Research by Experimentation
for Dependability on the
Internet of Things

D-2.1 – Report on Optimized and Newly Designed Protocols

Grant Agreement no: 317826
www.relyonit.eu

Date: May 8, 2014

Author(s) and affiliation: Marco Zúñiga (TUD), Faisal Aslam (TUD), Ioannis Protonotoarios (TUD), Koen Langendoen (TUD), Carlo Alberto Boano (TUG), Kay Römer (TUG), James Brown (ULANC), Utz Roedig (ULANC), Nicolas Tsiftes (SICS), and Thiemo Voigt (SICS).

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

Abstract This deliverable describes the progress made on Tasks 2.1 and 2.2 in the Description of Work. The overarching goal of these tasks is to design networking protocols that can overcome temperature and interference effects. The work in this deliverable builds significantly upon the insights gained in WP1 "Environmental and Platform Models" (for protocol design), and uses heavily the testbeds provided in WP4 "Experimentation and Evalution" (for protocol evaluation). We present four novel protocols and a general method: one protocol targets temperature effects (Temp-MAC), two protocols target interference (MiCMAC and JAG), and one protocol targets both (EverGreen). The general method, called Variable Packet Size, overcomes interference. Temp-MAC, MiCMAC, and JAG have been well evaluated, while Evergreen and Variable Packet Size are work-in-progress.

Disclaimer

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

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

Contents

List of Figures

List of Tables

Executive Summary

Initially, wireless protocols were not required to cope with external interference, but due to the exponential growth of wireless embedded devices, this requirement is now a must. Most wireless networks were designed to overcome *internal* interference, that is, the interference created by their peers in the network. In the last decade, the proliferation of wireless embedded devices has exploded, and with this explosion comes the imperative need to coexist with the *external* interference caused by 'other' networks. This problem is particularly acute in the unlicensed ISM bands where several technologies coexist, such as WiFi, Bluetooth, Zigbee, WirelessHart, to name some.

But external interference is not the only important effect that needs to be considered in future wireless deployments, temperature effects must be considered too. While the research community is aware of the *external* interference problem, and several efforts are underway to overcome its limitations, temperature effects have received marginal attention at best. In fact, one of the key contributions of our consortium has been to identify and quantify the detrimental effects of temperature on low-power wireless devices.

We argue that to have a practical impact, the next generation of wireless protocols needs to be (re)designed to cope with interference and temperature effects.

In this deliverable, we propose four novel protocols to overcome temperature and interference effects. Our first protocol aims at overcoming temperature effects by adapting the operation of the Clear Channel Assessment method according to the temperature profile of the environment (Temp-MAC). Two protocols aim at overcoming interference, one by utilizing a multi-channel approach (MiCMAC) and the other by providing a robust agreement exchange (JAG). The fourth protocol is a work-in-progress that attempts at tackling both effects at once (EverGreen). Two fundamental pillars, regarding the work presented in this deliverable, have been the insights obtained in Work Package 1 (for the design of protocols) and the testbeds developed within our consortium in Work Package 4, as well as the testbeds from the FIRE initiative, (for the evaluation of the protocols).

1 Introduction

During the first months of the RELYonIT project (in WP1), we focused on understanding the effects of temperature and interference on the network's performance. This understanding gave us the necessary background to develop a new set of protocols that overcomes various of the limitations of the state-of-the-art. This deliverable describes the design, implementation and evaluation of four new protocols that overcome interference and temperature effects, and a new method that adjusts the packet size according to the observed interference pattern.

Our protocol-design process followed a two-step approach. First, where possible, we tried to optimize the performance of existing protocols (Task 2.1). Second, if we found that the attempted optimization did not lead to significant improvements, we designed the protocol from scratch (Task 2.2). For the first group of protocols, we present three *optimizations* at the Data Link Layer: one that tackles temperature effects (Temp-MAC), and two that aim at overcoming interference effects (MiCMAC and Packet Variable Size). For the second group, we propose two *new* protocols: one tackling interference at the Data Link Layer (JAG) and the other tackling interference and temperature effects at the Data Link and Network Layers (Evergreen). Below, we provide a brief description of each one of these protocols, and the subsequent chapters describe in detail their design and evaluation.

Before proceeding it is important to highlight two important points. First, for the sake of brevity this deliverable describes only the 'success' stories. That is, we present only those protocols that were successfully optimized or those that were successfully designed from scratch. We do not describe those protocols that we tried to optimize, but whose optimization did not lead to significant improvements. These 'negative' results played a significant role in our learning and design process. In particular, we would like to highlight our efforts in trying to optimize the RPL protocol to improve its performance under temperature effects (Temp-RPL). We did not observe significant gains for Temp-RPL, but this effort lead to a deeper understanding of the problem: we found that the optimization should occur at the MAC Layer and not at the Routing Layer. This important insight resulted in the Temp-MAC protocol described in the next Chapter (Temp-RPL is described briefly in D4.3). The second important point to highlight is the central role of testbeds. It would not have been possible to obtain meaningful and practical results without the testbeds we built in WP4, or without the testbeds available from the FIRE initiative, in particular the TWIST testbed from TU-Berlin. We also enhanced the capabilities of FIRE testbeds by extending our JamLab method to create interference in TWIST (without adding any extra infrastructure), as described in D-4.3.

1.1 Optimized Protocols

1.1.1 Temp-MAC

In this protocol, we analyse in detail the behaviour of communication protocols in the presence of temperature variations and propose techniques to increase their performance. First, we experimentally show that fluctuations of the on-board temperature of sensor nodes significantly reduce the efficiency of data link layer protocols, leading to a substantial decrease in the packet reception rate and to a drastic increase in the energy consumption. Second, we investigate the reasons for such performance degradation, and show that high on-board temperatures reduce the effectiveness of clear channel assessment, compromising the ability of a node to avoid collisions and to successfully wake-up from low-power mode.

We propose two mechanisms to dynamically adapt the clear channel assessment threshold to temperature changes and make data link layer protocols temperature-aware. An extensive experimental evaluation shows that our approaches substantially increase the performance in the presence of temperature variations commonly found in real-world outdoor deployments, with up to 71% lower energy consumption and 194% higher packet reception rate.

1.1.2 MiCMAC

Exploiting multiple radio channels for communication has been long known as a practical way to mitigate interference in wireless settings. In Wireless Sensor Networks, however, multi-channel solutions have not reached their full potential: the MAC layers included in TinyOS or the Contiki OS for example are mostly single-channel. The literature offers a number of interesting solutions, but experimental results were often too few to build confidence. We propose a practical extension of low-power listening, MiCMAC, that performs channel hopping, operates in a distributed way, and is independent of upper layers of the protocol stack. The above properties make it easy to deploy in a variety of scenarios, without any extra configuration/scheduling/channel selection hassle.

We implement our solution in Contiki and evaluate it in the TWIST testbed while running a complete, out-of-the-box low-power IPv6 communication stack (UDP/RPL/6LoWPAN). Our experimental results, presented in D4.3, demonstrate increased resilience to emulated WiFi interference (e.g., data yield kept above 90% when the ContikiMAC drops in the 40% range). In noiseless environments, MiCMAC keeps the overhead low in comparison to ContikiMAC, achieving performance as high as 99% data yield along with sub-percent duty cycle and sub-second latency for a 1-minute inter-packet interval data collection.

1.1.3 Variable Packet Size

The interference models developed in WP1 can be used to provide a better understanding of the achievable Packet Reception Rate (PRR) for a given environment: if we could estimate the *idle* time between two consecutive *busy* periods, then we could adjust the size of the packet to fit in-between the estimated *idle* time. This knowledge can be used to select and configure protocols to mitigate much of the effect that interference causes to deliver the required network performance. For instance, an alternative packet size may be selected based on the interference

level to attain higher delivery targets, or the transmission channel can be changed (based on the existing interference patterns) to always select the least affected channel.

1.2 Newly Designed Protocols

1.2.1 JAG

Wireless low-power transceivers used in sensor networks typically operate in unlicensed frequency bands that are subject to external radio interference caused by devices transmitting at much higher power. Communication protocols should therefore be designed to be robust against such interference. A critical building block of many protocols at all layers is *agreement* on a piece of information among a set of nodes. At the MAC layer, nodes may need to agree on a new time slot or frequency channel; at the application layer nodes may need to agree on handing over a leader role from one node to another. Message loss caused by interference may break agreement in two different ways: none of the nodes uses the new information (time slot, channel, leader) and sticks with the previous assignment, or – even worse – some nodes use the new information and some do not. This may lead to reduced performance or failures.

We investigate the problem of agreement under external radio interference and point out the limitations of traditional message-based approaches. We propose JAG, a novel protocol that uses jamming instead of message transmissions to make sure that two neighbouring nodes agree, and show that it outperforms message-based approaches in terms of agreement probability, energy consumption, and time-to-completion. In contrast to all other protocols, the evaluation of JAG is not included in D-4.3, and hence, we present all the results in this deliverable.

1.2.2 Evergreen

Among the various disruptions caused by temperature and interference phenomena, there is one that is common to both: link quality variability. Interference and temperature affect dramatically the quality of wireless links, which in turn lead to packets losses and delay. From a higher level of abstraction, changes on link quality can be seen as 'link dynamics'. We hence decided to develop a *comprehensive* new protocol that can overcome link dynamics independently of their source, which can be temperature, interference or even mobility. This effort, called Evergreen, is work-in-progress. In this deliverable, we present our initial design and some preliminary evaluations under temperature and interference effects.

2 Protocols Tackling Temperature Effects

2.1 Temperature-Aware MAC [11]

Temperature has a strong impact on the performance of wireless sensor networks. Real-world deployments have shown that the on-board temperature of wireless sensor nodes deployed outdoors can be significantly higher than air temperatures measured by traditional weather stations [54]. Sensor nodes are indeed often exposed to direct sunlight and embedded into airtight packaging absorbing IR-radiation [7], causing the inner temperature in the casing to reach values as high as 70°C [9]. In a long-term outdoor deployment, Wennerström et al. [63] 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 [15], with large heterogeneity across the network [12].

These temperature fluctuations can have a strong impact on clock drift (slowing down processor operations [12] and affecting time synchronization between nodes [4]), as well as on the lifetime of sensor nodes (influencing the capacity and discharge curve of batteries [26, 51] and altering the current consumption of electronic components [42, 60]).

Temperature can also drastically affect the efficiency of low-power wireless transceivers and reduce the quality of wireless links. The performance of low-power radios employed in off-the-shelf wireless sensor nodes is indeed temperature-dependent [3], with a reduction in the strength of the transmitted and received signal at high temperatures. For example, a temperature variation of 40°C can decrease the strength of the received signal by up to 6 dB, with a negative effect on the correct reception of packets [15].

To better study the impact of temperature variations on low-power wireless communications and protocols, we have designed TempLab, 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 [12]. We have shown how this temperature-controlled testbed can be used to systematically analyse the performance of communication protocols, and to precisely characterize the slowdown of micro-controllers [12]. More importantly, we have shown that state-of-the-art communication protocols actually exhibit a lower efficiency at high temperatures. While this lower efficiency may not be surprising (most protocols are developed and tested using indoor testbeds running at a constant temperature), it definitely requires further investigation and calls for solutions to improve the robustness of wireless sensor networks employed in harsh environments.

In this deliverable, we exploit this temperature-controlled testbed to analyse in detail the performance of state-of-the-art communication protocols and to understand (i) *why* their performance decreases in the presence of temperature variations, and (ii) *how* we can mitigate the problem and improve their performance. We first show experimentally that fluctuations of the on-board temperature of sensor nodes reduce the efficiency of carrier sense multiple access data link layer protocols, leading to a substantial decrease in the packet reception rate and to an in-

crease of the energy consumption. We identify reduced effectiveness of clear channel assessment as the reason for such performance degradation, and show 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 clear channel assessment threshold to temperature changes: one based on the temperature measured locally, and one on the highest temperature measured across all neighbouring nodes. We finally show through an extensive experimental evaluation that the proposed approaches increase the robustness of existing protocols to temperature variations and significantly improve the performance also on a network level.

The contributions of this section are hence three-folded:

- **Inefficiency of clear channel assessment.** We describe how temperature variations affect the efficiency of clear channel assessment, and show experimentally that this inefficiency compromises the operations of data link layer protocols based on carrier sense.
- **Augmented data link layer protocols.** After modelling the behaviour of radio transceivers at different temperatures, we implement two strategies that increase the efficiency of clear channel assessment by making data link layer protocols temperature-aware.
- **Extensive experimental evaluation.** We show that our augmented protocols sustain a significantly higher performance than existing protocols, with up to 71% lower energy consumption and 194% higher packet reception rate in the presence of temperature variations commonly found in real-world outdoor deployments.

The next section describes the impact of temperature on low-power radios, and models the attenuation of signal strength on the platform used in our experiments. Sect. 2.1.2 analyses the impact of temperature on data link layer protocols, and highlights the inefficiency of clear channel assessment at high temperatures. In Sect. 2.1.3 we describe two mechanisms to correct this inefficiency and to make data link layer protocols temperature-aware. We evaluate the performance of our approaches in Sect. 2.1.4, showing large performance improvements on a link basis and on a network level. After describing related work in Sect. 2.1.5, we conclude this section in Sect. 4.

2.1.1 Impact of Temperature on Low-Power Radios

Experiences and reports from long-term outdoor deployments have highlighted that temperature has a strong impact on the performance of low-power radio transceivers.

Impact of temperature on link quality. Results by Bannister et al. [3] from an outdoor deployment in the Sonoran desert have revealed that an increase in temperature causes a reduction of the wireless link quality. These results were later confirmed by indoor and outdoor experiments [7, 9], and by a long-term outdoor deployment by Wennerström et al. [63] in Uppsala, Sweden. 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 several months. Fig. 2.1 shows the data collected by two nodes in this deployment: the top figure shows the temperatures measured on-board and the air temperature recorded by a nearby weather station; the other figures show the evolution of a number of link quality metrics over

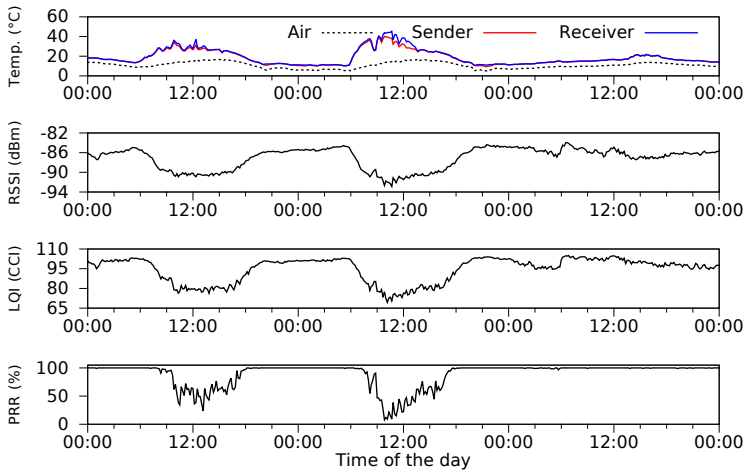


Figure 2.1: High temperatures decrease the performance of low-power radios. In traces from Wennerström et al.'s outdoor deployment [63], we can observe that during daytime (when temperature is high), the received signal strength indicator (RSSI) and link quality indicator (LQI) are lower than during the night. During daytime, also the packet reception rate (PRR) is reduced.

time. Firstly, we can observe that the on-board temperature of the sensor nodes is significantly higher than air temperature: this is very common in outdoor deployments when nodes are enclosed into airtight packaging absorbing IR-radiation. Secondly, we can observe a clear correlation between the on-board temperature of the two nodes and the quality of their link: the higher the temperature, the lower the received signal strength indicator (RSSI) and the link quality indicator representing the chip error rate (LQI).

Dependency between temperature and signal strength. Bannister et al. [3] have shown that the attenuation in received signal strength on the CC2420 radio chip is the result of the decreased efficiency of the transmitter's power amplifier and the receiver's low-noise amplifier at high temperatures. In their experiments in a climate chamber, the authors observed a decrease of 4-5 dB in the output power of the transmitter and a drop of 3-4 dB in the received power over the temperature range 25-65 °C, for a combined effect on received signal strength of 8 dB when both transmitter and receiver are heated.

We have confirmed in later experiments over a larger temperature range [15] that the relationship between temperature and signal strength attenuation is approximately linear, and that this also applies to other radio chips employed in off-the-shelf sensornet platforms. Fig. 2.2 shows the strength of the received signal at different temperatures between two (TelosB-based) Maxfor MTM-CM5000MSP sensor nodes while the transmitter, receiver, or both transmitter and receiver nodes are heated using TempLab [12]. We can notice that the received signal strength attenuation is similar when the two nodes are heated individually (a loss of 0.08 dB/°C[1]), and about twice as high when both nodes are heated at the same time (a loss of 0.17

[1] We estimate the attenuation by computing the slopes of the RSSI curve. Please note that an *exact* comparison between two curves is not possible, as RSSI readings are integer values that depend on the operation of the

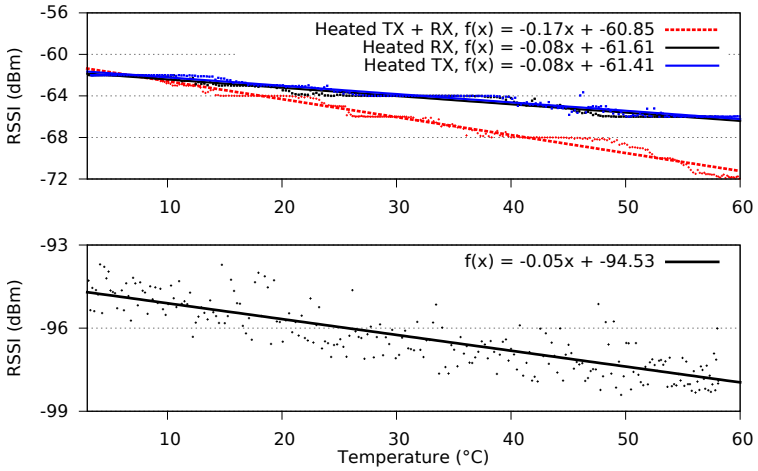


Figure 2.2: Signal strength attenuation as a function of temperature. The top plot shows the received signal strength of packets while transmitter (blue), receiver (black), or both transmitter and receiver (red) are heated: the attenuation is highest when both nodes are heated at the same time. The bottom plot shows the received signal strength attenuation in absence of packet transmissions.

dB/°C). Instead, the noise floor, i.e., the received signal strength measured in absence of radio activity, exhibits a lower variability in the presence of temperature variations.

Impact on packet reception. The attenuation of the signal strength at high temperatures can affect the reception of packets in two different ways. First, a weaker signal is more susceptible to bursts of external interference, and the probability that devices operating at higher powers (e.g., Wi-Fi access points and microwave ovens) corrupt or destroy a packet increases at high temperatures. Second, if temperature increases and the signal strength weakens to values close to the ambient RF noise (often called noise floor), the radio's ability to successfully demodulate a packet drastically decreases. When this happens, a physical limit is reached: the radio cannot correctly receive (most of) the packets that were transmitted, and the connectivity of the link is irreparably compromised. This situation is captured in Fig. 2.1 (bottom). In Wennerström et al.'s deployment, the nodes communicate using Contiki's nullMAC, a data link layer protocol in which the radio remains active all the time and packets are transmitted without first verifying the absence of other traffic. As soon as the received signal strength weakens to values close to the noise floor in the deployment environment (≈ -94 dBm), the packet reception rate (PRR) between the two nodes drops significantly, and the link becomes almost useless during daytime.

In the next section, we focus on carrier sense multiple access data link layer protocols and show that their performance decrease significantly at high temperatures, *but not as a result of the above observations.* The vast majority of duty-cycled MAC protocols *do not* actually reach the physical limit of the radio at high temperatures, and the lower reception rates are caused by design choices that neglect the inefficiency of clear channel assessment in the presence of temperature fluctuations.

automatic gain controller and on the hysteresis between different gain modes [15].

2.1.2 Impact of Temperature on CSMA Protocols

The attenuation of received signal strength at high temperatures described in Sect. 2.1.1 can affect two key functionalities of carrier sense multiple access (CSMA) protocols.

1. *Collision avoidance.* CSMA protocols rely on clear channel assessment (CCA) to determine whether another device is already transmitting on the same frequency channel, and defer transmissions that may otherwise collide with ongoing communications.

2. *Wake-up of nodes.* Duty-cycled protocols typically employ CCA to trigger wake-ups, i.e., to determine if a node should stay awake to receive a packet or whether it should remain in low-power mode.

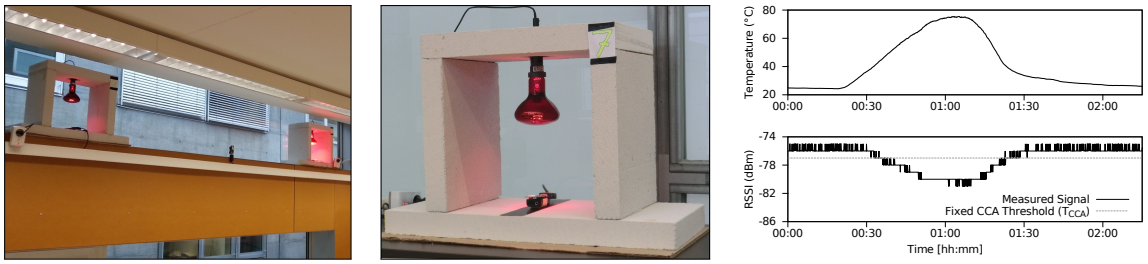
CCA implementations are typically based on energy detection, i.e., on the measurement of the received signal strength and on its comparison with a given threshold. When performing energy detection using a fixed CCA threshold, it is neglected that received signal strength readings are affected by temperature, and this leads to a number of problems. First, the transmitter can erroneously measure a weaker noise in the environment as a result of the increased temperature, and generate wasteful transmissions (see Sect. 2.1.2). Second, a receiver node may not receive a signal sufficiently strong to cause a wake-up of the radio, and constantly remain in low-power mode at high temperatures, causing the disruption of the link (see Sect. 2.1.2). We analyse these issues in the remainder of this section, after describing how CCA is typically implemented in sensornet MAC protocols.

Clear Channel Assessment in Sensornet MAC Protocols

In CSMA protocols, the correct operation of clear channel assessment (CCA) is fundamental to reduce the number of wasteful transmissions and to preserve the limited energy budget of the nodes in the network. The typical task of CCA is to avoid collisions, i.e., to determine whether another device is already transmitting on the same frequency channel. If there are ongoing transmissions, CSMA protocols defer transmissions using different back-off strategies [10], otherwise the packet(s) are immediately sent. CCA is also used in low-power duty-cycled MAC protocols to trigger wake-ups, i.e., to determine if a node should remain awake to receive a packet or whether it should remain in sleep mode [52]. Towards this goal, low-power MAC protocols typically perform an inexpensive CCA check and keep the transceiver on if some ongoing activity is detected on the channel [14, 19, 52].

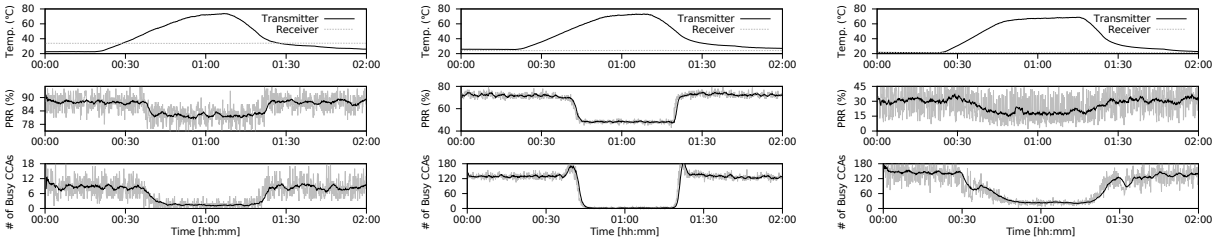
The CCA check can be carried out using energy detection or carrier sense, as described in the IEEE 802.15.4 standard. Energy detection consists in sampling the energy level in the wireless channel and determining whether another device is already transmitting by comparing the measured signal strength with a given CCA threshold T_{CCA}. Carrier sense consists in detecting the presence of a modulated compliant signal, irrespective of its strength. Both options can also be used at the same time: in the case of the widely used CC2420 radio transceiver, this is the default configuration.

Most protocols employ fixed CCA thresholds. When using energy detection, a critical design choice is the selection of T_{CCA}. Whilst sender-initiated, duty-cycling MAC protocols such as B-MAC [52], BoX-MACs [48], and ContikiMAC [19] include energy detection as an important feature to reduce idle listening, there is not yet a widespread practice of tuning the CCA



(a) Overview of our testbed infrastructure (b) IR heating lamp on top of a sensor node (c) Signal strength measured at high temperatures

Figure 2.3: Overview of the testbed infrastructure used in our experiments (a) with infra-red heating lamps on top of each sensor node to control their on-board temperature (b). The received signal strength weakens at high temperatures and can cause an intersection with T_{CCA}, causing several issues (c).



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

Figure 2.4: Temperature affects the efficiency of collision avoidance in CSMA protocols. Our experiments in different interference scenarios show that when the received signal strength weakens to values below T_{CCA} at high temperatures, the PRR decreases, as well as the number of CCAs identifying a busy channel.

threshold at run-time in relation to the noise floor of each network deployment. Rather, the current practice is to rely on the default system settings, i.e., on a *fixed* CCA threshold, which is either set at compile-time, or left untouched so that the default value of the radio device is used instead. The IEEE 802.15.4 standard suggests to use a T_{CCA} that is at most 10 dB greater than the radio's specified receiver sensitivity. Contiki uses the default value for most hardware platforms (CC2420's default threshold is -77 dBm), but did recently set T_{CCA} for TelosB-based platforms to -90 dBm.

Inefficient Collision Avoidance

When a protocol employs a fixed CCA threshold to determine whether another device is already transmitting, it essentially neglects that the strength of the received signal has a dependency on

temperature. We now show experimentally that this can lead to an increase in false negatives when a transmitter is assessing the presence of a busy medium.

Fig. 2.3(a) shows an overview of our testbed, equipped with 18 Maxfor MTM-CM5000MSP nodes. We use TempLab [12] to vary the on-board temperature of the nodes between 25 and 75°C using IR heating lamps (Fig. 2.3(b)). We carry out experiments consisting of several transmitter-receiver pairs running a basic Contiki application, in which the transmitter node periodically 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. In a first experiment in an environment rich of Wi-Fi interference, we use Contiki's nullMAC and nullRDC to avoid protocol-specific implementations and employ the CC2420's default CCA threshold (-77 dBm). Except from temperature, there is no significant change in the environment surrounding the nodes.

Fig. 2.3(c) shows the ambient noise captured using RSSI readings by a node in our testbed. The noise has a visible correlation with the on-board temperature of the node, and follows the attenuation described in Sect. 2.1.1. We can observe that at around 40°C, there is an intersection between the measured signal strength and the selected T_{CCA}. For temperatures lower than 40°C the measured RSSI is above T_{CCA} (and hence transmissions would be deferred); for temperatures higher than 40°C, instead, the RSSI is below T_{CCA} (and packets would be immediately sent). In other words, the MAC protocol erroneously deduces from RSSI readings obtained above 40°C that the channel is free from harmful interference. In reality, the interference in the environment is not weakened by temperature (the RSSI attenuation is only an artefact of the radio), and can still destroy transmitted packets. These erroneous clear channel assessments at high temperature may hence lead to an increase in the number of wasteful transmissions destroyed or corrupted by surrounding interference.

Fig. 2.4 shows the impact of erroneous clear channel assessments in the presence of different interference patterns. We 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 that experienced an intersection between measured noise and T_{CCA} at different temperatures as in Fig. 2.3(c). We can notice that in all scenarios the PRR decreases as soon as the on-board temperature of sensor nodes increases. In the presence of Wi-Fi video streaming, the PRR of the link decreases from 88 to 81% (Fig. 2.4(a)), whereas in the presence of an active microwave oven the PRR decreases from 70 to 45% (Fig. 2.4(b)). Similarly, also the PRR in the presence of a Wi-Fi file transfer decreases from 30 to 18% at high temperatures (Fig. 2.4(c)). We can also notice that the decrease in packet reception is correlated with a decrease in the number of clear channel assessments identifying a busy channel, i.e., with a decrease in the efficiency of clear channel assessment that does not identify potential collisions at high temperatures. These results prove our assumptions, and show that the intersection between the RSSI curve and the CCA threshold shown in Fig. 2.3(c) results in erroneous clear channel assessments leading to an increased packet loss rate at high temperatures.

Unsuccessful Wake-Up of Nodes

State-of-the-art MAC protocols often duty cycle the radio to reduce energy consumption, and employ clear channel assessment to wake-up the transceiver from sleep mode. Typically, a periodic CCA check is performed: if the channel is busy, the transceiver is kept on in order to receive the incoming packet, otherwise the radio remains in sleep mode.

High temperatures can affect the correctness of this mechanism. Imagine a sender A and a receiver B exchanging packets using a duty-cycled MAC protocol in which A sends short strobes before the actual packet (or repeatedly sends the same packet). If B receives the strobes from node A with a signal strength that is higher than T_{CCA}, it keeps its radio on and receives the payload message from A. If temperature increases, the received signal strength at node B may intersect T_{CCA} as shown in Fig. 2.3(c). When this happens, the transmissions from A are received with a strength lower than T_{CCA}, and B does not wake up to receive A's packets anymore, essentially disrupting the link. In the case shown in Fig. 2.3(c), the link would be disrupted for temperatures higher than 40°C, because node B would not wake up when the strength of the received signal from A decreases below T_{CCA}.

We now show experimental evidence of this problem. We let several transmitter-receiver pairs of nodes communicate using ContikiMAC, Contiki's default MAC protocol in which nodes sleep most of the time and periodically wake up to check for radio activity. In ContikiMAC, the transmitter sends repeatedly the same packet until a link layer acknowledgement (ACK) is received, whereas the receiver keeps its radio on as soon as a packet transmission is detected by means of a single CCA check [19]. Packets are exchanged every 20 seconds, and ACKs are sent using CC2420's hardware support. As in the previous experiment, we use TempLab to warm-up and cool-down the on-board temperature of the nodes, emulating the daily fluctuations that can be found in real-world deployments.

Fig. 2.5 shows an example of link disruption caused by a receiver not waking up at high temperatures. We can notice that what was a perfect link until approximately 54°C (essentially, no packet was lost), suddenly does not receive any packet at higher temperatures. Only once temperature decreases below 54°C, the link is restored and the node correctly receives the packets sent from the transmitter. This behaviour can significantly harm network performance, as links may disappear during the hottest times of the day, leading to high latencies, drastic topology changes, or in case no alternative paths for communication can be found, to a complete disconnection of some nodes from the network.

Please note that a receiver node can in principle detect a packet transmission using carrier sense, i.e., by identifying a valid sequence of bits without comparing if the received energy is above a given threshold. However, in off-the-shelf radio transceivers such as the CC2420, a valid sequence can be identified only prior detection and validation of the start frame delimiter. Therefore, carrier sense is ineffective when used in duty-cycled systems that periodically wake up and perform a single CCA check (in a non-duty-cycled protocol such as Contiki's nullRDC or nullMAC, instead, carrier sense would work well, as the radio remains always active). Indeed, despite the CC2420 radio is using by default a combination of carrier sense and energy detection, ContikiMAC experiences a complete loss at high temperatures that is dependent on T_{CCA}, i.e., on the chosen energy detection threshold.

It is also important to highlight that selecting by default a low CCA threshold is not a valid solution: the lower T_{CCA} the higher the number of activities in the channel (radio interference,

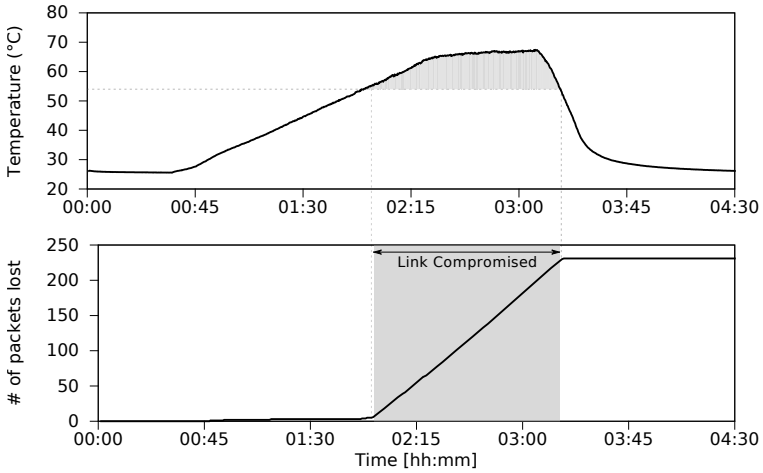


Figure 2.5: Temperature can affect the wake-up mechanism in duty-cycled MAC protocols. When the strength of the received signal from a transmitter weakens at high temperatures and intersect the CCA threshold as shown in Fig. 2.3(c), the receiver does not wake up anymore, disrupting the link's connectivity.

communications from surrounding nodes) that will trigger a wake-up and, consequently, a higher energy consumption. Indeed, selecting T_{CCA} close to the noise floor in a noisy environment, would essentially cause the radio to be constantly active, with a highly suboptimal energy expenditure.

2.1.3 Designing Temperature-aware MAC Protocols

Whenever a link delivers poor performance, it is typically the network layer's task to maintain connectivity and search for alternative routes that can sustain a high delivery rate. Using link quality estimation, the network layer can indeed filter out lossy links and pick a better topology, i.e., select a network configuration that avoids links that are asymmetric or that have a signal that is too weak to communicate reliably, as well as links that are negatively affected by temperature variations. The network layer, however, can only be effective if the network is sufficiently dense to offer link redundancy. Very often, there are no available links in the surroundings of a node offering better performance, especially in sparse networks. In such cases, the network layer is obliged to make use of lossy links, and cannot mitigate the impact of temperature variations on the lower layers of the protocol stack.

To mitigate the inefficiency of CSMA protocols at high temperatures shown in Sect. 2.1.2, we hence need to tackle the problem directly at the MAC layer. A link can indeed still offer good performance if the CCA threshold is dynamically adapted to the on-board temperature variations of the nodes. In this section, we propose two alternatives to achieve this goal.

Predicting the Attenuation of Signal Strength

In order to dynamically adapt T_{CCA} to temperature variations, we first need to model the relationship between signal strength attenuation and temperature. In Sect. 2.1.1 we have shown that the latter is approximately linear, and that there are two components that need to be considered: the attenuation on the receiver side, and the one on the transmitter side. Imagine a sender A and a receiver B exchanging packets. If the on-board temperature of B varies by ΔT_B degrees w.r.t. to an initial temperature τ, the signal will suffer an attenuation on the receiver side by $R = \beta \Delta T_B$, with ΔT_B being the difference between B's current temperature T_{now} and τ. Similarly, if the on-board temperature of A varies, its signal will be transmitted with an attenuation on the transmitter side of $T = \alpha \Delta T_A$. In case the temperatures of both A and B vary, the overall attenuation of the received signal strength on B is given by $R + T$. Please notice that if temperature has decreased, $\Delta T = (T_{now} - \tau)$ is negative, and R and T are not an attenuation, but instead a strengthening of the signal.

The parameters α and β are specific to the employed radio and differ only in a negligible way among different instances of the same chip. Hence, they can be characterized following the same approach shown in Sect. 2.1.1: using a pair of nodes that can be heated individually, we compute the variation of signal strength on a large temperature range and derive the slope of the RSSI curves of transmitter and receiver for a given platform [15]. In the case of the Maxfor nodes employed in our experiments we derive from Fig. 2.2 $\alpha = \beta = -0.08$. We further model the attenuation of the noise floor as $N = \gamma \Delta_T$ (which is typically smaller than R and T) and derive $\gamma = -0.05$.

Adapting the CCA Threshold at Runtime

Exploiting the above model, we can now adapt the CCA threshold at runtime. Each node needs to compute if temperature varied significantly enough to cause an attenuation of the signal strength w.r.t. an initial threshold T'_{CCA}. As we mentioned in Sect. 2.1.2, the default energy detection threshold is typically fixed. However, as nodes are typically uncalibrated and have radio irregularities, a good practice would be to select $T'_{CCA} = n'_f + K$, with n'_f being the noise floor of the node, and K a constant defined at compile time. If this is the case, T'_{CCA} and n'_f are typically computed during the start-up phase while the node experiences an on-board temperature τ. If T'_{CCA} is fixed, we assume $\tau = 25°\text{C}$ (ambient temperature). The updated threshold is then computed as $T_{CCA} = T'_{CCA} + T + R$, with T and R being computed using the difference between the current temperature and τ. We apply to the computation of T_{CCA} a lower bound $n_f + C$ (with $n_f = n'_f + N$) that avoids the selection of CCA thresholds that are too close to the noise floor (this would cause the radio to assess the channel as constantly busy and to continuously wake-up).

Obtaining up-to-date temperature measurements. All that is needed to adapt the threshold is hence an up-to-date information about the current on-board temperature of the nodes and the initial temperature τ stored in a 2-byte variable. Almost every off-the-shelf sensornet platform comes with an embedded temperature sensor. TelosB-based platforms carry the SHT11, a digital temperature and humidity sensor. Other platforms do not have a dedicated sensor, but several micro-controllers such as the MSP430 offer the possibility to obtain a rough estimate of the on-board temperature from a built-in temperature sensor using a specific input of the

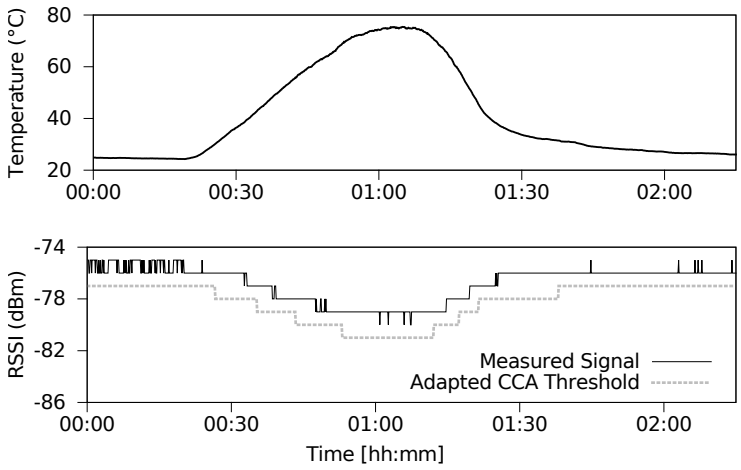


Figure 2.6: Dynamic adaptation of the CCA threshold based on the temperature measured locally on the node: T_{CCA} follows the attenuation of the signal, avoiding an intersection with the RSSI curve (in contrast with Fig. 2.3(c)).

analog-to-digital converter. By periodically sampling the on-board temperature, a node can hence compare its current temperature with τ and compute Δ_T. It is important to stress that the temperature sensor should be physically on the board, to get an estimate as close as possible to the temperature of the radio chip: external sensors measuring air temperature outside the packaging may not give a sufficiently accurate estimation.

Deriving the on-board temperature of the transmitter. Whilst N and R can be immediately derived based on the current on-board temperature, to derive T a receiver needs to know the temperature of the transmitter. This is a fundamental problem, as a receiver does not necessarily know the identity of the sender, and it may actually be recipient of packets sent by different nodes. A transmitter could piggyback the information about its temperature in the link-layer acknowledgements, but the latter are often exchanged using the hardware support of the radio and cannot be modified. Another option consists in assuming $T = R$: nodes in close proximity to each other may be experiencing the same temperature. However, this may not lead to accurate results: although intuitively there is a high likelihood that two nodes deployed close to each other have similar temperatures, real-world deployments have shown that there can be high gradients (more than 30°C) even across spatially close nodes [4, 12] because of cloud obstructions or shade from trees or buildings in the surroundings.
The information about the transmitter's temperature can be actually derived from the network layer, which by default keeps a neighbour table storing the addresses of the nodes in the surroundings, and could add an additional entry containing the information about the latest on-board temperature for each neighbour. This, however, requires a modification of the routing layer which may not be suitable in some applications. We hence propose two different adaptation mechanisms: one that only adapts T_{CCA} based on local temperature measurements, and one that exploits a cross-layer approach to derive T.

Local adaptation. A first approach adapts T_{CCA} based on local temperature measure-

ments only, i.e., it fixes $T = 0$ (an alternative would be to select $T = R$). In this case, $T_{CCA} = T'_{CCA} + R$, with a lower bound $n_f + C$. We found in our experiments that values of C lower than two trigger a continuous wake-up of the radio, and we therefore use $C = 2dBm$ in our implementation. Fig. 2.6 shows the adaptation of the CCA threshold based on the algorithm detailed previously. In this case, we replicate the setup of Sect. 2.1.2 and heat a receiver node while measuring the strength of the signal in an environment rich of Wi-Fi interference. If we compare the results with the ones shown in Fig. 2.3(c), we can immediately notice that the CCA threshold follows the same attenuation as the received signal, avoiding an intersection between the RSSI curve and T_{CCA}. This shows that the proposed model is sufficiently accurate to dynamically adapt the CCA threshold to local temperature changes. However, if the receiver's temperature does not vary significantly and there is a relevant increase of the on-board temperature, performance may still decrease at high temperatures.

Cross-layer adaptation. To prevent this, we propose an approach that allows the CCA adaptation mechanism to make more informed decisions by using temperature information from the neighbours. Our cross-layer adaptation uses existing routing beacons to piggyback temperature information efficiently. We implement this by using RPL, the standard IPv6 routing protocol for low-power and lossy networks [64]. Whilst we have chosen RPL because it is a standard protocol and several open-source implementations exist, we also note that it would be simple to disseminate the information at the application layer, albeit with a slightly higher energy cost.

We disseminate the temperature information by piggybacking it on RPL's routing beacons. RPL sends these beacons to the neighbour nodes with quickly increasing time intervals, as regulated by the Trickle algorithm [41]. Within the DODAG Information Object (DIO), there is room to embed a routing metric container object, which holds different parameters and constraints that are used to take routing decisions. Beside the metric container specified in the standard, it is possible to use implementation-defined metric containers. Hence, we make each node report its current temperature and maximum temperature through such a metric container. Once a node receives this information in an incoming routing beacon, it stores it as an attribute in Contiki's neighbour table, from whence it can be retrieved by the CCA adaptation module to calculate the maximum temperature in the neighbourhood.

2.1.4 Evaluation

The evaluation of TempMAC is presented in D4.3 and it shows that our approach alleviates the collision avoidance and wake-up problem in CSMA protocols. TempMAC can reduce the energy consumption of nodes by up to 71% and increase the packet reception rate by 194%.

2.1.5 Related Work

Several outdoor deployments and experimental studies have highlighted the impact of temperature on the quality of communications in wireless sensor networks. Bannister et al. [3] have reported that high temperatures can decrease the strength of the wireless signal. Wennerström et al. [63] have found experimental evidence of this problem on a long-term outdoor deployment. Boano et al. [9] have shown that the transmission power of communications at low temperatures can be safely decreased without deteriorating the performance of the network, and have

precisely characterized the attenuation in received signal strength on different platforms [15]. All these works, however, simply report the degradation of the wireless signal as a consequence of an increase in temperature and do not provide a deeper analysis of what the implications are on communication protocols when operating a network outdoors.

Keppitiyagama et al. [35] have presented a poster showing that network protocols are affected by temperature and proposed to enhance them with temperature hints. In our earlier work, we have presented TempLab, a testbed infrastructure to study the impact of temperature on communication protocols [12], and used it to confirm the low performance at high temperatures. In this work, we exploit this testbed infrastructure to analyse *why* communication protocols are affected and propose *how* to mitigate the problems by dynamically adapting the CCA threshold to temperature variations.

Several works have suggested the use of adaptive CCA thresholds to mitigate interference. Bertocco et al. [45] have provided hints for an optimal threshold selection in the presence of in-channel additive white Gaussian noise interference. Yuan et al. [68] have proposed to adjust the CCA threshold in the presence of heavy interference to reduce the amount of discarded packets due to channel access failures. Xu et al. [67] have designed a mechanism that dynamically adjusts the CCA threshold to enable concurrent transmissions on adjacent non-orthogonal channels and achieve high throughput.

Sha et al. [46] have studied the effects of the CCA threshold setting in noisy environments, and shown that interference can increase the number of false wake-ups in low-power-listening MAC protocols. To remedy this problem, they have proposed AEDP, an adaptive protocol that adjusts the CCA threshold in response to changes of ETX. While we share the idea that the CCA threshold cannot be set to an arbitrary value at compile-time, there are considerable differences with our work. First, AEDP is designed to achieve a desired performance in noisy environments and does not take into account the role of temperature. This may lead to problems, as AEDP requires an estimate of the noise floor and of the average RSSI value of all incoming links, which may change as temperature changes. Second, AEDP does not require a temperature model to adapt the CCA threshold, but instead requires information of observed interference in recent packet transmission attempts. In event-based networks, the reliance on ETX values may be a problem since packet transmissions are sparse.

An alternative approach to mitigate the impact of temperature may consist in increasing the transmission power at high temperatures, as suggested by the data-sheets of some radio chips. Although this would lead to an increased energy-consumption, it may simply not be possible: a node could already be using its highest power level. Furthermore, increasing the power based on the local temperature would only make the transmitted signal stronger, but would not solve the attenuation on the receiver side. Hence, our approach based on the signal strength attenuation modelling is more generic and effective.

3 Protocols Tackling Interference Effects

3.1 MiCMAC

Wireless Sensor Networks share their radio medium with other ambient technologies, such as WiFi, Bluetooth, low-power radios (*e.g.*, 802.15.4), or even microwave ovens [2, 10]. Dealing with such interference is of utmost importance in order to attain the quality of service required by a given application, in reliability, energy, and latency. In the IEEE 802.15.4 PHY standard, 16 independent channels are provided – some colliding with the WiFi spectrum and others disjoint from it.

Using multi-channel MAC layers (as the Bluetooth standard does for example) has been long known as a practical and efficient way to operate in noisy environments [61]. In addition to wireless interference, the nature of radio propagation and multi-path fading phenomenon cause challenging link dynamics that affect the signal strength and packet reception rate in relation to a number of parameters; namely, the used frequency, the shape of the wireless path, the objects standing/moving in the path and the location of the transceiver [62].

Although many studies showed the potential of multi-channel in 802.15.4 [61, 62], and in spite of many MAC layers available in the literature, the sensor networking community is struggling to adopt multi-channel. This is reflected by the default MAC layers in the two mainstream operating systems, TinyOS and Contiki, all being single-channel. A possible explanation to this is that existing solutions are either too complex, require ideal scheduling of transmissions, or are difficult to implement and use.

The IEEE 802.15.4-e amendment [1], published in 2012, tackles this issue and proposes a number of channel hopping solutions. TSCH for example, uses TDMA and channel hopping and schedules transmissions along two dimensions: time and channel. TSCH is extremely promising in terms of possible performance and energy gains, but connecting it to upper layers of the communication stack is non-trivial. For instance, using TSCH in IPv6-based scenarios raises a number of challenges, that led to the creation of the IETF Working Group 6TiSCH to tackle this single issue. 802.15.4e also proposes CSL, a low-power listening MAC that performs channel hopping. Low-power listening MAC layers are interesting in that they require zero configuration and emulate always-on links while having the nodes sleep most of the time. State-of-the art low-power listening solutions such as BoXMAC or ContikiMAC can be easily deployed in large networks, performing multi-hop routing while sleeping more than 99% of the time [21].

We argue that extending low-power listening with channel hopping is an effective and practical solution to mitigating interference in low-power, multi-hop networks. To this end, we design MiCMAC, a channel hopping variant of ContikiMAC. MiCMAC has a design similar to CSL – both MAC layers were in fact designed simultaneously and along the same principles. Both are based on low-power listening and have nodes wakeup periodically on different channels.

We implement MiCMAC in Contiki and validate it experimentally in the 97-node testbed

Indriya [17]. We run a full low-power IPv6 stack including 6LoWPAN and RPL on top of MiCMAC, demonstrating that our approach is practical and independent from other layers in the protocol stack. To the best of our knowledge, we present the most thorough experimental validation of multi-channel low-power listening in WSN.

Our experimental results show that on a noiseless channel MiCMAC achieves high performance, close to that of ContikiMAC. We attain end-to-end delivery ratios of 99% while keeping the radio duty cycle below 1% and the packet latency below 1 second. We compare to an integrated multi-channel data collection solution, Chrysso [30], and show that MiCMAC (with RPL) outperforms it in delivery ratio, duty cycle and latency. We also run experiments where we inject controlled interference to demonstrate the ability of MiCMAC to deal with losses and continue operating in bursty environments.

3.1.1 Related Work

Multi-channel communication has potential benefits for wireless networks that possibly include: improved resilience against external and internal interference, enhanced reliability, reduced latency, and increased throughput [55]. Moreover, frequency diversity implemented by frequency-hopping is suggested to mitigate the effects of multipath fading [62]. In this section, we review a selected set of existing low-power multi-channel MAC protocols.

A number of multi-channel solutions for low-power sensor networks focus on the issue of reducing interference between nodes and improving throughput. However, most of these works allocate fixed channels to data collection trees [44] or sub-trees [65], a practice that is not only difficult to coordinate over multiple hops, but also that does not handle the issue of *localized* interference within a network. An exception is the work by Le *et al.* [40] that allows nodes to independently switch channels based on observed channel contention. However, the protocol design features specific policies for data aggregation networks alone, as opposed to the predominant class of data gathering WSNs. Multi-channel protocol such as MC-LMAC [29], Y-MAC [36], MuChMAC [13] and EM-MAC [58] typically allow nodes to switch channels independently of one another. MC-LMAC [29], Y-MAC [36] are inherently TDMA-based, which entails a need for time synchronization between nodes. In contrast, MuChMAC [13] and EM-MAC [58] facilitate asynchronous channel access with a pseudo-random channel hopping sequence on every node. Nodes execute a lightweight time synchronization primitive to communicate with each other efficiently. Specifically, EM-MAC introduces interesting features such as channel black-listing, clock-drift estimation and correction. However, these features make the rendezvous procedure between nodes more difficult, requiring neighboring nodes to discover each other before proceeding to broadcast. A noteworthy observation in the aforementioned works is the lack of a routing solution over multiple channels. Furthermore, in most cases, the experimental evaluation is restricted to networks comprising less than 20 nodes. In contrast, large networks of up to 100 nodes are witnessed to increased channel contention and message collisions, which raises a concern of protocol scalability.

Chrysso [30] is a multi-channel solution that is specifically designed for mitigating external interference in data collection WSNs. Chrysso supposes that the network is formed as a tree with a sink node, parent nodes and children nodes. Each parent uses two channels for inbound and outbound communication with children, and decides to hop either of the channels when the channel quality degrades. Deviating from other related multi-channel protocols, Chrysso

interfaces to the routing layer with an additional *scan* procedure that facilitates neighborhood discovery over multiple channels. The core of Chrysso's functionality comprises a set of channel switching policies that interface to both the MAC layer (*i.e.* X-MAC) and the network layer (*i.e.* Collect). However, the specific allocation of in and out-channels restricts its applicability to data collection networks alone. In contrast, MiCMAC is suited to general purpose applications and 6LoWPAN network stack as it does not suppose a structure of any kind for it to operate.

The IEEE 802.15.4e amendment to the original 802.15.4 standard introduces a number of multi-channel MAC layers, including TSCH and CSL. TSCH (Time Synchronized Channel Hopping) employs TDMA and channel hopping such that it schedules communications in two dimensions: time and frequency. TSCH promises high reliability but has the drawback that it requires schedules to operate. Defining a schedule that copes with the dynamic nature of wireless communication and the bursty IP traffic is a great challenge. In contrast, CSL follows an unscheduled low-power listening approach. In CSL, nodes periodically wake up to sense the radio medium and hop the channel in an increasing order every wakeup. Senders need to send long wakeup strobes before transmitting the actual packet, but they can learn the receiver wakeup schedule later on from the information included in acknowledgments. MiCMAC employs a similar overall design with a few exceptions such as that we use the actual data frame as the wakeup strobe, and we employ pseudo-random channel hopping sequences. Furthermore, we are not aware of any large-scale evaluation of the CSL MAC (related experiments are limited to a handful of nodes [5]), while we provide a practical implementation and thorough evaluation of MiCMAC.

3.1.2 Design of MiCMAC

This section covers the design of MiCMAC, a multi-channel low-power listening MAC for WSNs. MiCMAC inherits its basic design from ContikiMAC [19] and extends it to for efficient multi-channel support.

3.1.3 Overview

Since ContikiMAC proved to be very efficient in the single-channel case [21, 38], we choose to inherit its design and integrate channel hopping in it.

We can summarize the steps for communication between two nodes in: (1) medium access; (2) finding receiver's wakeup-time and channel; (3) data transmission and acknowledgment; (4) and dealing with losses/collisions. Moreover, we need to take care of selecting wakeup channels and maintaining wakeup time and channel for future communication with the same receiver.

Idle nodes, which do not have packets to send, keep their radios off for most of the time, and wake up periodically to sense the radio with two short channel clear assessments (CCA) spaced carefully to avoid falling in the inter-frame period. The wake-up period is constant and shared by all nodes. Each time a node wakes up to listen, it *hops* (switches) channel according to a pseudo-random sequence. When a node detects activity on the channel through CCA, it keeps the radio on for a longer time trying to receive a potential frame. Only if a frame is received correctly, the node sends an acknowledgment frame; then, it goes back to sleep.

If a node S has a packet to send to a node R, it needs to know the wake-up time and channel of R. Assuming that it already has this information for all neighboring nodes (described

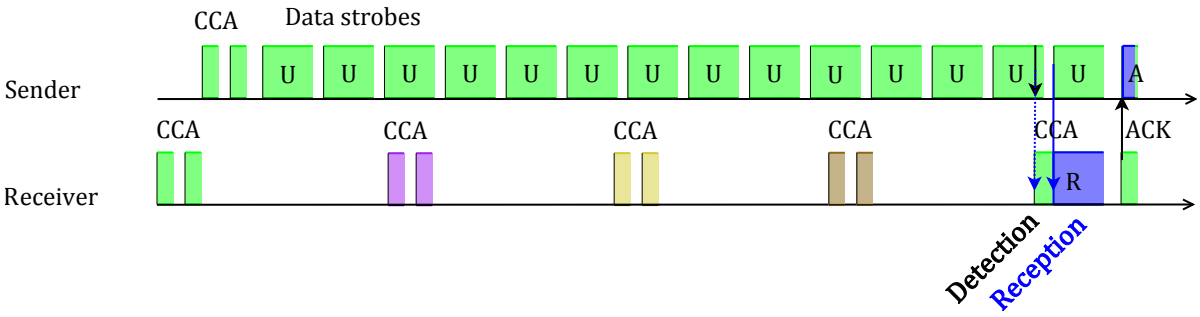


Figure 3.1: MiCMAC Initial Rendezvous (4 Channels). The sender strobes over one available channel until it receives an acknowledgment, for a maximum of 4 consecutive wakeup periods. The receiver wakes up periodically to sample the channel with two short CCA. It hops through all available channels according to its own sequence. In the figure, different colors signify the use of different channels, with the exception that blue means reception.

in more details later), S schedules the packet for sending just before R's expected wake-up, switches to R's expected channel, samples it to ensure it is clear, sends the packet and waits for acknowledgment (ACK). If S receives the ACK, it knows that communication was successful; thus, it updates its information of R's wake-up time and channel and goes back to sleep. Otherwise, S retries the same steps. After a number of failed retries, S assumes that its information of R's wake-up time and channel is wrong and needs to be updated.

Frequency Hopping

The choices made in this step affect the design of other parts of MiCMAC; specifically, channel rendezvous and broadcast. Each node switches its channel periodically on every wakeup cycle following a pseudo-random sequence. We generate the pseudo-random channel numbers using a *Linear Congruential Generator* (LCG) [37]. We choose this kind of generators because the sequences they generate are uniformly distributed and they are computationally simple. With LCG, the pseudo-random sequence X is defined as:

$$X_{n+1} = (aX_n + c) \bmod N, \quad n \geq 0$$

where N –the modulus– is the total number of available channels, X_0 is the seed ($0 \leq X_0 < N$), a is the multiplier ($0 \leq a < N$), and c is the increment ($0 \leq c < N$).

We obtain the actual channel numbers from this sequence by adding the first channel to X, *i.e.*, 11 in the case of IEEE 802.15.4.

The properties of the pseudo-random sequence depend on the chosen parameters: a, c, N. We select these parameters such that the generated sequences appear random and *contain each possible number in the range exactly once* before repeating the whole sequence again (as described by Knuth [37]). We use this property to our advantage when we want to find a node's wakeup channel. Note that any generated sequence will be of length N. However, we can combine several of these sequences to generate one longer sequence.

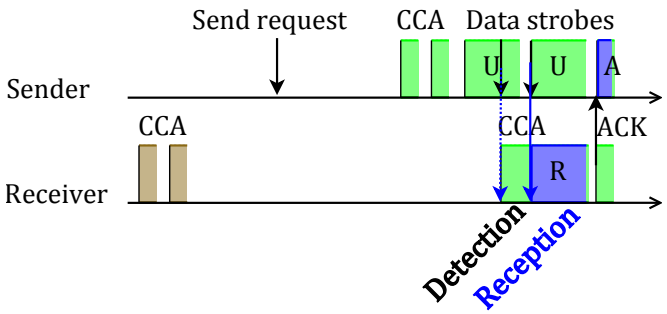


Figure 3.2: MiCMAC Channel-locked Transmission. The sender anticipates both the phase and channel of the target node's next wakeup, making the strobing shorter (saves energy and bandwidth).

We assign each node in the network one sequence which is parametrized with a set of tuples $\{< a, c >\}$, thus, the length of each hopping sequence will be $\|\{< a, c >\}\| \times N$. The choice to use one short hopping sequence (*i.e.*, of length N) or a long sequence (*i.e.*, formed from a combination of sequences) affects the initial channel rendezvous and broadcasts, as explained in the next subsection.

We choose to do blind channel hopping because of simplicity in the first place; as local blacklisting would involve some overhead for synchronizing the blacklists among neighbors. Secondly, previous work has shown that even random blind channel hopping improves network connectivity, efficiency and stability when compared to single-channel [61].

Unicast Transmission and Channel-lock Mechanism

Sending unicasts requires a continuous transmission of preambles –which are copies of the data frame in our case– until the receiver wakes up, receives and acknowledges the frame. To make this process more efficient, ContikiMAC has a *phase-lock* mechanism, where nodes learn their neighbor's schedule in order to anticipate their wakeup for the next transmission. We extend ContikiMAC's phase-lock with a *channel-lock* to anticipate the wakeup channel of the target node as well.

Initial Rendezvous When communicating with a neighbor for the first time, the sender picks any channel and transmits strobes repeatedly for a maximum of W wakeup periods, where $W = N$ the number of channels when using hopping sequences of length N, or $W = 2N - 1$ when using a long hopping sequence. Doing so guarantees that an idle receiver will wake up exactly once on the channel where the strobing occurs, getting one opportunity to receive and acknowledge the frame. The sender waits for a short period of time between strobes to allow the ACK to be received. Figure 3.1 illustrates the initial rendezvous for unicast in the case of four channels (and with a sequence of length 4).

Phase- and Channel-lock Upon successful unicast reception, the receiver sends an ACK frame that includes the pseudo-random generator parameters a, c, X_0 so that the sender can compute the next wakeup channels. The sender stores the time and channel of reception of the ACK,

respectively for phase- and channel-lock. Next time the same pair of nodes communicates, the sender will (1) calculate the next wakeup time, using unmodified ContikiMAC phase-lock, and (2) calculate the next wakeup channel, by generating the receiver's next wakeup channel; taking into account the number of periods elapsed since the last successful unicast. This saves the sender from the long strobing incurred in initial rendezvous. However, if the transmission fails for a few subsequent tries, the sender repeats the initial rendezvous. Figure 3.2 illustrates channel-locked unicast transmissions.

Broadcast Support

In ContikiMAC, broadcasts are supported by transmitting non-acknowledged frames repeatedly for one wakeup period. This gives the opportunity to every neighbor to receive the frame exactly once.

To make broadcast transmissions possible in a channel hopping scenario, we devise two variants of MiCMAC:

MiCMAC We provide basic support for broadcast in MiCMAC by strobing only one of the possible channels continuously for N times the wakeup period (or $2N-1$ in the case of long sequences). This is similar to the initial rendezvous for unicasts, but done with non-acknowledged frames, as illustrated in Figure 3.3. The downside of this design is the increased cost in energy and increased channel use, especially for large N (many channels used).

MiCMAC-BC We provide an alternative solution where nodes wake up on a dedicated broadcast channel at every period, in addition to their baseline wakeup on the unicast pseudo-random channel. Broadcast transmissions are always done over this channel for a duration of only one wakeup period, as illustrated in Figure 3.4. The downsides are (1) reduced robustness as all broadcast occur on the same channel and (2) increased baseline, where two wakeups are needed instead of one at every period. This design can be extended with channel hopping for the broadcast channel, where the number of channels used for broadcast would be lower than that used for unicast, resulting in a trade-off between MiCMAC and MiCMAC-BC.

Miscellaneous Optimizations

Always-on Nodes In order to reduce reception latency for nodes that are always on (such as the border-router), we do not use channel- and phase-lock when sending to them. Instead, these always-on nodes change the channel more frequently (we use a period of 10 ms, which can accommodate up to two full 802.15.4 frames). Nodes wishing to send to them simply pick any channel and start transmitting as early as possible if the channel is clear.

Use of Predefined Hopping Sequences Instead of calculating the hopping sequences at runtime, we provide a static table of all sequences used in the network. Each node simply selects its sequence according to its MAC address.

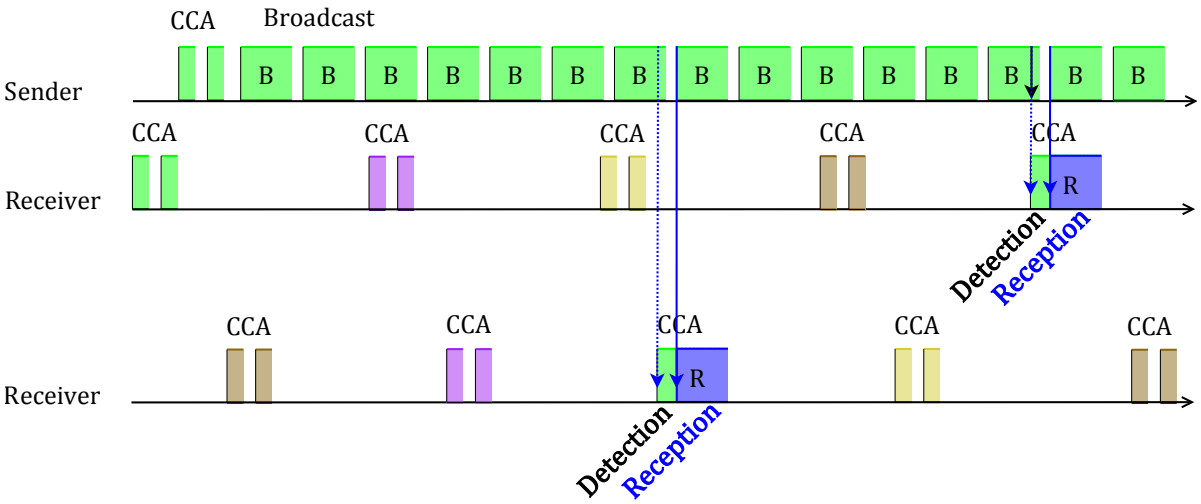


Figure 3.3: MiCMAC Broadcast (4 Channels). The sender strobes over one channel for exactly 4 wakeup periods without expecting acknowledgments.

Use of Short Hopping Sequences Using short hopping sequences (of size N, instead of long sequences formed of several short ones), relieves the receiver from including the channel index in ACK frames. The sender identifies the receiver's sequence uniquely, based on the receiver's MAC address. It can then infer the channel index by searching for the current send channel in the receiver's sequence, since each sequence contains every possible channel exactly once.

3.2 JAG

Wireless low-power transceivers used in sensor networks typically operate in unlicensed frequency bands that are subject to external radio interference caused by devices transmitting at much higher power. Communication protocols should therefore be designed to be robust against such interference. A critical building block of many protocols at all layers is *agreement* on a piece of information among a set of nodes. At the MAC layer, nodes may need to agree on a new time slot or frequency channel; at the application layer nodes may need to agree on handing over a leader role from one node to another. Message loss caused by interference may break agreement in two different ways: none of the nodes uses the new information (time slot, channel, leader) and sticks with the previous assignment, or – even worse – some nodes use the new information and some do not. This may lead to reduced performance or failures.

We investigate the problem of agreement under external radio interference and point out the limitations of traditional message-based approaches. We propose JAG, a novel protocol that uses jamming instead of message transmissions to make sure that two neighbouring nodes agree, and show that it outperforms message-based approaches in terms of agreement probability, energy consumption, and time-to-completion. We further show that JAG can be used to obtain performance guarantees and meet the requirements of applications with real-time constraints.

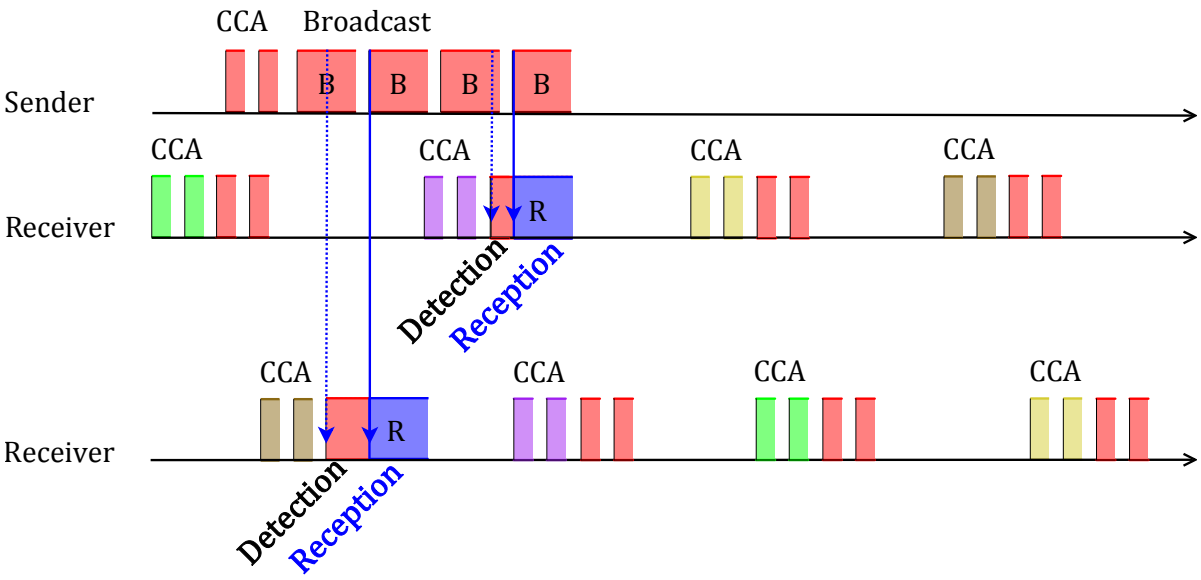


Figure 3.4: MiCMAC-BC Broadcast (4 Channels). The sender strobes over a dedicated broadcast channel for only one wakeup period. Receivers check both their current unicast and the broadcast channel at every wakeup.

3.2.1 Introduction

Wireless sensor nodes often need to agree on fundamental pieces of information that can drastically affect the performance of the entire network. For example, sensor nodes may need to agree on handing over a leader role from one node to another. An agreement failure would break the leader election, leading to a situation in which either more than one node becomes leader, or no leader is selected, causing reduced performance or failures in the network [57]. Similarly, at the MAC layer, several state-of-the-art protocols use time division multiple access (TDMA) or frequency diversity techniques to optimize their performance, in order to maximize network lifetime and minimize battery depletion. In such protocols, vital information such as the TDMA schedule, the channel-hopping sequence derived by interference-aware protocols, or the seed used to regulate the random channel hopping, need to be agreed upon by two or more sensor nodes in a reliable fashion. Failure to agree on such information correctly (e.g., nodes using inconsistent TDMA schedules) may disrupt network connectivity or substantially degrade performance.

When sharing information using an unreliable medium (such as wireless), no delivery guarantee can be given on the messages that are sent. Akkoyunlu et al. [22] have shown that, in an arbitrary distributed facility, it is impossible to provide the so called *complete status*, i.e., one cannot guarantee that two distributed parties know the ultimate fate of a transaction and whether they are in agreement with each other.

The problem is further exacerbated in the presence of external interference: the low-power transmissions of wireless sensor networks are highly vulnerable to interference caused by radio signals generated by devices operating in the same frequency range. Several studies have highlighted the increasing congestion of the unregulated ISM bands used by wireless sensor

networks to communicate, especially the 2.4 GHz band [70]. Sensornets operating on such frequencies must cope with simultaneous communications of WLAN and Bluetooth devices, as well as with the electromagnetic noise generated by domestic appliances such as microwave ovens, video-capture devices, or baby monitors.

As a result, wireless sensor nodes often communicate through interfered channels that have low chances of successfully delivering a packet. Hence, it is important to derive reliable techniques to ensure agreement even in the presence of interference, and make sure that they are efficient enough to meet the limited computational capabilities and energy resources of sensor nodes.

In this work, we design, implement, and evaluate JAG, a simple yet efficient agreement protocol for wireless sensor networks exposed to external interference. JAG introduces a jamming sequence as the last step of a packet handshake between two nodes to inform about the correct reception of a message carrying the information to be agreed upon. The key insight behind this approach is that detecting a jamming sequence in the presence of external interference is more reliable than using acknowledgement (ACK) packets to verify whether the information was successfully shared.

In environments that experience high levels of external interference, the probability of successfully transmitting a sequence of packets and completing an handshake is small, even when using short ACK packets. Despite the minimal amount of information they carry, acknowledgements are embedded into IEEE 802.15.4 frames, and hence can be destroyed if any of the bits in the header, payload, or footer is corrupted by interference. Performance can be improved by means of redundancy (i.e., by sending multiple ACK packets), but this results in a significantly higher energy expenditure and latency, which is undesirable when using resource-constrained wireless sensor nodes.

Using JAG, instead, one can minimize the energy expenditure and provide agreement guarantees under weaker and more realistic assumptions about the underlying interference pattern compared to message-based approaches. By appropriately tuning the length of the jamming sequence, one can parametrize JAG to obtain predictable performance and to guarantee agreement in a finite amount of time, even in the presence of external interference: a perfect fit for applications with timeliness requirements. We focus on the unicast case (agreement between two neighbouring nodes) and show that JAG outperforms traditional packet-based agreement protocols in the presence of interference with respect to agreement probability, energy consumption, and time-to-completion.

JAG is intended as a building block to construct protocols at different layers of the protocol stack. It could be embedded into a MAC protocol to agree on time slots or frequency channels as discussed in Sect. 3.2.6, at the transport level to agree on connection establishment or teardown, or at the application level to agree on handover of a leader role.

Our description of JAG proceeds as follows. First we define the agreement problem in wireless sensor networks challenged by external radio interference. Then we convey our main idea: using jamming as a binary signal for acknowledging the reception of packets. Thereafter, in Sect. 3.2.4, we illustrate JAG, a protocol for reliable agreement under external radio interference. We experimentally evaluate the performance of JAG under interference in Sect. 3.2.5, and after discussing the integration of JAG into existing sensornet MAC protocols in Sect. 3.2.6, we review related work in Sect. 3.2.7.

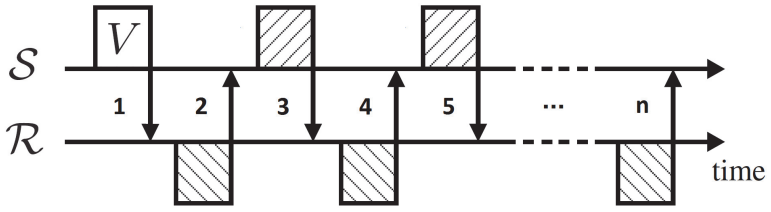


Figure 3.5: n-way handshake between nodes $\mathcal{S}$ and $\mathcal{R}$.

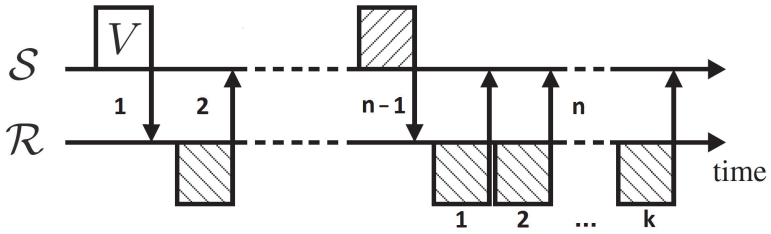


Figure 3.6: Enhanced n-way handshake between nodes $\mathcal{S}$ and $\mathcal{R}$ using redundancy: the last ACK message is transmitted k times.

3.2.2 Problem Description

Agreeing on a given piece of information is a classical coordination problem in distributed computing. The *Two Generals' Agreement Problem*, formulated by Jim Gray to illustrate the two-phase commit protocol in distributed database systems [25], is often used to explain the challenges when attempting to coordinate an action by communicating over a faulty channel, and can be described as follows.

Two battalions are encamped near a city, ready to launch the final attack. Because of the redoubtable fortifications, the attack must be carried out by both battallions at the same time in order to succeed. Hence, the generals of the two armies need to agree on the time of the attack, and their only way to communicate is to send messengers through the valley. The latter is occupied by the city's defenders, and a messenger can be captured and its message lost, i.e., the communication channel is unreliable. Since each general must be aware that the other general has agreed on the attack plan, messengers are used also to exchange acknowledgements. However, because the acknowledgement of a message receipt can be lost as easily as the original message, a potentially infinite series of messages is required to reach an agreement[1].

Agreement in Wireless Networks

In the context of wireless communications, the problem can be rephrased as follows. When two nodes, $\mathcal{S}$ and $\mathcal{R}$, need to agree on a common value V, they exchange a sequence of n messages in an alternating manner (Fig. 3.5). Node $\mathcal{S}$ is the initiator of the exchange. After the transmission

[1] A different problem that we are not addressing in this work is how to guarantee the identity of the sender of the message, as well as how to cope with misbehaving parties.

of V, each subsequent message acknowledges the receipt of the previous message, i.e., a node sends message $i > 1$ only if it correctly received message $i-1$. Each node uses a simple rule to determine the success of the exchange: if all expected messages are received, the exchange is deemed successful, otherwise the exchange is deemed unsuccessful.

The scenario described above corresponds to an *n-way handshake* between nodes $\mathcal{S}$ and $\mathcal{R}$, where n is the number of packets exchanged. The n-way handshake is a widely used mechanism in communication networks. For example, TCP employs a 3-way handshake ($n=3$) to establish connections over the network, whereas IEEE 802.11i (WPA2) uses a 4-way handshake ($n=4$) to carry out the key exchange.

An n-way handshake can have three possible outcomes:

1. **Positive Agreement.** The n messages are all received correctly, and both nodes deem the exchange as successful, accepting V.

2. **Negative Agreement.** A message m with $m < n$, i.e., a message prior to the last message n, is lost. None of the nodes receives all the expected messages, hence both nodes deem the exchange as unsuccessful, discarding V.

3. **Disagreement.** The last message n is lost. One of the two nodes receives all the expected messages, deems the exchange as successful and accepts V; whereas the second node misses the last message and therefore deems the exchange as unsuccessful, rejecting V.

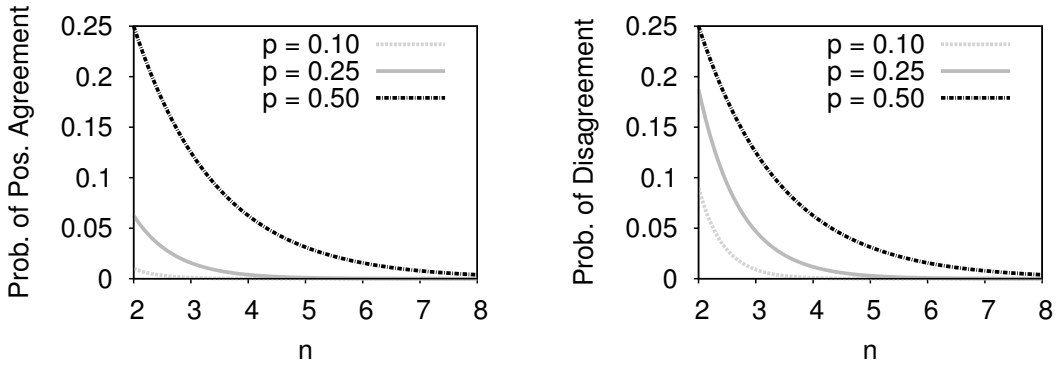
In the original two generals' scenario, a *positive agreement* would lead to a simultaneous attack of the city by both battalions and a consequent victory, a *negative agreement* would cause both battalions to stall, while a *disagreement* would trigger the attack of only one battalion and a consequent defeat of the attacking forces.

While *disagreements* are potentially fatal, *negative agreements* are often less severe. For example, if the shared value contains the next channel to be used for communication, two nodes are better off staying in the same lossy channel, rather than having only one of them move to a different frequency. The probability of negative agreements should, however, be minimized, as it may lead to reduced performance. Hence, an agreement protocol should strive to minimize disagreement as a first priority, maximize positive agreements as a second (almost equally high) priority, and minimize negative agreements as a third (substantially lower) priority. A metric to measure the quality of an agreement protocol (whose value should be minimized) is therefore the *DPA ratio* of the probability of disagreements over the probability of positive agreements.

The importance of the last message

It is important to emphasize that, in an n-way handshake, disagreements only occur if the last message is lost. Hence, depending on the application, it may be desirable to devote extra-resources to increase the successful delivery of the last packet by means of redundant packet transmissions (i.e., repeating a message several times and assuming successful transmission if at least one copy is received).

A possibility is to employ a n-way handshake in which the last packet is repeated k times, as shown in Fig. 3.6. Using this approach, the final outcome of the handshake is strongly dependent on the link quality, on the length n of the n-way handshake, and on the redundancy



(a) Positive Agreement

(b) Disagreement

Figure 3.7: Distribution of the probabilities of positive agreement and disagreement of the n-way handshake shown in Fig. 3.5 as a function of the probability of successful packet transmission p and length of the handshake n.

factor k. Letting p represent the probability that a generic message is successfully received (assuming that p remains constant over time and that it is independent for each packet), and $q = 1-(1-p)^k$ the probability of successfully receiving at least one of the k redundant packets, we obtain:

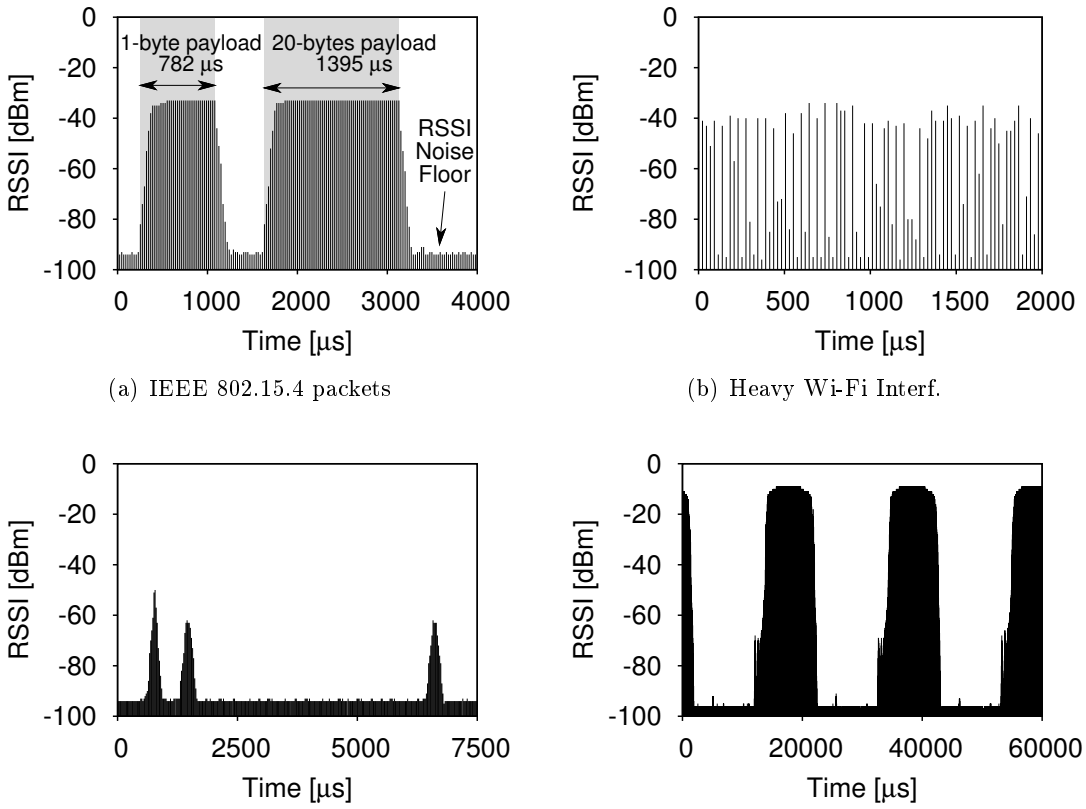
$$\begin{aligned} \text{Prob}(PositiveAgreement) &= p^{n-1}q \\ \text{Prob}(NegativeAgreement) &= 1-p^{n-1} \\ \text{Prob}(Disagreement) &= p^{n-1}(1-q) \end{aligned}$$

These equations show that in order to maximize the frequency of positive agreements and, at the same time, minimize the frequency of disagreements, we need to maximize the link quality p and maximize the level of redundancy k. The choice of a suitable n becomes a catch-22 dilemma in the presence of unreliable links, as illustrated in Fig. 3.7: long n-way handshakes minimize the probability of disagreement, but also the probability of positive agreement, whereas short n-way handshakes maximize the probability of positive agreement, but also the chances of disagreement.

Agreement in Sensor Networks Challenged by External Interference

In the context of wireless sensor networks, minimizing the amount of exchanged packets is mandatory because of the limited energy resources available, i.e., sensor nodes need to minimize the time during which the radio is active as much as possible. Therefore, the use of redundant packet transmissions and long handshakes is not advisable, as it would increase the energy consumption.

Another aspect is the channel quality affecting p. Wireless sensor nodes operate in the unlicensed ISM radio bands, and often use a very low transmission power, which makes them vulnerable to external interference. Any wireless appliance operating in the same frequency range



(a) IEEE 802.15.4 packets

(b) Heavy Wi-Fi Interf.

(c) Bluetooth interf.

(d) Microwave oven interf.

Figure 3.8: RSSI values measured using off-the-shelf wireless sensor nodes operating in the 2.4 GHz ISM band. Please notice the different scale of the x-axis.

of sensornets can potentially interfere with their communications and decrease the probability of a successful packet exchange p. In the 2.4 GHz ISM band, for example, Wi-Fi and Bluetooth networks, as well as domestic appliances such as microwave ovens, can create noise levels that overwhelm the interference resistance capabilities of DSSS radios and radically decrease the packet reception rate [50, 70]. Hence, we need to investigate ways to encode transmissions such that their success probability p is maximized despite interfered channels.

Analysis of Common Interference Sources

In order to understand the impact of external interference on the probability of successful transmission p in wireless sensor networks communications, we study the interference patterns produced by common devices operating in the 2.4 GHz ISM band. Using Sentilla Tmote Sky nodes employing a CC2420 radio, we perform a high-speed sampling of the RSSI register ($\approx$ 50 kHz as in [6]). We call this operation *fast RSSI sampling* over a time window t_{samp}. Fig. 3.8 shows the outcome of fast RSSI sampling in the presence of sensornet communications

and external interference.

Absence of external interference. When neither interference nor IEEE 802.15.4 communications are present, the fast RSSI sampling returns the so called RSSI noise floor. The latter has typically values in the proximity of the radio sensitivity threshold (e.g., in the range $[-100, -94]$ dBm for the CC2420 radio). In the presence of IEEE 802.15.4 communications, the fast RSSI sampling returns a stable value corresponding to the strength and the length of the transmitted packet (Fig. 3.8(a)). As packets have a constrained maximum payload size of 127 bytes according to the 802.15.4 PHY standard, a packet transmission at 250 Kbit/sec would not last more than 4.3 ms.

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

Wi-Fi transmissions are typically much stronger than sensornet transmissions, and can affect several IEEE 802.15.4 channels at the same time. Hauer et al. [27, 28] have shown that with a sufficiently high sampling rate, one can identify the short instants in which the radio medium is idle due to the Inter-Frame Spaces (IFS) between 802.11 b/g packets. Fig. 3.8(b) shows the outcome of fast RSSI sampling in the presence of heavy Wi-Fi interference (caused by a file transfer): it is indeed possible to identify RSSI values matching the radio sensitivity threshold between consecutive Wi-Fi transmissions.

Fig. 3.8(c) shows an example of interference generated by Bluetooth. The latter uses an Adaptive Frequency Hopping mechanism to combat interference, and hops among 1-MHz channels around 1600 times/sec., hence it remains in a channel for at most 625 μs. Since Bluetooth channels are more narrow than the ones defined by the 802.15.4 standard, it may happen that communication in multiple adjacent Bluetooth channels affects a single 802.15.4 channel.

Fig. 3.8(d) shows an example of the interference pattern caused by microwave ovens: high-power noise ($\approx$ 60 dBm) is emitted in the 2.4 GHz frequency band in a very periodic fashion. The period mostly depends on the power grid frequency, but can also slightly vary depending on the oven model. Works in the literature report a power cycle of roughly 20 ms (at 50 Hz) or 16 ms (at 60 Hz) with an active period of at most 50% of the power cycle [6, 34].

The Role of Idle Periods

In the presence of external interference, n-way handshakes need to take advantage of idle periods. In principle, the longer the idle period and the shorter the handshake, the higher the likelihood of obtaining positive agreements. However, the interplay between idle periods and n-way handshakes is complex because of the particular patterns of each interfering source. Some devices, such as microwave ovens, generate periodic interference patterns with relatively long idle periods (Fig. 3.8(d)), while others, such as Wi-Fi stations, generate interference patterns with short idle periods of a highly variable length (Fig. 3.8(b)).

Having short idle periods reduces the probability of successfully completing a handshake, and this is especially critical in the presence of heavy Wi-Fi interference. Fig. 3.9 shows the cumulative distribution function (CDF) of idle and busy periods measured by a Maxfor MTM-

CM5000MSP node in the presence of a laptop continuously downloading a file from a nearby access point. A channel is defined as busy if the RSSI is higher or equal than a configurable threshold R_{thr} and idle otherwise. In such a scenario, the probability of having an idle period longer than 2 ms is smaller than 5%. Therefore, there is only a little chance that a message-based handshake successfully completes within an idle period. In order to escape interference, one would need to use short messages and send them as close as possible to each other, in order to increase the chances of fitting into an idle period.

Off-the-shelf IEEE 802.15.4-compliant radios such as the CC2420 offer the ability to automatically generate and send ACKs for data frames in hardware. The advantage of hardware acknowledgements is a significant reduction of latency compared to solutions in which the ACK is generated via software [53]. However, hardware ACKs cannot be used to carry out a complete n-way handshake (with $n > 2$), since they cannot be used in reply to another hardware ACK. Imagine a node $\mathcal{S}$ starting a handshake by sending a message to $\mathcal{R}$. The latter can reply with a hardware ACK, but $\mathcal{S}$ will have to receive and extract the packet, analyse its validity, as well as to prepare a new ACK frame, load it into the buffer, and send it over-the-air[2]. This may cause long latencies that break the agreement in the presence of short idle periods.

Furthermore, it is also highly inefficient to encode the binary information carried by an ACK message inside an IEEE 802.15.4 frame, especially in the presence of interference. Despite the payload contains only a single ACK bit, the whole packet consists of synchronization preamble and a physical header (4-bytes preamble, 1-byte Start of Frame Delimiter (SFD), 1-byte length field), as well as a MAC header and footer (2-bytes frame control, 1-byte sequence number, 4-20-bytes address, 2-bytes Frame Check Sequence (FCS)). If any of the bits in the headers and preamble is corrupted by interference, the packet may become undecodable [32, 43].

Therefore, instead of encoding the last ACK as packet transmission, we propose to encode it by means of **jamming**, where the presence of a jamming sequence signals the receipt of the previous message. The key advantage of this approach is that jamming, as generated by off-the-shelf wireless sensor nodes, can be reliably detected even under interference.

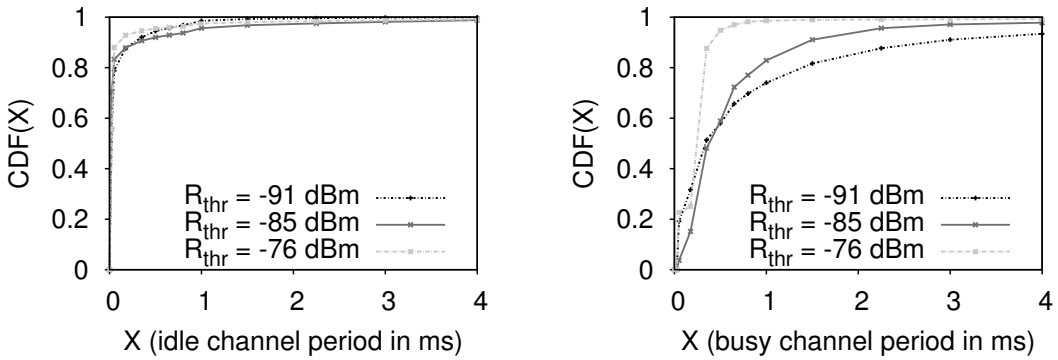
3.2.3 Jamming as Binary ACK Signal

We propose to encode the last acknowledgement of a n-way handshake by means of **jamming** (i.e., transmission of a carrier signal), where the presence of a jamming sequence signals the receipt of the previous message. The key advantage of this approach is that precisely timed jamming signals can be generated using off-the-shelf wireless sensor nodes and can be reliably detected even under heavy interference.

Generating a Jamming Sequence

In a recent study, we showed that off-the-shelf radios can be used to generate controllable and repeatable jamming signals in specific IEEE 802.15.4 channels by transmitting a modulated or unmodulated carrier signal that is stable over time [6, 8]. This approach is superior to packet-based jamming, as the generated signal is independent of both packet sizes and inter-packet times. We hence generate precisely timed jamming signals by configuring the MDMCTRL1

[2] In case a train of k redundant software ACKs is sent, the packet can be loaded into the buffer once and sent repeatedly.



(a) Idle periods (b) Busy periods

Figure 3.9: Cumulative distribution function (CDF) of idle and busy periods measured by a Maxfor MTM-CM5000MSP node in the presence of a laptop continuously downloading a file from a nearby access point.

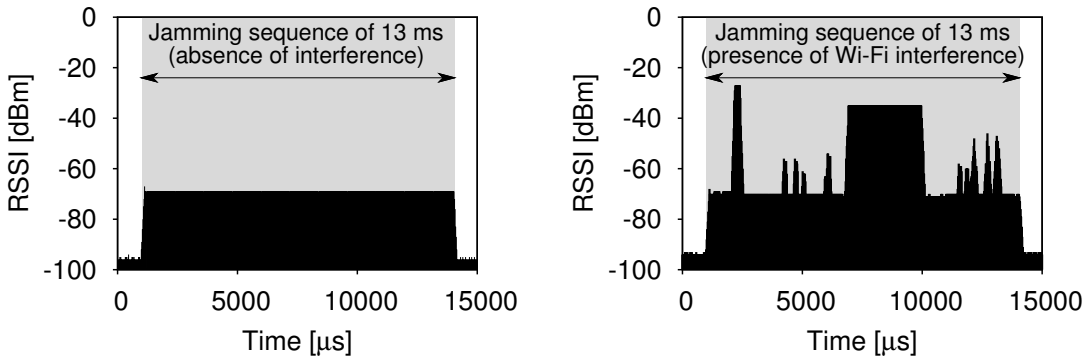
register, so that the CC2420 radio outputs a continuous modulated carrier signal. The detection of the latter is based on high-frequency RSSI sampling, as discussed next.

Detecting a Jamming Sequence

Common radio chips offer the possibility to read the RSSI in absence of packet transmissions. Several researchers have shown that it is a useful way to assess the noise and the level of interference in the environment [27, 50, 52]. RSSI readings close to the sensitivity threshold of the radio indicate absence of interference, whereas values above this threshold identify a packet transmission, or a busy/congested medium (see Fig. 3.8).

Hence, we use the fast RSSI sampling mechanism mentioned in Sect. 3.2.2 to detect the presence or absence of a jamming signal generated by a sensor node. A jamming sequence generated using the method described in Sect. 3.2.3 results in a stable RSSI value above the sensitivity threshold of the radio, as shown in Fig. 3.10(a). Therefore, one can detect if a jamming signal was transmitted by making sure that no RSSI sample falls down to the sensitivity threshold of the radio.

In the presence of additional external interference, the RSSI register will return the maximum of the jamming signal and the interference signal due to the co-channel rejection properties of the radio [6]. Fig. 3.10(b) illustrates this for a jamming signal sent in the presence of Wi-Fi interference. As we have shown in Sect. 3.2.2, typical interference sources – in contrast to our jamming signal – do not produce continuous interference for long periods of time, rather they alternate between short idle and busy periods. That is, *if the jamming signal lasts longer than the longest busy period of the interference signal, we are unequivocally able to detect the absence of the jamming signal* by checking if any of the RSSI samples equals the sensitivity threshold of the radio. We exploit this property to design JAG, a protocol for reliable agreement under external interference.



(a) Absence of Interference

(b) Presence of Wi-Fi Interference

Figure 3.10: RSSI values measured by a Maxfor MTM-CM5000MSP node during the transmission of a jamming sequence in absence of interference (a), and in the presence of external Wi-Fi interference (b).

Identification of the Interfering Source

While a jamming signal can encode the binary acknowledgement information, it cannot encode the identities of sender and receiver as a regular packet would. When carrying out a handshake, however, these identities are already included in the message V to be acknowledged, and therefore are implicitly known to the two nodes, as long as the communication channel remains allocated exclusively for the whole duration of an exchange. In this way, intra-network interference is avoided, and a jamming sequence acknowledging the reception of V can be identified reliably by means of an RSSI threshold, as we discuss in Sect. 3.2.4. Any protocol that embeds JAG as a building block for agreement needs to meet this requirement. At the MAC layer, RTS/CTS can be used to allocate the channel in CSMA protocols, whereas in TDMA protocols the timeslots must be long enough to complete an exchange.

3.2.4 JAG: Reliable Agreement under Interference

We call *JAG (Jamming-based AGreement)* the three-way handshake in which the last ACK is sent in the form of a jamming signal as shown in Fig. 3.11. The choice of three-way handshakes (as opposed to two-way) is motivated by two facts. First, a three-way handshake increases the reliability of identifying the jamming signal because it provides a reference RSSI value (this will be explained in more detail in Sect. 3.2.4). Second, three-way handshakes avoid disagreements due to asymmetric links: for instance, if $\mathcal{S}$ has a link with $\mathcal{R}$ but the reverse link is not present, a two-way handshake would always lead to disagreements, since $\mathcal{R}$ is not able to confirm the reception of V.

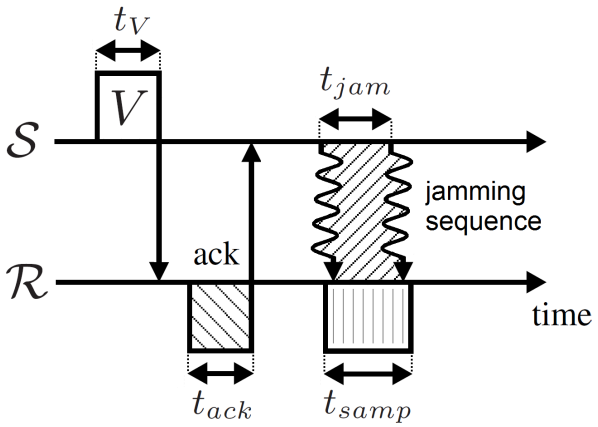


Figure 3.11: Illustration of JAG: the last acknowledgement of the 3-way handshake between nodes $\mathcal{S}$ and $\mathcal{R}$ is sent in the form of a jamming signal.

Protocol Design

The protocol proceeds as follows. $\mathcal{S}$ initiates the exchange and sends the information V towards a receiver $\mathcal{R}$. If V is successfully received, $\mathcal{R}$ saves the signal strength r_s of the received packet and sends an ACK message back to $\mathcal{S}$. We can send either hardware or software acknowledgements: in the remainderwe assume that hardware ACKs are available. If $\mathcal{S}$ receives the acknowledgement, it transmits a jamming signal for a period t_{jam}. Meanwhile, $\mathcal{R}$ carries out a fast RSSI sampling for a period $t_{samp} \leq t_{jam}$ that is synchronized in such a way that the fast RSSI sampling is carried out while the jamming signal is on the air. The message V is used as the synchronization signal: given that clock drift is not too high at timescales of a few milliseconds, it is sufficient to include a short safety margin to compensate for drift (more details in Sect. 3.2.4). For simplicity, we assume $t_{jam} = t_{samp}$.

If $\mathcal{R}$ detects the presence of the jamming signal, it deems the exchange as successful; otherwise, V is discarded. $\mathcal{S}$ deems the exchange as successful if the ACK is received within a short timeout period, otherwise the jamming sequence is not generated and the handshake immediately terminated.

After the reception of V, node $\mathcal{R}$ carries out a fast RSSI sampling as described in Sect. 3.2.3 to detect the absence or the presence of the jamming sequence transmitted by $\mathcal{S}$. The method to detect the jamming signal is simple: if a jamming sequence is sent, *all* RSSI samples should be above r_{noise}, with the latter being the RSSI noise floor threshold of the radio. Hence, if during t_{samp} we observe *at least one RSSI sample* with a value comparable to r_{noise}, we conclude that the jamming sequence was not transmitted.

This process can be described as follows. Denoting $\{x_1, x_2, \ldots, x_n\}$ as the sequence of RSSI values sampled during t_{samp}, we define the binary sequence $\{X_1, X_2, \ldots, X_n\}$ as follows: if $x_i \leq r_{noise}$, then $X_i = 1$, else $X_i = 0$. $\mathcal{R}$ makes a decision about the presence of the jamming sequence as follows: if $\sum_{i=1}^{n} X_i = 0$, then $\mathcal{S}$ was transmitting a jamming signal and hence V is accepted; otherwise, V is discarded.

Using this algorithm, JAG would operate correctly and would be able to recognize the pres-

ence or absence of a jamming signal reliably. However, we can enhance its performance significantly by exploiting the knowledge of the received signal strength r_s of the packet containing V.

The Role of r_s

Under the hypothesis that the jamming signal has a reasonably similar signal strength to r_s (RSSI does not change significantly between consecutive transmissions spaced by only a few milliseconds), $\mathcal{R}$ can filter out any interference source weaker (i.e., resulting in an RSSI range smaller) than $(r_s - \Delta_r)$, with Δ_r being a tolerance margin to compensate for the inaccuracy of low-power radios and the instability of the RSSI readings. This allows to shorten t_{jam} and achieve a higher energy-efficiency: as we can see in Fig. 3.9(b), the higher R_{thr}, the shorter the duration of busy periods.

Hence, if $(r_s - \Delta_r) > r_{noise}$, JAG's algorithm is executed as follows: if $x_i < (r_s - \Delta_r)$, then $X_i = 1$, else $X_i = 0$. $\mathcal{R}$ still makes a decision about the presence of the jamming sequence in the following way: if $\sum_{i=1}^{n} X_i = 0$, then $\mathcal{S}$ was jamming and hence V is accepted; otherwise, V is discarded.

Furthermore, r_s also increases the reliability of fast RSSI sampling. The maximum distance over which a packet can be successfully received and decoded is shorter than the distance over which a jamming signal can be captured. This may lead to confusion in a scenario in which two nodes that cannot communicate with each other are allocated the same time slot in a TDMA protocol and transmit a message concurrently. By using a threshold r_s, we make sure that a receiver $\mathcal{R}$ is in the communication range of $\mathcal{S}$, and therefore r_s cannot be achieved by any other node transmitting simultaneously.

The Role of t_{jam}

The length of the jamming sequence t_{jam} can be tuned in order to provide probabilistic guarantees on the fraction of disagreements. Denoting t_{busy}^{max} as the maximum busy period that can be encountered in the presence of interference, we can guarantee that $\mathcal{S}$ and $\mathcal{R}$ will agree on V by setting $t_{jam} > t_{busy}^{max}$. In such a case, an idle period will surely be encountered during t_{samp}, and the absence of a jamming sequence unequivocally detected, as discussed in Sect. 3.2.3. Hence, the most pernicious outcomes (disagreements) are eliminated, and only positive or negative agreements can occur.

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

JAG Implementation

We implement JAG on Maxfor MTM-CM5000MSP and Sentilla Tmote Sky nodes. Our implementation, based on Contiki [20], uses two main building blocks: the generation of a jamming sequence and the high-frequency RSSI sampling. The former uses the CC2420 transmit test modes as described in Sect. 3.2.3. The latter is implemented as in our previous work [6], so

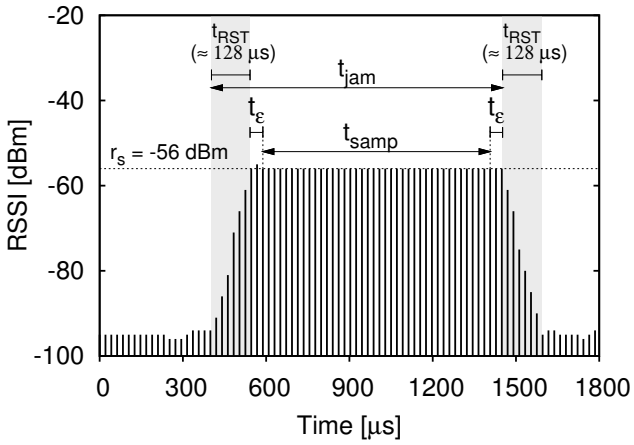
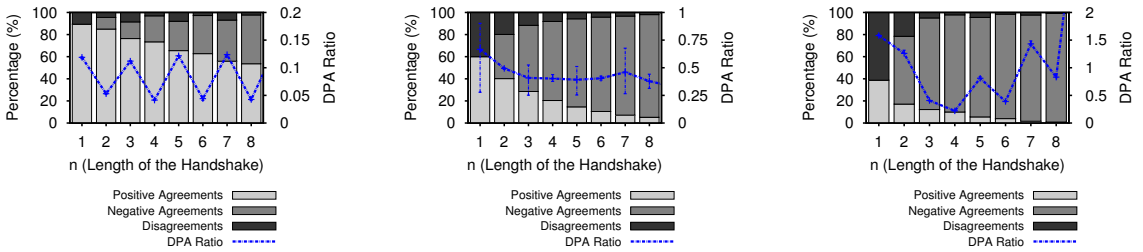


Figure 3.12: Alignment between t_{samp} and t_{jam}: RSSI readings obtained during t_{RST} and t_ϵ are discarded to compensate for synchronization inaccuracies.

that we roughly obtain one RSSI sample every 20 μs. Although a sampling rate of 50 kHz does not capture the transmissions from all wireless devices operating in the same frequency band of sensor networks (e.g., IEEE 802.11n devices), it is still enough to identify most of the idle periods that occur between Wi-Fi transmissions and hence to distinguish the jamming sequence from external interference.

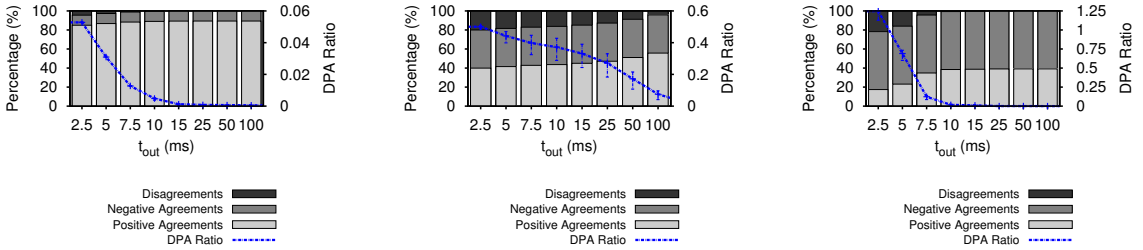
For all our experiments we use NULLMAC, a MAC layer that just forwards packets to the upper or lower protocol layer and does not perform any duty cycling, but reports the presence of hardware acknowledgements. We chose NULLMAC in order to obtain results that are independent of specific MAC features and parameters. To ensure that the execution time of the entire handshake is bounded and independent of clear channel assessment (CCA) back-off times, we do not postpone transmissions until the channel becomes clear. Instead, we carry out a single clear channel assessment before sending V: if the channel is found busy, the transmission is cancelled. This is an optimization, as sending V despite the busy channel would result in a negative agreement (V would be lost).

To ensure alignment between jamming t_{jam} and sampling t_{samp}, we implement a simple synchronization mechanism. $\mathcal{S}$ and $\mathcal{R}$ synchronize their operations based on the reception of V: the transmission or reception of the Start of Frame Delimiter (SFD) is used as the synchronization signal. Although at timescales of a few milliseconds clock drift is minimal, the beginning of t_{samp} may not be aligned with the beginning of the jamming sequence because of the time required for RSSI to settle. The RSSI of the CC2420 radio is indeed an average of the last 8 bit symbols [6] and hence one needs to wait for the RSSI to stabilize (this takes $\approx t_{RST} = 128\mu$s) before being able to measure r_s (see Fig. 3.12). Since RSSI readings are not instantaneous and their duration may slightly differ among different nodes, we introduce a safety margin t_ϵ during which the RSSI readings are discarded: this allows us to compensate for possible synchronization inaccuracies. The actual length of t_{jam} must therefore be increased by $2 \cdot (t_{RST} + t_\epsilon)$ to make sure that t_{samp} is correctly aligned.



(a) Bluetooth (b) Real Wi-Fi (c) Emulated Wi-Fi

Figure 3.13: Performance of a packet-based n-way handshake under different types of interference.



(a) Bluetooth (b) Real Wi-Fi (c) Emulated Wi-Fi

Figure 3.14: Performance of 2-MAG (2-way handshake in which the last acknowledgement packet is sent k times) under different types of interference. The longer t_{out}, the lower the amount of disagreements in favour of positive agreements, at a price of an increased energy consumption.

3.2.5 Experimental Evaluation

Experimental Setup

We carry out our experiments in two small-scale sensornet testbeds with USB-powered sensor nodes. The first testbed consists of 15 MTM-CM5000MSP nodes deployed in an office environment, whereas the second testbed uses the same type of sensor nodes deployed in a residential building. We use our first testbed to evaluate the performance of several agreement protocols under different types of interference. To this end, we use JamLab [6], a tool for controlled and realistic interference generation in specific IEEE 802.15.4 channels. We configure JamLab to emulate a continuous file transfer produced by either Bluetooth or Wi-Fi devices in specific IEEE 802.15.4 channels. We further carry out experiments in the presence of a Wi-Fi interference generated by a laptop continuously downloading a file from a nearby access point. We validate our first set of results using a second testbed deployed in residential buildings surrounded by Wi-Fi stations: we run different agreement protocols for several days and compare their performance over time.

In our experiments, we use several pairs of nodes $\mathcal{S}$ and $\mathcal{R}$. Node $\mathcal{S}$ always initiates the handshake, and transmits a data packet composed of a 6-byte payload containing the information to be agreed upon V and the transmission power used T_P. For each handshake (which is initiated after a random interval in the order of hundreds of milliseconds), we select a random transmission power between -25 dBm and 0 dBm in order to create different types of links. $\mathcal{R}$ replies to the packet using T_P, i.e., the same transmission power used by $\mathcal{S}$. Hardware ACKs are enabled by default, and nodes remain on the same channel during the whole duration of the experiment, in which we perform several hundred thousand handshakes.

Packet-based n-way handshake

We firstly analyse the performance of the packet-based n-way handshake shown in Fig. 3.5 (redundancy factor $k = 1$) under different interference patterns. In our implementation, every packet from $\mathcal{R}$ to $\mathcal{S}$ is sent using the hardware ACK support, so to minimize the latency between the reception of the previous packet and the dispatch of the following one.

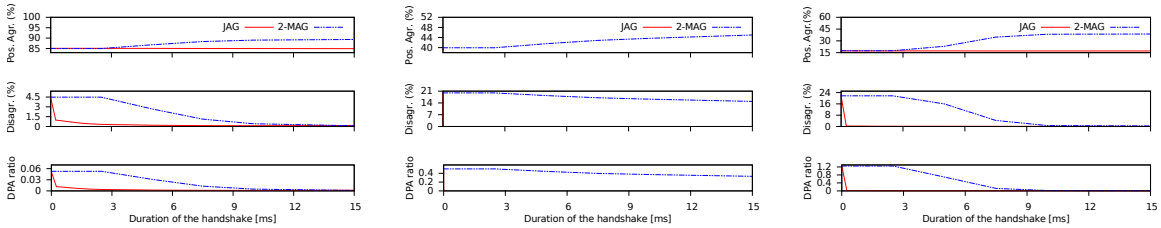
Fig. 3.13 shows the percentage of positive/negative agreements and disagreements obtained under different interference patterns. The values are computed as an average over all transmission power values T_P used in our experiments, excluding the ones leading to asymmetric links.

Fig. 3.13(a) depicts the performance of the protocol under JamLab's emulated Bluetooth file transfer. As discussed in Sect. 3.2.2, the longer the handshake, the smaller the amount of disagreements and positive agreements. Hence, the DPA ratio does not decrease when increasing the length of the handshake n. The alternating performance of the DPA ratio is caused by the interchange between software and hardware ACKs: the former require a higher latency to be transmitted, and hence offer a worse performance with respect to the latter. Fig. 3.13(b) and 3.13(c) show the performance of the n-way handshake protocol under JamLab's emulated Wi-Fi transfer and under Wi-Fi interference generated by a continuously active laptop, respectively. As the interference becomes heavier, the amount of positive agreements and the amount of disagreements drastically decrease after few iterations, hence the DPA ratio does not improve significantly. Our experiments therefore confirm our observations in Sect. 3.2.2: packet-based n-way handshakes are not optimal under external interference.

2-MAG: 2-way handshake enhanced with redundancy

To minimize the DPA ratio, we introduce redundancy of the last ACK packet as discussed in Sect. 3.2.2, and we analyse the performance of a 2-way handshake in which the last ACK packet is sent k times, as illustrated in Fig. 3.6. For simplicity,we will refer to this protocol as 2-MAG (2-way handshake Message-based AGreement).

Given the structure of JAG, a more fair comparison would involve a 3-way handshake message-based agreement protocol in which the last packet is sent k times. The choice of a 2-way handshake is driven by the results obtained in Fig. 3.13: a low n minimizes the probability of negative agreements, and therefore there are higher chances that 2-MAG sustains more positive agreements and outperforms JAG thanks to its redundant transmissions. We make sure to carry out a fair comparison by eliminating asymmetric links that would always lead to disagreements when using a two-way handshake.



(a) Bluetooth (b) Real Wi-Fi (c) Emulated Wi-Fi

Figure 3.15: Compared to the 2-way handshake in which the last acknowledgment packet is sent k times, JAG performs better independent of the interfering source, as it reduces the duration of the handshake required to minimize the amount of disagreements.

In our implementation, hardware ACKs are enabled, i.e., the first ACK packet sent from $\mathcal{R}$ to $\mathcal{S}$ has a short and fixed-delay latency. Every other ACK packet will be generated via software by pre-loading the ACK into the radio buffer and by repeatedly sending its content k times. Please note that the preparation of the software ACK is time-critical, as one need to extract and analyse V before creating the ACK and loading it into the radio buffer.

In order for $\mathcal{S}$ to consider V as successfully exchanged, it is sufficient to receive one ACK packet within a maximum waiting time t_{out}. Clearly, the longer t_{out}, the higher the likelihood that at least one ACK packet will be correctly decoded and the better 2-MAG will perform (at the price of an increased energy consumption). Hence, we compute t_{out} as the maximum time in which node $\mathcal{S}$ waits for a valid ACK packet from $\mathcal{R}$.

Fig. 3.14 shows the percentage of positive and negative agreements as well as disagreements obtained in the presence of interference using 2-MAG as a function of t_{out}. As expected, the longer t_{out}, the lower the amount of disagreements in favour of positive agreements. As this minimizes the DPA ratio, 2-MAG outperforms a generic n-way handshake without redundancy in the presence of external interference.

JAG: Jamming-based AGreement

We now evaluate the performance of JAG and compare it against 2-MAG. In particular, we are interested in comparing how the percentage of positive/negative agreements and disagreement change when we increase the duration of the handshake. Intuitively, the longer t_{out} for 2-MAG and the longer t_{jam} for JAG, the better the performance. However, it is important to see their distribution to study the protocols' energy-efficiency and their DPA ratio under interference.

Fig. 3.15 shows the results: JAG sustains a significantly lower amount of disagreements compared to 2-MAG already for small values of t_{jam}. For example, 2-MAG requires more than 7.5 ms to obtain less than 1% disagreement under Bluetooth interference, whereas JAG achieves this amount with a $t_{jam} \leq 250\mu$s.

Even though 2-MAG has a high number of positive agreements, it requires significantly higher values of t_{out} to reduce the amount of disagreements and the DPA ratio. JAG, instead, has a very low rate of disagreements under every type of interference even with small t_{jam}, which enables significant energy savings, as shown in Fig. 3.16. Furthermore, when t_{jam} is longer than

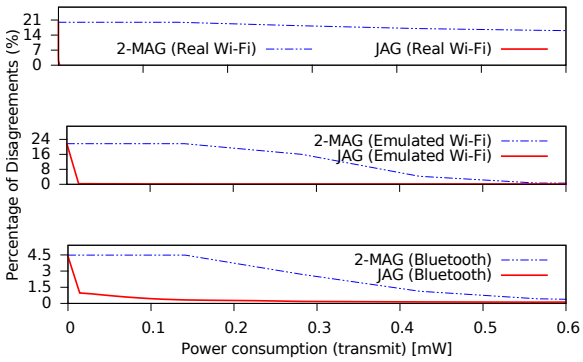


Figure 3.16: Disagreements as function of energy for JAG and 2-MAG.

the longest interference burst, we do not have any disagreements as discussed in Section 3.2.4. Obtaining this behaviour using packet-based approaches would require a significantly higher cost: Fig. 3.14(b) shows that even when sending bursts of ACKs for 100 ms, one cannot still guarantee the absence of disagreements. Hence, compared to packet-based approaches, JAG performs better and guarantees agreement with less costs and with weaker and more realistic assumptions about the underlying interference pattern.

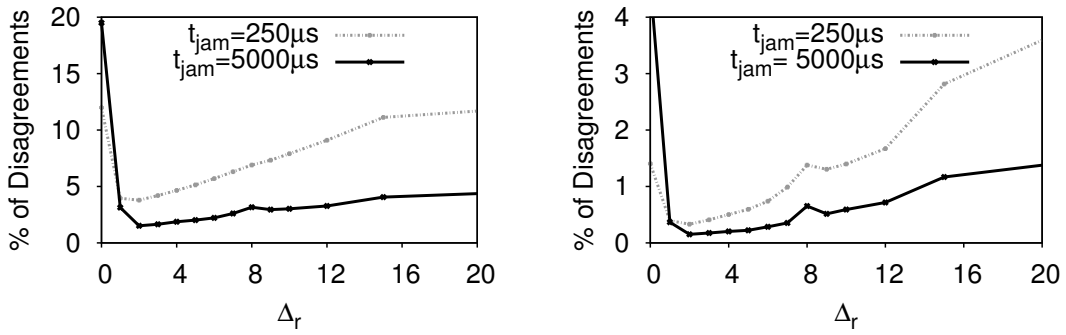
Fig. 3.15(c) shows that the rate of disagreements obtained in the presence of emulated Wi-Fi interference tends to zero faster than the one obtained in the presence of real Wi-Fi interference. This is because the interference generated by JamLab contains fast transmissions with short idle and busy periods. Therefore, JAG has high chances to detect an idle period already when using a short t_{jam}.

In addition to t_{jam}, another parameter to be configured in JAG is Δ_r, which helps in compensating changes between r_s and the strength of the received jamming signal. Δ_r should be selected not too small (so to account for the inaccuracy of the RSSI readings), but at the same time not too large, as this would neutralize the benefits of having knowledge of r_s. Fig. 3.17 depicts the percentage of disagreements as a function of Δ_r: a value of 3 dBm offers a good trade-off.

Finally, we validate the goodness of JAG by running a long-term experiment in our second testbed deployed in a residential environment. In particular, we compare the performance of JAG and 2-MAG over time when using $t_{jam} = 500\mu s$ for JAG and $t_{out} = 5ms$ for 2-MAG (Fig. 3.18). We do not change the configuration of the two protocols throughout the duration of the experiment. The interference in the environment changes significantly over the day: a lot of Wi-Fi activity was present during daytime in the weekend (May, 12-13), but it was quiet during night and on Monday (May, 14) during the day, as most people were not in their homes. Despite selecting a t_{out} 10 times higher than t_{jam}, JAG sustains a significantly lower amount of disagreements and outperforms 2-MAG during the whole duration of the experiment.

3.2.6 Integration of JAG into MAC Protocols

As previously discussed, JAG is intended as a building block to construct protocols at different layers of the protocol stack. For example, it could be embedded into a MAC protocol to agree



(a) Real Wi-Fi (b) Heavy Wi-Fi

Figure 3.17: Role of Δ_r on the probability of disagreement.

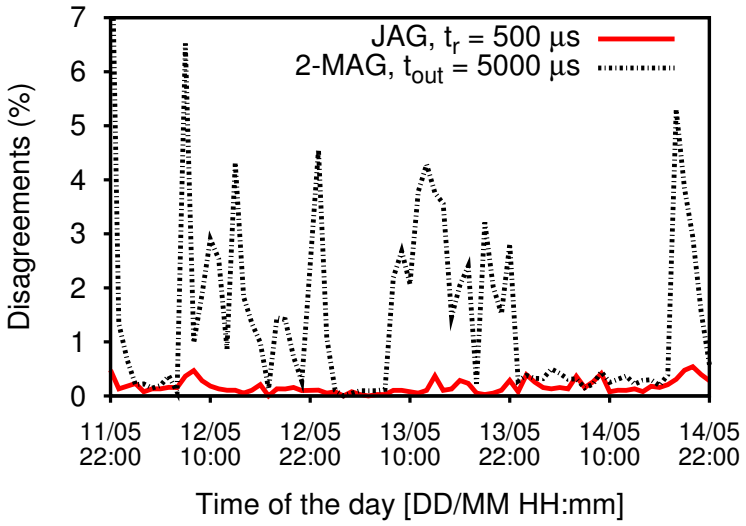


Figure 3.18: Long-term experiment in a residential environment.

on the TDMA schedule or the next frequency channel. We now discuss how JAG can be integrated in existing MAC protocols to enhance their performance.

As many deployments gather environmental data and send them to a number of sinks, several convergecast MAC protocols have been proposed in sensor networks, such as Chrysso [30] and CoReDac [59]. In these protocols, nodes are logically organized into parent-children groups that may operate on different channels. In Chrysso [30], individual parent-children pairs collaboratively switch their communication channel as soon as performance degrades. In particular, a parent node monitors the average back-off time, and as soon as it exceeds a given threshold, it instructs all its children to carry out a channel switch by piggybacking the "switch-channel

command" onto ACK messages, and then switches to the next channel. This operation is carried out for each parent-child pair individually, and can be considered a two-way handshake between child and parent (2-MAG) in which the information V to be agreed upon is contained in the second message. Please note that, on a high-level basis, V does not have to be necessarily included in the first message of the exchange: in a n-way handshake, V is in any case only used once the last message has been received, so it can be embedded in any of the messages exchanged in the handshake. The only difference with respect to 2-MAG is that, when piggybacking an information V into an ACK message, the latter cannot be sent as a hardware ACK as it contains extra-information.

JAG can be embedded into Chrysso by replacing the 2-way handshake between child and parent with a 3-way handshake in which the child sends an initial packet P, the parent answers with a software ACK containing the new channel to be used (V), and the child confirms the reception of V by jamming for a predefined amount of time t_{jam}. The parent node deems the exchange as successful (jamming sequence detected) or unsuccessful (jamming sequence not detected) depending on the results of a fast RSSI sampling, as described in Sect. 3.2.4.

The same principle can be used to enhance the performance of CoReDac [59], a TDMA-based convergecast protocol in which parent nodes split their reception slots into subslots, and assign one slot to each child in order to build a collection tree that guarantees collision-free radio traffic. As in Chrysso, also in CoReDac the assignment information used for synchronizing the TDMA-schedules is piggybacked onto ACK messages, and one can introduce a three-way handshake using JAG in the same way as described above. However, in the current version of CoReDac, there is a single aggregated ACK message containing the identifier of all children: this can be easily changed to individual ACKs to each child without affecting the overall protocol architecture.

The use of a 3-way handshake requires additional energy compared to the traditional message-based 2-way handshake implemented by Chrysso and CoReDac. However, this may pay off in the presence of interference, as it would increase the chances of agreement. As we have shown in our previous work [10], CoReDac performs poorly in the presence of interference, since when an ACK is lost, a sensor node needs to keep its radio on until it hears a new one, and integrating JAG may lead to substantial performance improvements.

3.2.7 Related Work

Agreement is a well-known problem in distributed systems. Pioneering work in the late 1970s highlighted the design challenges when attempting to coordinate an action by communicating over a faulty channel [22, 25].

In the context of wireless sensor networks, the agreement problem has not been widely addressed. The main focus has been on security for the exchange of cryptographic keys [18], and on average consensus for nodes to agree on a common global value after some iterations [66]. Similarly to these studies, our work aims at protocols that allow a set of nodes to agree on a piece of information. In addition, we also tackle agreement under interference and provide a lightweight energy-efficient solution that fits applications with strict performance requirements.

Our work is motivated by studies reporting the degrading QoS caused by the overcrowding of the RF spectrum in unlicensed bands [70]. Several solutions have been proposed: Chowdhury and Akyildiz identify the type of interferer and schedule transmissions accordingly [33]. Liang

et al. increase the resilience of packets challenged by Wi-Fi interference using multi-headers and FEC techniques [43]. Other protocols, such as Chrysso and ARCH, dynamically switch the communication frequency as soon as interference is detected [30, 56]. As these protocols rely on packet exchanges to coordinate the channel switching, one can use JAG to improve their performance, as discussed in Sect. 3.2.6.

Another set of studies propose to cope with interference by exploiting its idle or busy periods. Noda et al. have proposed a channel quality metric based on the availability of the channel over time, which quantifies spectrum usage [16]. Hauer et al. report the interference observed by a mobile body area network in public spaces, and the study shows the intermittent interference caused by Wi-Fi AP in all IEEE 802.15.4 channels [28]. Similarly, Huang et al. have shown that Wi-Fi traffic inherently leaves "a significant amount of white spaces" between 802.11 frames [23]. BurstProbe uses a probing mechanism to periodically measure burst error patterns of all links used in the deployment and, whenever the interference patterns leave predicted bounds, a warning is issued so that one can reconfigure the deployed network [31]. Similarly to these studies, JAG exploits idle times for data packets, but also leverages the bursty nature of interfering sources to achieve reliable agreements through the use of jamming signals.

3.3 Evergreen

A typical wireless sensor network (WSN) has hundreds of battery-powered remote wireless motes and a few sinks, called root nodes. Compared to a normal mote, a root node is usually more resourceful, easier to access, and mains powered. Humans interact with a WSN using a root node, which automatically collects the data of the whole network, freeing them from the tedious task of accessing each and every mote individually. To assist the root node in collecting all the data, sensor motes run a *collection protocol* that is responsible of sending the data from the source node(s) to the (closest) root node.

Existing collection protocols are typically created to work well under a set of specific assumptions. Consequently, the performance of these protocols deteriorates significantly when used in (hostile) 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 favorable environments achieving at least the same performance as existing protocols designed for those conditions.

The next subsection outlines related work in the context of Evergreen. Subsequent subsections provide a brief overview of the contributions made by Evergreen. To put Evergreen into perspective, the penultimate subsection compares its performance with that of two widely used existing collection protocols namely, CTP and ORW [24][39]. Finally, the last subsection draws conclusions.

3.3.1 Related Work

The Collection Tree Protocol (CTP) is a well-known and one of the most widely used collection protocols [24]. It was the first protocol to use adaptive beaconing (based on the Trickle timer) instead of sending period control messages to reduce protocol overhead to the bare minimum [24][69]. CTP also uses data-path validation to avoid routing loops, and is based on an accurate link quality estimation, adding to its efficiency. Overall CTP works very well if the network is stable, and is shown to deliver on average more than 90% data packets while using only a 3 % duty cycle in those conditions [24]. However, CTP's performance deteriorates when the nodes are mobile or when the network links are volatile with rapidly changing link qualities [47]. This is because CTP's link estimation might not quickly detect changes on some links, and its forwarding algorithm can then take up to a few seconds to converge to the new routes.

The Backpressure Collection Protocol (BCP) improves on CTP [47]. A node running BCP forwards packets to that neighbor whose forwarding queue has the smallest number of packets among all of the neighbors. This makes traffic automatically flow towards root nodes where data is drained from the network; nodes surrounding a root node will have smaller queue sizes compared to nodes farther away. Thus, BCP provides a forwarding algorithm (based on queue sizes) that converges faster and performs better than CTP when the network is *not* static. However, BCP runs at a higher duty cycle than CTP when the network *is* static. Furthermore, packets experience higher end-to-end delay and lower delivery rates in the static network as compared to CTP. Thus, CTP works better if the network is static whereas BCP outperforms CTP in dynamic situations.

Recently the Opportunistic Routing Protocol for WSN (ORW) was proposed [39]. In ORW, instead of sending a data packet to a specific neighbor, a node n sends packets to its immediate neighbors, with the route cost from node n inside that data packet. A neighboring node m receiving the data packet, checks if it has a routing cost greater than what is mentioned in the data packet, in which case the data packet is silently dropped. Otherwise, node m accepts the packet and sends back an acknowledgment packet to node n. Subsequently, node m unicasts the packet to its own neighbors and the packet continuing the flow through the network. The advantage of ORW is that end-to-end delay is reduced significantly because node n, instead of waiting for a specific node to wake up, starts sending the packet immediately and the first node that wakes up from sleep replies back with an acknowledgment. Thus at each hop several milliseconds are saved accumulating into a significant amount of time over the whole length of the data path. Another advantage of this forwarding scheme is that data packets flow through multiple paths instead of majority of packets taking the same path to the root node. This increases network lifetime by distributing the forwarding task among different nodes and also avoids congestion in the network. However, there are several shortcomings of ORW, due to which it cannot be used in all scenarios. First, the MAC layer needs to be changed to support ORW-style forwarding, and currently only the `CC2420` radio is supported. Second, ORW has the potential of producing a large number of duplicate packets. When a node (say m_1) sends back an acknowledgement, other nodes in the vicinity know they should not forward the packet too. However, if a node m_2 cannot overhear the acknowledgement by (e.g., because it is out of range) it will assume it will need to forward the packet; m_2 will then send an ACK, which is not received by m_1 due to channel symmetry, ending up with both nodes forwarding the packet,

hence, introducing duplicates. When following disjoint paths towards the root node, no scheme can be used to filter out such duplicates on intermediate nodes. Duplicate packets can create congestion and drain the motes' energy. The likelihood of having multiple nodes receiving the same packet increases when nodes are awake often and when the data rate is high. Therefore, ORW is best suited when motes have a relatively long sleep cycle and packets are generated at a slow pace.

The Case For Evergreen: We have discussed three prominent collection protocols for WSN. However, each works in a specific scenario and its performance deteriorates if used otherwise. In summary, CTP does not perform well if used in a non-static network. BCP performs not as good as CTP when the network has static settings. ORW works well only if packets are generated at a low pace and nodes have reasonably long sleep cycles. Furthermore, ORW is MAC and hardware dependent. The goal of Evergreen is to perform well in all the above mentioned scenarios (the reason it is named ***Ever****green*!), that is

- When the network is static or non-static. We refer to a non-static network, when it has mobile nodes, encounters high interference on its link, or experiences significant temperature variations.
- When data rates are very high or very low. At low data rates Evergreen should match (or reduce on) the duty cycle of CTP, and when data rates are high, Evergreen should continue to provide high delivery rates.

In addition, Evergreen should

- be independent from the nodes' sleep cycles, and
- be hardware independent and work on every MAC layer (with minimal porting effort).

3.3.2 Contributions

Evergreen has the following main characteristics.

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. Our LQE is explained in Subsection 3.3.3
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 an alternative algorithm is used when the network is in a transitional phase and no route to a root node is known. The main and alternative forwarding algorithms are explained in Subsection 3.3.4.

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 and are discussed in Section 3.3.5.

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. We explain our control message design in Section 3.3.5

5. Miscellaneous Optimizations: Evergreen also employs several other optimizations including packet aggregation, packet queue management, and short-circuit data forwarding. We explain these optimizations in Section 3.3.6.

3.3.3 Link Quality Estimation

We designed our link quality estimation (LQE) protocol based on three main goals:

- To quickly identify a newly available link, even if that link is lossy.
- To quickly identify a link failure when a link goes down permanently or for an extended period of time.
- To keep LQE up-to-date even if data packets are not traversing through a set of links.

Achieving the first goal is essential so that even if the quality of all links is lossy, data packets still continue to flow though the network. The second goal is set so that packet losses and delay could be minimized by avoiding sending packets on the links that have gone down due to interference or temperature variations. The last goal is set so that link quality estimation is not dependent on the arrival of data packets alone and LQE always remains up-to-date to facilitate any future data forwarding needs.

Evergreen uses arrival and lack-of-arrival of data and control packets to estimate a link's quality. Given that a link may not always have data packets traversing through it, it would be desirable to have explicit link quality control packets. Furthermore, even if a link has frequent data packets passing through it that can be tracked to measure its quality, using control packets can help in faster convergence of the link's quality estimation.

Using Control Packets For LQE: A control packet is generated by a node at a pre-scheduled time interval when the node is not experiencing drastic changes in its neighborhood. Otherwise, a node may send an out-of-schedule control packet to propagate any drastic changes in its neighborhood. Unlike other protocols, in Evergreen a control packet received at a node also contains information about the next scheduled control packet. In case a control packet arrives

well in advance of its scheduled time it is ignored. However, in case the next control packet fails to arrive at its scheduled time, then this is used as an indication of a deterioration in link quality (referred to as a link-fault LF). Likewise, the arrival of a control packet at its scheduled time is taken as an improvement indicator of the link's quality (referred to as a link-success LS). Similarly if a link received an ACK for a data packet then it is taken as an LS, whereas a lacking ACK is considered an LF. Link quality estimation is updated based on the LSs and LFs, produced either by control or data packets.

3.3.4 Alternative Forwarding Algorithm

Evergreen has two forwarding algorithms: a primary and a secondary algorithm. The primary forwarding algorithm finds a path with the lowest aggregated LQE to the root node. This primary forwarding algorithm is similar to that of CTP and works well if the network is stable. However, the key problem with this algorithm is that in large WSNs it takes several seconds to propagate a route from a root node to the whole network. Thus, when the network undergoes some rapid changes (e.g. due to interference or node mobility), no route might be available for some time until the network converges to the new routes. In case of CTP, if a node cannot find a path to the root it uses a two step approach:

1. It starts a repair mechanism by triggering the Trickle timer.
2. The node waits for some time (10 seconds) for the network to converge, before attempting to forward the data packet again.

After the wait is over, but when a path has still not become available, the node repeats the above two-step process. This could lead to a long (even eternal) cycle of waits if the network is continuously changing (e.g. frequent high interference across the network links).

Evergreen uses a simplified version of BCP's forwarding algorithm [47] as its secondary forwarding algorithm. In case there is no route to the root available then, just like CTP, Evergreen starts the repair mechanism, but instead of waiting to send data it continues to send data, like BCP, using the queue lengths to the neighboring nodes favoring the least congested neighbor. Thus, in Evergreen, packets continue to flow even if the network is undergoing some rapid changes, leading to higher throughput and shorter delays.

3.3.5 Dynamic Timed and Sized Control Messages

To compute paths to a root node, a collection protocol uses a locally stored "network state". The network state maintained at a node represents how that node visualizes the network. An outdated network state may result in path computation errors. To keep the state up-to-date in its neighborhood, a node broadcasts its network state using control messages to its immediate neighbors, and based on the control messages it receives from its neighbors the node updates its own state. However, a node has to meticulously decide when to send a control message and what part of its state information should be shared. Frequently sending control messages will result in draining a node's battery lifetime as well as those of its neighbors. In contrast, not sending an important update will result in stale network state information at the neighbors. Thus, a node must try to minimize the number of control messages sent to save energy while

at the same time minimize the time the network takes to converge to a new state. Evergreen has two main contributions regarding this trade off:

1. It includes a comprehensive strategy that outlines when to send a control message.
2. It dynamically selects the information to be sent (reducing control message size) based on the current state of the network.

Sizing Control Messages: There are three kinds of tasks fulfilled by sending control messages: i) a root announcing its existence, ii) a node communicating its path cost to a root and iii) maintaining link quality.

CTP uses a 16-byte control message to fulfill these three tasks. In contrast, Evergreen uses three different-sized control messages. A root sends only 5-byte messages that contain information about its node ID, so other nodes can infer the root's existence. To maintain a link's existence and update its quality a 4-byte control message (C_l) is sent. Whereas to communicate any drastic changes in the network a 9-byte control message (say C_t) is sent.

Every time when a non-root node sends C_t it saves the information it has advertised. When the time arrives to send out the next control message, the node compares the information it has sent previously with the current state of the network. In case the network state has not changed significantly then instead of sending C_t, the node advertises C_l, which is less than half the size of C_t and contains only summarized information.

3.3.6 Miscellaneous Optimizations

Evergreen has many other small contributions and attributes that together improve its performance. This subsection lists them briefly.

Short-Circuit Data Forwarding: When a node receives a packet at the network layer (from the application layer or from a neighboring node), that packet enters the node's FIFO forwarding queue. The network layer takes packets from the queue and forwards them one by one. If the application is sending packets faster than the network layer's forwarding speed, then the FIFO queue will eventually overflow, leading to dropped packets. To avoid this situation, Evergreen computes the path to the root node only once every m milliseconds. Thus by amortizing the path computation overhead, Evergreen increases its forwarding speed; we call this optimization "Short-Circuit Data Forwarding". The value of m should not be too large so that packets are forwarded to the best possible path most of the time.

Data Packet Aggregation: Another measure taken by Evergreen to avoid overflow of its FIFO forwarding queue is data aggregation, effectively combining multiple data packets into one packet with a common 7-byte header. However, Evergreen only aggregates packets when the forwarding queue is almost full for the following reasons. First, any aggregated packet must be de-aggregated at the root node, increasing the overhead at root node. Second, the larger the data packet, the higher the chance of corruption due to interference, which has the additional effect of losing the data of multiple single packets at once. Lastly, a duplicated aggregated packet could actually result in increased radio duty cycles.

Faster Data Pull: In a typical collection protocol, an application receives an acknowledgment (or a negative acknowledgment) for each packet it sends to the network layer. This is done to provide the application an opportunity to resend the same packet to the network layer, in case of a negative acknowledgment. Such acknowledgments between the application and the network layer are referred to as *"application-level ACKs"*. Note that, an application is *not allowed* to send a packet (to the network layer) during the waiting period even if the application does not wish to resend the same packet. The network layer itself retries to send a packet several times if it does not receive an acknowledgment from its neighboring nodes. Such acknowledgments between two nodes are referred to as *"node-level ACKs"*.

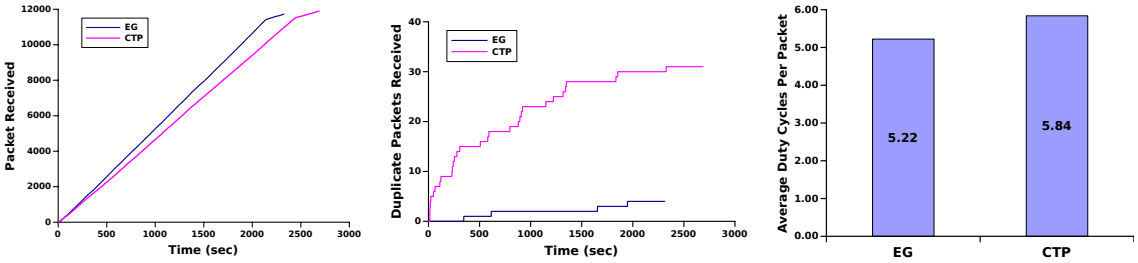
Both CTP and Evergreen deploy an aggressive retry policy that only drops a packet when no node-level ACK is received after retrying dozens of times. For instance, the current implementation of CTP retries 35 times to receive a node-level ACK before sending a negative application-level ACK. We believe that, this aggressive retry policy at the network level has rendered application-level ACKs useless. Therefore, in case of Evergreen, if the application has some packets to send then Evergreen does not force the application to wait for a packet application-level ACK. Instead Evergreen allows the application to send other data packets as long as the forwarding queue of Evergreen has enough empty slots to accommodate them. Note that, Evergreen also supports application-level ACKs but at the same time allows an application to ignore them. This is useful for increasing application throughput when a link experiences a short-lived interference; the buffered packets in the forwarding queue can be sent out in a burst once the link comes back to live.

3.3.7 Results and Discussion

This subsection compares Evergreen with the widely used collection protocol CTP. We used our Templab testbed for the experiments, which has 17 nodes [12]. TempLab is equipped with infrared heating lamps, to study the impact of temperature variations in real-time controlled environment. For the experiments, a root node and three source nodes 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 to find duplicate data packets received at the root node, at the end of an experiment, by processing the trace files. Each source node sends 4000 packets, hence, each experiment lasts at least $4000 \times 0.5 = 2000$ seconds (or 33.33 minutes). Each node has a half a second long sleep cycle. Both, Evergreen and CTP are MAC and hardware independent protocols and hence we used the default MAC protocol of TinyOS 2.x in our evaluations [49].

To have a fair comparison, the size of the forwarding queue of both Evergreen and CTP is kept the same. Also, in our experiments we have not used Evergreen's data aggregation module to give CTP the best possible chance to perform well when compared to Evergreen. The following subsections compare Evergreen with CTP in three different scenarios and Evergreen is shown to comprehensively outperform CTP in each one of these scenarios.

Static Network: We start by comparing Evergreen with CTP in a static network. In this scenario all the links of the network have favorable conditions. That is, links are not experiencing any interference or rapid temperature variations, links are only exposed to the normal effects



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

Figure 3.19: Static Network: Figure 3.19(a) outlines the number of packets received per second in the static network. Figure 3.19(b) shows the number of duplicate packets received at the root node per second for each protocol. Figure 3.19(c) depicts the duty cycles consumed on average to transmit a single data packet.

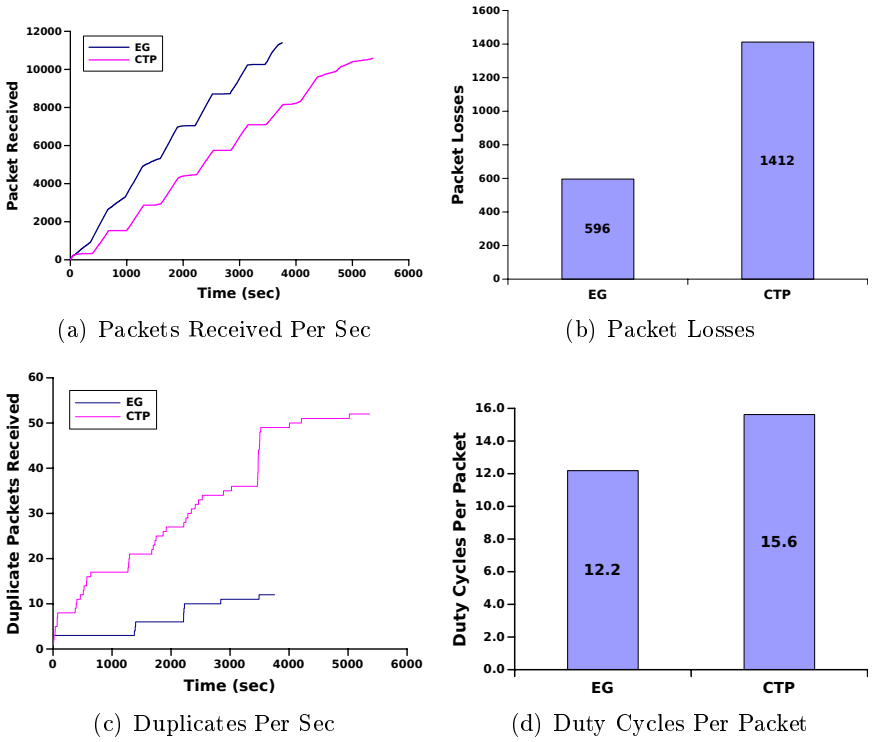
multipath RF. The results of this scenario are presented in Fig. 3.19. Evergreen takes 13.4% less time to transmit the set of data packets as compared to CTP (Figure 3.19(a)). While at the same time, Evergreen has a lower duty cycle, 10.6% on average. Furthermore, Evergreen reduces packet duplicates by 87%.

These results show that even under benign conditions Evergreen has a better performance than CTP, but Evergreen's main goal is to work in extreme conditions. We will now see how Evergreen's relative performance is even better under extreme conditions.

High Interference: The second scenario used to measure Evergreen's and CTP's performance is interference. To emulate interference used JamLab, a tool that generates controlled and realistic interference patterns [6]. JamLab can emulate different kinds of interference from Bluetooth to microwaves. For our experiments we used Jamlab to emulate Wi-Fi video streaming, which is a very heavy interference pattern. A centralized node in the network is selected as an interferer that can block all the links in the network when transmitting at maximum power. Hence, when a video streaming starts, no protocol can continue to communicate with the root node or with any other node in the network. During the course of the experiment, interference is switched *on* for a period of five minutes, followed by an interference-free period of five minutes and the cycle is repeated. Thus, under our settings, during the transmission of 2000 packets, interference is switched *on* seven times. In this type of scenarios, the challenge for a protocol is to configure itself quickly to transmit as many packet as possible in the five minutes when links are interference-free. This experiment and the subsequent experiment on temperature variations are mainly testing two qualities of a protocol:

- The ability of the protocol's link quality estimation to quickly detect changes in the environment and adapt to those changes.
- How effective a protocol is in disseminating sudden changes in link quality estimation using control messages.

If a protocol takes too much time to estimate link quality, then it will drop more packets as it will be slow in detecting deteriorating links. Also if a link starts improving, a slow estimation processes reduces the ability to send packets earlier. In Fig. 3.20(b) we observe that CTP drops



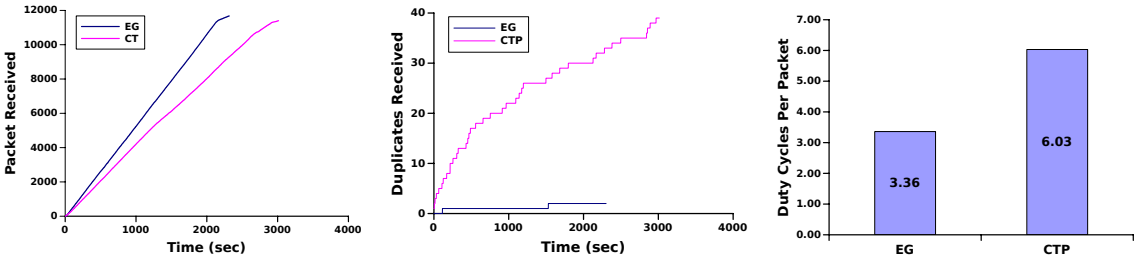
(a) Packets Received Per Sec (b) Packet Losses

(c) Duplicates Per Sec (d) Duty Cycles Per Packet

Figure 3.20: High Interference: Figure 3.20(a) outlines the number of packets received per second in the network that whose links are experiencing high interference. Figure 3.20(b) shows that CTP has 2.3 times higher loss of data packets as compared to Evergreen under high interference scenario. Figure 3.20(c) shows the duplicate data packet produce by Evergreen and CTP over time. Figure 3.20(d) Evergreen outperforms CTP by consuming significantly less duty cycles.

2.36 times more packets as compared to Evergreen, which is mostly attributed to Evergreen's ability to quickly estimate and disseminate link qualities. Furthermore, Evergreen is on average 53% faster in transmitting a single packet to the root node as compared to CTP, while at the same time having 22% less duty-cycles.

Temperature Variations We now present Evergreen's performance under temperature variations. Interference on a link can be switched on or off within a few milliseconds, however heating or cooling down a lamp takes several minutes. Similarly, in a real setting environment, interference is much more volatile then temperature. Thus protocols have much more time to adapt to the changes in network dynamics due to variations caused by temperature than by interference. 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. Fig. 3.21 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



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

Figure 3.21: Temperature Variations: Number of data packets and duplicate data packets received per second are shown in Fig. 3.21(a) and 3.21(b) respectively. Duty cycles consumed on average to transmit a single data packet are shown in Fig. 3.21(c).

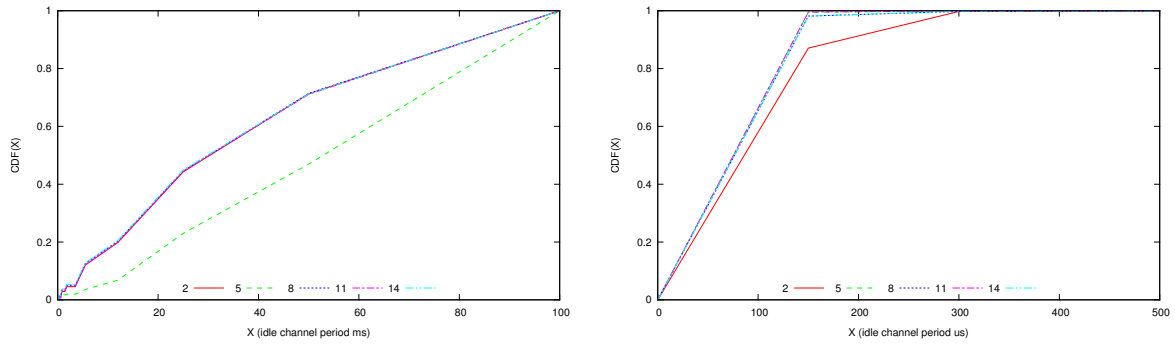
44% less duty cycles as compared to CTP while transmitting slightly more data packets in a significantly reduced time period. To be precise, Evergreen takes 33% less time to transmit a single data packet while delivering 2.5% more packets than CTP. There are multiple factors that contribute to Evergreen's 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.

3.4 Interference Mitigation

Many network protocols and mechanisms benefit from an understanding of achievable Packet Reception Rates (PRR) on links. If the expected PRR on links is known protocols can be optimised and it is possible to estimate achievable network performance. For example, a routing protocol may use PRR prediction to decide if a link should be pruned from the currently used topology. A multichannel MAC protocol may use PRR information to decide if a channel change is necessary. In a TDMA protocol PRR information may be used to decide how many transmission slots must be allocated to achieve sufficient reliability levels. With knowledge of PRR on each link the achievable end-to-end reliability in a network can be calculated. And finally, as transmission latency can be traded for transmission reliability (as retransmissions require time) PRR information can be used to reason on achievable end-to-end delay.

Transmissions on links used in a deployment can be monitored and the PRR can be recorded. The recorded PRR achieved in the past can then be used to estimate the future PRR on the link. This method is commonly used to estimate link quality in form of achievable PRR. Unfortunately, this technique is only useful to estimate PRR at the time that the network is operational and of links which are already in use. It is not possible to estimate PRR on links that may be used as alternative and, more importantly, it is not possible to estimate PRR in a deployment area before the network is deployed.

PRR is mainly affected by interference. Hence, it is possible to capture interference and to use this data to reason about achievable PRR. Thus, it is possible to overcome the shortcomings of the aforementioned PRR estimation technique based on recording of actual transmission results. Interference can be measured in a deployment area and the the obtained interference recordings can be used to estimate achievable PRR for links subject to this interference pattern. As it is



(a) Idle Periods

(b) Busy Periods

Figure 3.22: CDF for both IDLE and Busy periods for Channel 15.

not feasible to record directly interference over long time periods a method must be found that allows us to record essential characteristics of interference with reduced effort. This can be achieved by measuring and recording the statistical distributions of idle and busy interference periods. The resulting IDLE and BUSY Probability Density Function (PDF) contains sufficient information to estimate *PRR* in an environment subject to the interference.

We have described the methods for measuring the *IDLE-PDF* and *BUSY-PDF* in D1.1. How to derive the estimated *PRR* from a captured *IDLE-PDF* is described in D1.2. In this section we discuss how this mechanism can be used as a building block in the context of network protocols.

3.4.1 The *IDLE-PDF* and *BUSY-PDF*

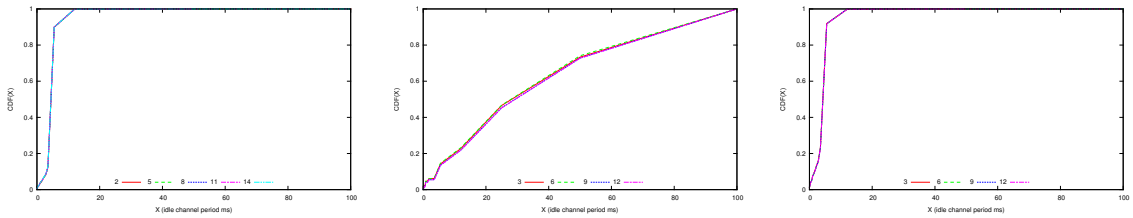
Interference levels can be measured by sampling the energy level in a transmission channel over time. If the sampled energy level is above a given threshold R_{Thr}, a packet transmission would be destroyed by concurrent activities in the frequency channel. Thus, interference can be represented as a sequence of idle (free channel) and busy (ongoing activity in the medium) periods of different length. Using such a recorded interference trace it is then possible to analyse success rates of packet transmissions.

To reduce the required storage space for interference traces we store the actual sequence of idle and busy periods and their respective length. Instead, we record the distributions of observed busy and idle period lengths. This measurement does not allow us to reproduce the exact measured interference trace but it allows us to produce an interference trace which exhibits the same statistical distributions of idle and busy periods and period lengths.

Figure 3.22 shows an example of a recorded IDLE-CDF and BUSY-CDF. We have described the methods for measuring the *IDLE-PDF* and *BUSY-PDF* in D2.2.

3.4.2 PRR Estimation Based on the IDLE-PDF

The recorded distribution of idle and busy period lengths (*IDLE-PDF* and *BUSY-PDF*) can be used for analysis of *PRR* instead of using an actual recorded interference trace (Alternatively



(a) Channel 12 (b) Channel 17 (c) Channel 21

Figure 3.23: CDF of idle periods for Channels 12, 17 and 21.

the *IDLE-CDF* and *BUSY-CDF* can be used as Cumulative Distribution Functions can be transformed into Probability Distribution Functions). It is assumed that a node carries out a Clear Channel Assessment (CCA) before transmission. Thus, transmissions are started in an idle period. The transmission is successful if the packet transmission is completed before the start of the next busy period. The *IDLE-PDF* can be used to determine (analytically or via a Monte Carlo simulation) the achievable *PRR*. Details on how to derive the estimated *PRR* from a captured *IDLE-PDF* is described in D2.2.

Figure 3.23 shows the *IDLE-CDF* for 3 different channels recorded at different times of day. As it can be seen, interference levels on different channels are distinctively different. It is also shown that measurements taken at different times provide the same *IDLE-CDF* which indicates that in this example interference levels are constant over time. For this example, the expectd *PRR* can be computed for the different channels. As input for the calculation the *IDLE-CDF* and the packet size L is used. Using $L = 50$ bytes a *PRR* of $P_{12} = 70.5\%$, $P_{17} = 95.8\%$ and $P_{21} = 73\%$ for the three different transmission channels is calculated. As the results show, Channel 17 provides the highest *PRR*. The *IDLE-CDF* shape helps to explain this outcome; for Channel 17 it is more likely to encounter long idle periods than for the other channels and hence transmissions have a higher success rate as they have a better chance to complete before being interfered with.

3.4.3 Application Examples of PRR Estimation

The *PRR* information extracted from the interference measurement via *IDLE-CDF* can be used to tune network protocols and allows us to reason on achievable end-to-end transmission reliability and delay.

Packet Size Selection

The most simple application of the previously described mechanism is to use expected *PRR* to select a packet size L such that a given reliability goal can be achieved. For example, an application may have the requirement to deliver messages with a reliability of 99%. Interference present in the deployment area can be measured and the *IDLE-CDF* is constructed. Now it is possible to calculate which maximum size L can be used to ensure a packet delivery reliability above 99%. In some application scenarios it is possible to reduce packet sizes. For example,

a packet may contain necessary and optional information; in a sensor network a packet may contain the essential sensor reading and optional data such as node statistics (battery level, transmission statistics, ...). In other scenarios it may not be possible to adjust the volume of data that has to be transmitted but it might be possible to distribute the data onto a number of transmissions. In this case the *PRR* estimation can be used to decide on how many packets should be used to deliver data.

Channel Ranking and Selection

Communication in 802.15.4 IoT deployments is often subject to interference by other wireless systems such as Wi-Fi. A common method for interference mitigation is to identify 802.15.4 communication channels which are free of interference. Unfortunately, there are only 16 channels available and all are potentially subject to WiFi interference. As WiFi is used very widely it is often next to impossible to find a transmission channel which is free of any interference. Hence, the only option is to rank channels according to the levels of interference and to then select the best channel for communication. A number of existing work has addressed this domain and a different methods exist. The existing methods differ regarding the metric used for channel ranking and the method used to obtain measurement data for channel ranking.

We suggest to use the *PRR* estimation based on a measured *IDLE-CDF* in a deployment area as channel ranking metric. The estimated *PRR* gives a very good indication on how useful a transmission channel is for transmitting packets of a given size L. Observed interference patterns are directly related to potential transmission success in the environment.

MAC Protocol Dimensioning

Medium Access Control protocols generally provide a mechanism for packet re-transmission. Elements of the retransmission mechanism can be configured before deployment. For this configuration it is often necessary to estimate the number of expected retransmissions. If the estimation is too conservative the protocol may operate inefficiently.

For example, in an asynchronous MAC protocol such as ContikiMAC nodes wake periodically to check for incoming messages. More frequent wake-ups and/or longer wake periods improve transmission reliability. However, increasing wake-up frequency and duration decreases a nodes lifetime as more energy is consumed. Thus, an indication on achievable *PRR* can provide information on how to tune a MAC protocol for a given interference environment.

4 Conclusions

The need to develop new protocols for wireless sensor and actuator networks arises from the fact that the state-of-the-art is still highly vulnerable to interference and temperature phenomena. This document described the efforts of the RELYonIT consortium in deriving a new generation of protocols that are robust to interference and temperature effects. After following a detailed analysis to identify the shortcomings of current protocols, we enhanced the performance of four key mechanisms of the network stack: our Temperature-Aware MACs overcome the limitations of the *clear channel assessment* process under high temperatures, MiCMAC leverages *multiple channels* to skip the ones that have high interference, JAG provides a resilient *handshake mechanism* against interference, and the *variable packet size* method exploits the idle times in-between busy (interfering) periods to minimise packet losses. We also identified the common effects of temperature and interference on link dynamics, and we present our initial work on a protocol that aims to be resilient at both phenomena (Evergreen).

This deliverable marks a middle-point in the technical development of the RELYonIT project. Based on the insights obtained in the first part of the project (from WP1), we have proposed new mechanisms to overcome the effects of temperature and interference. The next step is to develop mathematical models for these protocols to provide quality of service guarantees.

Bibliography

[1] T. I. 802.15.4e Task Group, "Ieee standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (lr-wpans) amendment 1: Mac sublayer," *IEEE Std 802.15.4e-2012 (Amendment to IEEE Std 802.15.4-2011)*, April 2012.

[2] B. Azimi-Sadjadi, D. Sexton, P. Liu, and M. Mahony, "Interference effect on ieee 802.15. 4 performance," in *Proceedings of 3rd International Conference on Networked Sensing Systems (INNS), Chicago, IL*, 2006.

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

[4] J. Beutel, B. Buchli, F. Ferrari, M. Keller, L. Thiele, and M. Zimmerling, "X-SENSE: Sensing in extreme environments," in *Proc. of DATE*, 2011.

[5] S. Bhadra, S.-H. Choi, Y. Sun, and X. Lu, "Demo: Achieving a 10x lifetime increase with ieee 802.15.4e motes," in *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*. New York, NY, USA: ACM, 2011, pp. 375–376.

[6] C. Boano, T. Voigt, C. Noda, K. Römer, and M. Zúñiga, "JamLab: Augmenting Sensornet Testbeds with Realistic and Controlled Interference Generation," in *Proceedings of the 10th international conference on information processing in sensor networks (IPSN)*, 2011.

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

[8] C. A. Boano, Z. He, Y. Li, T. Voigt, M. Zuniga, and A. Willig, "Controllable Radio Interference for Experimental and Testing Purposes in Wireless Sensor Networks," in *Proceedings of the 4th International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, Zurich, Switzerland, Oct. 2009.

[9] 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, 2010.

[10] C. A. Boano, T. Voigt, N. Tsiftes, L. Mottola, K. Römer, and M. A. Zúñiga, "Making sensornet mac protocols robust against interference," in *Proceedings of the 7th European conference on Wireless Sensor Networks*, ser. EWSN'10.

Berlin, Heidelberg: Springer-Verlag, 2010, pp. 272–288. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-11917-0_18

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

[12] 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 WSNs," in *Proc. of the* 13^{th} *IPSN*, 2014.

[13] J. Borms, K. Steenhaut, and B. Lemmens, "Low-overhead dynamic multi-channel mac for wireless sensor networks," in *EWSN*, 2010, pp. 81–96.

[14] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Boulder, Colorado, USA, 2006.

[15] C. A. Boano et al., "Hot Packets: A systematic evaluation of the effect of temperature on low power wireless transceivers," in *Proc. of the* 5^{th} *ExtremeCom*, 2013.

[16] C. Noda et al., "Quantifying the Channel Quality for Interference-Aware Wireless Sensor Networks," *ACM SIGBED Review*, vol. 8, no. 4, 2011.

[17] 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.

[18] W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney, "A Key Management Scheme for WSN using Depl. Knowledge," in *23rd INFOCOM*, 2004.

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

[20] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors," in *Proceedings of the Conference on Local Computer Networks (IEEE LCN)*, 2004.

[21] S. Duquennoy, O. Landsiedel, and T. Voigt, "Let the Tree Bloom: Scalable Opportunistic Routing with ORPL," in *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys 2013)*, Rome, Italy, Nov. 2013.

[22] E. Akkoyunlu et al., "Some Constraints and Tradeoffs in the Design of Network Communications," in *Symp. on Op. Syst. Princ. (SOSP)*, 1975.

[23] G. Huang et al., "Beyond Co-Existence: Exploiting WiFi White Space for Zigbee Performance Assurance," in *18th IEEE ICNP*, 2010.

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

[25] J. Gray, "Notes on Data Base Operating Systems," in *Operating Systems, an Advanced Course, pp. 393–481*, 1978.

[26] W. Guo, W. M. Healy, and M. Zhou, "Experimental study of the thermal impacts on wireless sensor batteries," in *Proc. of the* 10^{th} *ICNSC*, 2013.

[27] J. Hauer, A. Willig, and A. Wolisz, "Mitigating the Effects of RF Interference through RSSI-Based Error Recovery," in *7th EWSN*, 2010.

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

[29] O. D. Incel, P. Jansen, and S. Mullender, "Mc-lmac: A multi-channel mac protocol for wireless sensor networks," Centre for Telematics and Information Technology University of Twente, Enschede, The Netherlands, Tech. Rep. TR-CTIT-08-61, 2008.

[30] V. Iyer, M. Woehrle, and K. Langendoen, "Chrysso: A multi-channel approach to mitigate external interference," in *8th IEEE SECON*, 2011.

[31] J. Brown et al., "BurstProbe: Debugging Time-Critical Data Delivery in Wireless Sensor Networks," in *8th EWSN*, 2011.

[32] K. Jamieson and H. Balakrishnan, "PPR: Partial Packet Recovery for Wireless Networks," in *ACM SIGCOMM*, 2007.

[33] K. Chowdhury and I. Akyildiz, "Interferer Classification, Channel Selection and Transmission Adaptation for WSN," in *IEEE ICC*, 2009.

[34] A. Kamerman and N. Erkocevic, "Microwave Oven Interf. on Wireless LANs Operating in the 2.4 GHz ISM Band," in *IEEE PIRMC'97*, 1997.

[35] C. Keppitiyagama, N. Tsiftes, C. A. Boano, and T. Voigt, "Temperature hints for sensornet routing," in *Proc. of the* 11^{th} *SenSys*, 2013.

[36] Y. Kim, H. Shin, and H. Cha, "Y-mac: An energy-efficient multi-channel mac protocol for dense wireless sensor networks," in *Proceedings of the 7th international conference on Information processing in sensor networks*, ser. IPSN '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 53–63. [Online]. Available: http://dx.doi.org/10.1109/IPSN.2008.27

[37] D. E. Knuth, *The art of computer programming, volume 2 (3rd ed.): seminumerical algorithms*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997.

[38] M. Kovatsch, S. Duquennoy, and A. Dunkels, "A Low-Power CoAP for Contiki," in *Proceedings of the Workshop on Internet of Things Technology and Architectures (IEEE IoTech 2011)*, Valencia, Spain, Oct. 2011.

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

[40] H. K. Le, D. Henriksson, and T. Abdelzaher, "A practical multi-channel media access control protocol for wireless sensor networks," in *Proceedings of the 7th international conference on Information processing in sensor networks*. IEEE Computer Society, 2008, pp. 70–81.

[41] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, "The trickle algorithm," IETF, Tech. Rep., 2011.

[42] Q. Li, M. Martins, O. Gnawali, and R. Fonseca, "On the effectiveness of energy metering on every node," in *Proc. of the* 10^{th} *DCOSS*, 2013.

[43] C. Liang, N. Priyantha, J. Liu, and A. Terzis, "Surviving wi-fi interference in low power zigbee networks," in *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Zurich, Switzerland, Nov. 2010.

[44] C. Liang, J. Liu, L. Luo, A. Terzis, and F. Zhao, "Dcnet: A high-fidelity data center sensing network," in *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, 2009.

[45] Matteo Bertocco and Giovanni Gamba and Alessandro Sona, "Experimental Optimization of CCA Thresholds in Wireless Sensor Networks in the Presence of Interference," in *Proceedings of the Workshop on ElectroMagnetic Compatibility (EMC)*, Honolulu, Hawaii, USA, Jul. 2007.

[46] Mo Sha and Gregory Hackmann and Chenyang Lu, "Energy-efficient low power listening for wireless sensor networks in noisy environments," in *Proc. of the* 12^{th} *IPSN*, 2013.

[47] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing without routes: The backpressure collection protocol," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM, 2010, pp. 279–290.

[48] D. Moss and P. Levis, "BoX-MACs: Exploiting Physical and Link Layer Boundaries in Low-Power Networking," Stanford University, Tech. Rep. SING-08-00, 2008.

[49] ——, "Box-macs: Exploiting physical and link layer boundaries in low-power networking," *Computer Systems Laboratory Stanford University*, 2008.

[50] R. Musaloiu-E. and A. Terzis, "Minimising the Effect of WiFi Interference in 802.15.4 WSN," *IJSNet*, vol. 3, no. 1, pp. 43–54, Dec. 2007.

[51] C. Park, K. Lahiri, and A. Raghunathan, "Battery discharge characteristics of wireless sensor nodes: An experimental analysis," in *Proceedings of the IEEE Conf. on Sensor and Ad-hoc Communications and Networks (SECON)*, Santa Clara, California, USA, Sep. 2005.

[52] J. Polastre, J. Hill, and D. Culler, "Versatile Low-Power Media Access for Wireless Sensor Networks," in *2nd ACM SenSys*, 2004.

[53] W.-B. Pöttner, S. Schildt, D. Meyer, and L. Wolf, "Piggy-Backing Link Quality Measurements to IEEE 802.15.4 ACKs," in *8th MASS*, 2011.

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

[55] B. Raman, K. Chebrolu, S. Bijwe, and V. Gabale, "PIP: A Connection-Oriented, Multi-Hop, Multi-Channel TDMA-based MAC for High Throughput Bulk Transfer," in *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Zürich, Switzerland, 2010.

[56] M. Sha, G. Hackmann, and C. Lu, "ARCH: Practical Channel Hopping for Reliable Home-Area Sensor Networks," in *17th IEEE RTAS*, 2011.

[57] T. Abdelzaher et al., "EnviroTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks," in *24th ICDCS*, 2004.

[58] L. Tang, Y. Sun, O. Gurewitz, and D. B. Johnson, "Em-mac: a dynamic multichannel energy-efficient mac protocol for wireless sensor networks," in *Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '11. New York, NY, USA: ACM, 2011, pp. 23:1–23:11. [Online]. Available: http://doi.acm.org/10.1145/2107502.2107533

[59] T. Voigt and F. Österlind, "CoReDac: Collision-Free Command-Response Data Collection," in *13th IEEE ETFA*, 2008.

[60] L. Wanner, C. Apte, R. Balani, P. Gupta, and M. Srivastava, "Hardware variability-aware duty cycling for embedded sensors," *IEEE Transactions on VLSI Systems*, vol. 21, no. 6, 2013.

[61] T. Watteyne, A. Mehta, and K. Pister, "Reliability through frequency diversity: why channel hopping makes sense," in *Proceedings of the 6th ACM symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, ser. PE-WASUN '09. New York, NY, USA: ACM, 2009, pp. 116–123. [Online]. Available: http://doi.acm.org/10.1145/1641876.1641898

[62] T. Watteyne, S. Lanzisera, A. Mehta, and K. S. J. Pister, "Mitigating multipath fading through channel hopping in wireless sensor networks," in *ICC*. IEEE, 2010, pp. 1–5.

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

[64] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6550.txt

[65] Y. Wu, J. A. Stankovic, T. He, and S. Lin, "Realistic and efficient multi-channel communications in wireless sensor networks," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE, 2008, pp. 1193–1201.

[66] L. Xiao, S. Boyd, and S. Lall, "A Scheme for Robust Distributed Sensor Fusion based on Average Consensus," in *4th ACM/IEEE IPSN*, 2005.

[67] X. Xu, J. Luo, and Q. Zhang, "Design of non-orthogonal multi-channel sensor networks," in *Proc. of the* 30^{th} *ICDCS*, 2010.

[68] W. Yuan, J.-P. M. Linnartz, and I. G. M. M. Niemegeers, "Adaptive CCA for IEEE 802.15.4 wireless sensor networks to mitigate interference," in *Proc. of IEEE WCNC*, 2010.

[69] T. ZHENG, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," *Listening*, vol. 10, p. 8, 2004.

[70] G. Zhou, J. Stankovic, and S. Son, "Crowded Spectrum in Wireless Sensor Networks," in *3rd Worksh. on Emb. Net. Sens. EmNetS*, 2006.