temperature fluctuates.

The other major cause of problems that we strive to mitigate is interference. The integrated prototype includes two protocols developed to this end: MiCMAC and Evergreen. The former operates at the link layer, whereas the latter operates at the routing layer. We have tested these protocols using the JamLab interference generation tool developed in Task 4.1 *Testbeds with Realistic Environmental Effects*. As can be observed through the experimental results in Chapter 3, both protocols provide good performance under interference. JamLab has proven highly valuable in testing the interference in a repeatable manner.

Next we describe the lessons learnt for each individual protocol that has been tested in our experimental evaluation, and outline the next steps to be taken within the project.

MiCMAC The asynchronous and unscheduled nature of MiCMAC makes it practical in low-power IP scenarios. We implement our protocol in Contiki and run it in a 97-node testbed, running a complete low-power IPv6 stack, with RPL at the routing layer. MiCMAC achieves performance that makes it suitable in very demanding scenarios, conciliating 99% end-to-end reliability, sub-percent duty cycle and sub-second latency. Our experiments with injected external interference show that MiCMAC hides losses from the routing layer, resulting in a more stable topology. It maintains high reliability even during heavily interfered periods, where ContikiMAC drops the delivery ratio below 40%. For the next steps, we believe that it may be possible to improve MiCMAC further by having automatic channel selection.

Evergreen Evergreen is a hardware independent collection protocol that is currently under development. Preliminary results show that this new protocol has the potential to improve performance in presence of high interference and temperature variations. Evergreen's benefits are three-fold: i) higher packet delivery rates, ii) lower energy consumption, and iii) lower delays. Evergreen has been evaluated on Twist (96 motes) and on TempLab

(17 nodes), and in both networks it increases throughput and conserves power. However, given that Evergreen is a new protocol, which is implemented without using any code from existing protocols, it is desirable to evaluate it further on more networks. We observed that the benefits of Evergreen are more apparent with interference than with relatively gradual temperature variations. In general, current collection protocols adapt better to temperature variations than sudden influxes of interference. A limitation of Evergreen is that it is built as a best-effort protocol for catering a variety of situations, and its current implementation is not able to provide performance guarantees for a specific environmental condition or network setting. Adressing these limitations will be the subject of future work.

RPL++ One of our hypotheses has been that we can use temperature hints to improve the performance of the system at the routing layer. Unfortunately, we were not able to attain any statistically significant improvements over standard RPL. In sparse networks, there are few options to create routes, and excluding nodes that can be highly affected by temperature may create poor routing paths. In dense networks, the number of candidate routes is large, so the switch to another good parent can be done quickly as soon as the performance for the current parent deteriorates. Rather than working with the temperature information at the routing layer, we have found that the performance is best improved at the link layer. Hence, we will discontinue the work on RPL++ and focus on Temperature-Aware MAC to create dependable performance under temperature variations.

Temperature-Aware MAC Using temperature hints to improve the performance at the MAC layer proved to increase the robustness of sensornet protocols to temperature variations. The latter can affect the efficiency of clear channel assessment and may compromise the operations of data link layer protocols based on carrier sense, especially the ability of a node to avoid collisions and to successfully wake-up from low-power mode. We have designed and evaluated two mechanisms to mitigate the problem by dynamically adapting the CCA threshold to temperature changes: one based on the temperature measured locally, and one based on the highest temperature measured across all neighbouring nodes. Through an extensive experimental evaluation, we have shown that the proposed approaches increase the robustness of existing protocols to temperature variations and significantly improve the performance both on a link basis and on a network level. Our initial tests in the SmartParking deployment in Barcelona revealed that the gateway can reach high temperatures, which makes it important to use TempMAC in such a network. As a next step we should make sure that this solution can be seamlessly integrated with the ones addressing radio interference. A possibility could be to exploit the adaptation of the CCA threshold to achieve a given performance metric; e.g., desired lifetime or delivery rate.

6 Conclusion

In this deliverable, we have described the first integrated prototype and the first integrated experiment. The first integrated prototype combines newly developed or optimized protocols, learning of environmental models and parameters, and run-time assurance. In the first integrated experiment, we have tested this prototype in order to find out whether we can reduce the impact of the environment on network performance, and whether we fulfil the requirements of the selected use case. We have evaluated the performance of the protocols of the integrated prototype when challenged by temperature variations and interference. The experimental evaluation has been conducted using several different testbeds, including two FIRE testbed facilities in Santander, Spain and Berlin, Germany.

Our results are encouraging: the first integrated experiment has shown that the protocols and models developed hitherto within RELYonIT have been able to provide higher and more dependable performance when challenged by temperature variations and interference. As a baseline for our comparison, we have used state-of-the-art protocols available in widespread IoT operating systems such as Contiki and TinyOS. We have also shown that the tools developed within the project to generate controllable and repeatable interference and temperature patterns in testbeds have been successful in replaying traces collected from real deployments.

Lastly, we analyzed the results within the context of the requirements for the selected Outdoor Parking Management (Smart Parking) use case. We found that the performance was for the most part within the bounds of the MUST requirements specified by the industrial partners, but we also identified some points to improve upon. Based on the outcome of this task, we have stated the lessons learned from the experiments and outlined the next steps to be taken for the next iteration on work packages 1-3. The knowledge acquired in this task will be used to implement a refined version of our integrated prototype, which will be evaluated in Task 4.4 *Second Integrated Experiment*.

Bibliography

[1] "Tmote sky: Ultra low power IEEE 802.15.4 compliant wireless sensor module," San Francisco, CA 94105, 2006, available from www.snm.ethz.ch/Projects/TmoteSky.

[2] B. Al-Nahas, S. Duquennoy, V. Iyer, and T. Voigt, "Low-Power Listening Goes Multi-Channel," in *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2014.

[3] Barcelona, "What is 22@ barcelona?" http://www.22barcelona.com/index.php?lang=en.

[4] C. A. Boano, J. Brown, Z. He, U. Roedig, and T. Voigt, "Low-power radio communication in industrial outdoor deployments: The impact of weather conditions and ATEX-compliance," in *Proceedings of the* 1^{st} *International Conference on Sensor Networks Applications, Experimentation and Logistics (SENSAPPEAL)*, Sep. 2009, pp. 159–176.

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

[6] 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)*. IEEE, Apr. 2011, pp. 175–186.

[7] C. A. Boano, H. Wennerström, M. 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.

[8] C. A. Boano, K. Römer, and N. Tsiftes, "Mitigating the adverse effects of temperature on low-power wireless protocols," in *Under Submission*, May 2014.

[9] C. A. Boano, M. Zúñiga, J. Brown, U. Roedig, C. Keppitiyagama, and K. Römer, "TempLab: A testbed infrastructure to study the impact of temperature on wireless sensor networks," in *Proceedings of the* 13^{th} *International Conference on Information Processing in Sensor Networks (IPSN)*, Apr. 2014, pp. 95–106.

[10] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing," *Wirel. Netw.*, vol. 11, no. 4, pp. 419–434, Jul. 2005.

[11] M. Doddavenkatappa, M. C. Chan, and A. Ananda, "Indriya: A Low-Cost, 3D Wireless Sensor Network Testbed," in *Proceedings of the Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TridentCom)*, 2011.

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

[13] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He, "Software-based on-line energy estimation for sensor nodes," in *Proceedings of the IEEE Workshop on Embedded Networked Sensor Systems (IEEE Emnets)*, Cork, Ireland, Jun. 2007.

[14] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He, "Software-based On-line Energy Estimation for Sensor Nodes," in *Proceedings of the Workshop on Embedded Networked Sensor Systems (IEEE Emnets)*, 2007.

[15] J. Flathagen, E. Larsen, P. Engelstad, and O. Kure, "O-CTP: Hybrid Opportunistic Collection Tree Protocol for Wireless Sensor Networks," in *Proceedings of the Workshop on Practical Issues in Building Sensor Network Applications (IEEE SenseApp)*, 2012.

[16] A. Gavras, A. Karila, S. Fdida, M. May, and M. Potts, "Future internet research and experimentation: the fire initiative," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 89–92, 2007.

[17] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection Tree Protocol," in *Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys)*, 2009.

[18] V. Handziski, A. Köpke, A. Willig, and A. Wolisz, "TWIST: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks," in *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality (REALMAN'06)*, 2006.

[19] V. Iyer, M. Woehrle, and K. Langendoen, "Chrysso - a multi-channel approach to mitigate external interference." in *SECON*. IEEE, 2011.

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

[21] R. Szewczyk et al., "Lessons from a sensor network expedition," *Wireless Sensor Networks*, vol. 2920, 2004.

[22] H. Wennerström, F. Hermans, O. Rensfelt, C. Rohner, and L.-A. Nordén, "A long-term study on the effects of meteorological conditions on 802.15.4 links," 2012.

[23] T. Winter (Ed.), P. Thubert (Ed.), and RPL Author Team, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks," Mar. 2012, rFC 6550.

