Technical Report

JamLab: Augmenting Sensornet Testbeds with Realistic and Controlled Interference Generation

Carlo Alberto Boano

Thiemo Voigt

Claro Noda

Kay Römer

and Marco Zúñiga

HURRAY-TR-110708

Version:

Date: 07-03-2011

JamLab: Augmenting Sensornet Testbeds with Realistic and Controlled Interference Generation

Carlo Alberto Boano, Thiemo Voigt, Claro Noda, Kay Römer, and Marco Zúñiga

IPP-HURRAY!

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail:

http://www.hurray.isep.ipp.pt

Abstract

Radio interference drastically affects the performance of sensornet communications, leading to packet loss and reduced energy efficiency.As an increasing number of wireless devices operates on the same ISM frequencies, there is a strong need for understanding and debugging the performance of existing sensornet protocols under interference. Doing so requires a low-cost flexible testbed infrastructure that allows the repeatable generation of a wide range of interference patterns. Unfortunately, to date, existing sensornet testbeds lack such capabilities, and do not permit to study easily the coexistence problems between devices sharing the same frequencies.This paper addresses the current lack of such an infrastructure by using off-the-shelf sensor motes to record and playback interference patterns as well as to generate customizable and repeatable interference in real-time. We propose and develop JamLab: a low-cost infrastructure to augment existing sensornet testbeds with accurate interference generation while limiting the overhead to a simple upload of the appropriate software. We explain how wetackle the hardware limitations and get an accurate measurement and regeneration of interference, and we experimentally evaluate the accuracy of JamLab with respect to time, space, and intensity. We further use JamLab to characterize the impact of interference on sensornet MAC protocols.

JamLab: Augmenting Sensornet Testbeds with Realistic and Controlled Interference Generation

Carlo Alberto Boano[†], Thiemo Voigt[‡], Claro Noda[¶], Kay Römer[†], and Marco Zúñiga[§]

[†]Institute of Computer Engineering
University of Lübeck, Germany
{cboano, roemer}@iti.uni-luebeck.de

[‡]Swedish Institute of Computer Science
Kista, Sweden
thiemo@sics.se

[¶]CISTER Research Unit
Polytechnic Institute of Porto, Portugal
cand@isep.ipp.pt

[§]Networked Embedded Systems Group
University of Duisburg-Essen, Germany
marco.zuniga@uni-due.de

ABSTRACT

Radio interference drastically affects the performance of sensornet communications, leading to packet loss and reduced energy-efficiency. As an increasing number of wireless devices operates on the same ISM frequencies, there is a strong need for understanding and debugging the performance of existing sensornet protocols under interference. Doing so requires a low-cost flexible testbed infrastructure that allows the repeatable generation of a wide range of interference patterns. Unfortunately, to date, existing sensornet testbeds lack such capabilities, and do not permit to study easily the coexistence problems between devices sharing the same frequencies. This paper addresses the current lack of such an infrastructure by using off-the-shelf sensor motes to record and playback interference patterns as well as to generate customizable and repeatable interference in real-time. We propose and develop JamLab: a low-cost infrastructure to augment existing sensornet testbeds with accurate interference generation while limiting the overhead to a simple upload of the appropriate software. We explain how we tackle the hardware limitations and get an accurate measurement and regeneration of interference, and we experimentally evaluate the accuracy of JamLab with respect to time, space, and intensity. We further use JamLab to characterize the impact of interference on sensornet MAC protocols.

Categories and Subject Descriptors

B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids.

General Terms

Design, Experimentation, Measurement, Performance, Reliability.

Keywords

JamLab, HandyMote, Interference Generation, Wireless Sensor Networks, Augmenting Sensornet Testbeds.

1. INTRODUCTION

The reliability and robustness of sensornet communications are affected by radio interference. As an increasing number of standardized communication technologies operate in ISM bands, the congestion in the radio spectrum is inflating, and the quality of communications decreases. In safety-critical sensornet applications such as industrial automation and health care, in which the reliability and stability of communications are vital, radio interference represents a major challenge, as it leads to packet loss, high latencies, and reduced energy-efficiency due to retransmissions.

This issue is especially serious in the 2.4 GHz ISM band, as wireless sensor networks that operate at such frequencies must compete with the ongoing communications of WLAN, Bluetooth, and other IEEE 802.15.4 devices. Furthermore, sensornet communications in these frequencies can also be affected by several domestic appliances that are source of electromagnetic noise, such as microwave ovens, video-capture devices or baby monitors. This high number of different wireless devices sharing the same frequencies and space, raises the need for coexistence and interference mitigation techniques in 802.15.4-based sensor networks, as highlighted by previous studies [1, 2].

In particular, there is a strong need for understanding the performance of existing sensornet protocols under interference, as well as designing novel protocols that can deliver high and stable performance despite changing interference patterns. This, however, requires a proper testbed infrastructure where realistic interference patterns can be easily created in a precise and repeatable way. Unfortunately, existing sensornet testbeds lack such capabilities for interference generation, or they are limited to static WiFi access points randomly placed in the testbed [3], which does not enable the creation of a wide range of interference patterns in a repeatable way. Upgrading existing testbeds with additional heterogeneous devices in order to introduce interference sources is a costly, inflexible, labor-intensive, placement-dependent operation.

We therefore propose to augment existing sensornet testbeds with JamLab, a low-cost infrastructure for the creation of realistic and repeatable interference patterns. Such an infrastructure should support the recording and playback of interference traces in sensornet testbeds, as well as the customizable generation of typical interference patterns resulting from WiFi, Bluetooth, microwave ovens, or any other device operating in the frequency of interest.

To ensure a low-cost and hence widely applicable solution, we propose to use off-the-shelf motes. In this way, a fraction of the already deployed nodes of a testbed could be used for interference generation with the overhead limited to the simple uploading of the appropriate software. However, building such a low-cost solution is challenging due to the limitations of the available hardware. In order to obtain an accurate playback, the interference-pattern levels need to be measured precisely at a high sampling rate, so that also short interference patterns (e.g., resulting from WiFi traffic) can be recognized. We show in the paper how to obtain accurate readings of the RSSI noise floor while achieving a sampling frequency at up to 60 kHz. We show how at such high frequencies, many erroneous RSSI readings occur, and we correct such wrong readings by properly configuring the internal automatic gain control of the CC2420 radio. As a side effect, this technique also increases significantly the efficiency of Clear Channel Assessment (CCA) under interference. We exploit this approach to study the spectro-temporal characteristics of the most common interference sources such as WiFi, Bluetooth, and microwave ovens.

We further analyze and tackle the problem of (re)generating interference: the patterns have to be reproduced accurately in both frequency and time domains. This turned out to be hard to obtain, given the coarse output power levels available from the radio transceiver and the limited memory available on the motes. We show that to achieve an accurate regeneration, voluminous records of interference patterns need to be stored on the mote in real-time and later played back accordingly. Moreover, we provide precise and lightweight models of common interference sources in the 2.4 GHz ISM band to generate (emulate) realistic patterns.

Finally, the placement of the nodes inside the testbed is also critical. We study the implications in the spatial domain when measuring and generating interference in an indoor testbed and propose an optimal placement of the sensor nodes.

Our paper proceeds as follows: Section 2 describes the architecture of JamLab. Section 3 describes how we can use common sensor motes to accurately measure interference at high sampling rates avoiding erroneous RSSI readings. In Section 4 we show how to reproduce customized and repeatable interference patterns using sensor motes. In Section 5, we model several interference sources and show how we emulate specific interference patterns. We then discuss in Section 6 how to configure the testbed and the placement of nodes. We evaluate JamLab's accuracy in Section 7, and show how to practically augment an existing sensornet infrastructure. We further exploit JamLab to characterize the performance of sensornet protocols under emulated, but realistic interference. We review related work in Section 8 and conclude our paper in Section 9.

2. JAMLAB OVERVIEW

In sensor networks – especially in safety-critical applications with stringent quality-of-service requirements – robustness against interference is crucial. Interference may not only increase packet loss, the number of retransmissions, and therefore also power consumption, but timing constraints of the application may be violated and lead to failures. The observation that the environment has a profound impact on radio propagation has led to the excessive use of testbeds by the sensornet community, as simplified simulation models of radio propagation do not capture the complexity of the real world. The same holds true for interference: testbed infrastructures need to be augmented with means to generate realistic interference patterns in a repeatable manner to develop, test, and evaluate sensornet protocols and applications under interference.

With JamLab we propose a *low-cost* approach to augment *existing* testbeds with a way to generate *realistic* and *repeatable* interference patterns. The key idea behind JamLab is to use off-the-shelf motes to record and playback interference patterns instead of bringing WiFi access points, microwave ovens, or other equipment to the testbed. The latter approach is not only costly and hard to reproduce exactly by other researchers, but it is even difficult to exactly reproduce a given interference pattern with the same appliance. For example, the sequence and timing of the WiFi frames generated by a file download may differ between repeated trials due to TCP adaptation mechanisms (e.g., timeouts, window sizes). Furthermore, every device used to generate interference in the testbed needs to be programmed remotely. Programming several heterogeneous devices such as WiFi access points or microwave ovens would create a significant overhead, whereas using JamLab the installation overhead is minimal.

Indeed, with JamLab, either a fraction of the existing nodes in a testbed are used to record and playback interference patterns, or a few additional motes are placed in the testbed area. We call those motes used for interference generation *HandyMotes*. The HandyMotes support two modes of operation: *emulation*, where a simplified model is used to generate interference patterns that resemble those generated by a specific appliance (such as a WiFi device or a microwave oven); and *regeneration*, where each HandyMote autonomously samples the actual interference, compresses and stores it locally, and regenerates the recorded patterns later. The latter mode is especially useful to record realistic interference patterns in a crowded shopping center or on a lively street by placing a few HandyMotes to record interference, and bringing them to the testbed to playback the recorded traces there.

One fundamental challenge results from the fact that the maximum RF output power of motes (0 dBm) is typically much smaller than the RF output of other typical interference sources (25 and 60 dBm for WiFi and microwave ovens, respectively). Therefore, a WiFi transmitter or a microwave oven may disturb sensornet communications over much larger distances than a HandyMote can. We address this issue by subdividing the testbed area into cells as depicted in Figure 1, such that a HandyMote placed at the center of the cell can interfere with all testbed motes contained in the cell, but the interference with motes outsides of the cell is minimized. This requires a careful placement or selection of HandyMotes and control of their RF output power. We investigate this issue and propose a procedure for HandyMote placement and power control in Section 6. Note that there is a tradeoff between the realism of the generated interference patterns and the number of HandyMotes: the more cells, the more accurate is the spatial distribution of interference, but the more HandyMotes are required.

Another challenge is that many interference sources emit wideband signals, i.e., they interfere with many 802.15.4 communication channels at the same time. In contrast, a mote can only transmit on a single channel at a time. Fortunately, most existing sensornet protocols use only a single channel. However, there is a trend to use multiple 802.15.4 channels at different nodes to increase robustness and bandwidth. Our approach to deal with this issue is to place multiple HandyMotes in each cell, each one interfering on one 802.15.4 channel as detailed in Section 4.3. The use of Software Defined Radio (SDR) techniques using USRP devices would provide more accurate jamming signals on a wider bandwidth, but their high cost represents a sizeable limitation. To synchronize the generation of interference patterns within the HandyMotes in one cell and across cells, we need time synchronization, and we propose to use the testbed infrastructure (i.e., wired backchannels) to send synchronization signals to the HandyMotes.

Due to the constrained resources of a mote, also the accurate recording and playback of interference represent a challenge. To capture short interference patterns such as those generated by WiFi beacons, we need high sampling rates with low jitter, which requires data compression due to the limited amount of available memory. Our solutions to these problems are described in Sec-

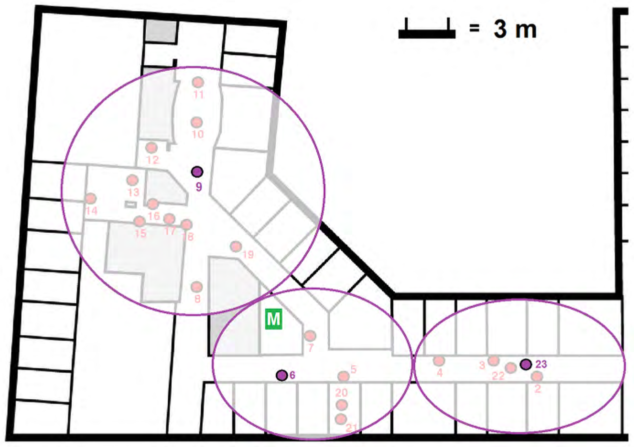


Figure 1: Testbed augmented with JamLab. Nodes 6, 9, and 23 are selected as HandyMotes, and take care of interference (re)generation in their cell.

tion 4.1. The accurate measurement of the interfering signal strength turned out to be a challenge in itself due to the gain control in the radio. Our solution to this problem is detailed in Section 3.

For the playback of recorded interference traces, normal packet transmissions are not appropriate, as this would offer only limited control over the exact timing of the transmitted signals. Therefore, we use special test modes of 802.15.4 radios to generate modulated or unmodulated carrier signals as detailed in Section 4.2. Those radios offer only a small number of discrete output power levels. While this can be exploited for compression of recorded traces, it limits the control over the generated interfering signal strength. However, we are primarily interested in *binary interference* generation, where a HandyMote either blocks the communication of the motes in its cell by emitting a strong-enough interference signal, or by not interfering at all. Nevertheless, HandyMote also supports the generation of a small number of output power levels as supported by the radio hardware, as discussed in Section 4.1.

JamLab has been designed specifically for the Texas Instruments CC2420 radio [4], and tested on several sensor motes such as Maxfor MTM-CM5000MSP, Crossbow TelosB, and Sentilla JCreate, but the framework can be applied to any sensornet platform. Based on the analysis of the datasheets, the Handymotes should be easily ported to similar radios such as the Ember EM2420 transceiver, and to newer radios such as the CC2520. We develop the HandyMotes based on Contiki, a lightweight and flexible operating system for tiny networked sensors [5].

3. MEASURING INTERFERENCE ACCURATELY USING MOTES

Measuring interference accurately on a mote is a key functionality, both for recording and later playback of interference, as well as for acquiring a deep understanding of common interference sources such as WiFi or Bluetooth. We describe in this section the techniques we used in order to let a common sensor mote measure the interference accurately at a sufficiently high sampling rate.

3.1 Measuring at High Sampling Rates

Link quality indicators such as RSSI and LQI provide an indication of the signal strength and quality, but only upon the reception of a packet. The only feasible way to assess the interference status is hence the continuous measurement of the RSSI noise floor, i.e., the RSSI in absence of packet transmissions.

In order to retrieve the spectro-temporal characteristics of different interference sources, we improve existing Contiki tools [6] and develop two applications that scan the 2.4 GHz frequency spectrum by reading the RSSI noise floor from the CC2420 radio transceiver:

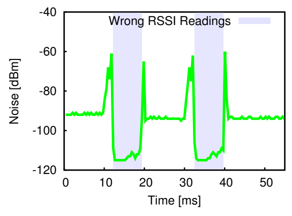
- The *time scanner* scans a single predefined IEEE 802.15.4 channel at its middle frequency with a very high sampling rate, and returns the RSSI noise floor readings over time;
- The *frequency scanner* scans sequentially the whole 2.4 GHz spectrum by switching between all 802.15.4 channels.

A first requirement of both scanners is to achieve a high sampling rate, given that we need to detect short transmissions periods. After boosting the CPU speed, optimizing the SPI operations, as well as buffering and compressing the RSSI noise floor readings using Run-Length Encoding (RLE), we reached a maximum sampling rate of approximately 60.5 kHz when sampling a single channel with the *time scanner*. The highest sampling frequency reachable by the *frequency scanner* is instead 3.4 kHz, since it is constrained by the settling time of the radio when switching channels. Hence, the limitations of low-power radios do not permit to achieve a sampling rate sufficiently high to capture all WiFi transmissions, as the maximum speed of 802.11b/g/n standards is 11, 54, and 150 Mbit/s, respectively. The minimum size of a WiFi packet is 38 bytes (ACK and CTS frames), which would make a resolution of 60 kHz sufficient to detect all 802.11b frames, but not all 802.11g/n frames. As most WiFi frames are data frames and typically contain higher layer headers, one can sample at 60 kHz frames with TCP/IP headers having a payload size higher than 27 and 227 bytes for 802.11g/n, respectively. Despite the use of large PDUs to reduce preamble overhead [7], this resolution does not guarantee to capture all the VoIP traffic over 802.11g/n [8].

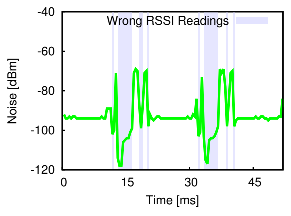
Another requirement for the scanners is to accurately measure the strength of the ongoing interference in the radio spectrum by means of precise RSSI noise floor readings. The CC2420 radio specifies an accuracy of ± 6 dBm, and a linearity of ± 3 dB in the dynamic range $[-100, 0]$ dBm. Such accuracy and linearity has so far been acknowledged by the research community as enough to carry out operations such as Clear Channel Assessment (CCA) and low-power channel sampling for activity recognition [9]. However, our experiments show that the RSSI noise floor readings captured at high sampling rate suffer of a systematic problem in three specific scenarios, namely: (i) when a narrow unmodulated carrier is transmitted, (ii) when microwave ovens are switched on, and (iii) in the presence of Bluetooth transmissions. In these scenarios, the CC2420 radio often returns RSSI values that are significantly below the supported range and the sensitivity threshold, e.g., -110 or -115 dBm. Figure 2 reports examples of such wrong readings, which represent an important problem, since they also impact the correct functioning of CCA in the presence of narrow-band signals, as shown in Figure 2(c). Our investigation also shows that the same problems applies to other sensornet platforms employing similar versions of the chip, such as the Ember EM2420 transceiver. We experimentally identified that the problem is due to the saturation of the Intermediate Frequency (IF) amplifier chain: we have observed that maximum gain is used in the Variable Gain Amplifier (VGA) when the incorrect RSSI readings occur.

3.2 Avoiding Saturation in RSSI Readings

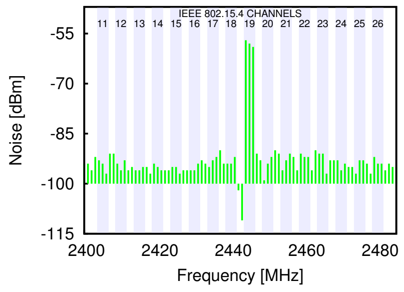
The reason of this saturation problem can be found in the radio demodulation chain. The CC2420 chip implements part of the IF filtering in analog domain and further filtering is later performed in digital domain. It employs an Automatic Gain Control (AGC) loop to maintain the signal amplitude close to a certain target value that guarantees the correct operation of the Analog-to-Digital Converter



(a) Active Microwave Oven



(b) Bluetooth Transmission



(c) Unmodulated Carrier

Figure 2: Examples of wrong RSSI readings: several values are significantly below the sensitivity threshold of -100 dBm due to receiver saturation. This error is caused by an incorrect operation of the AGC loop in presence of narrow-band signals.

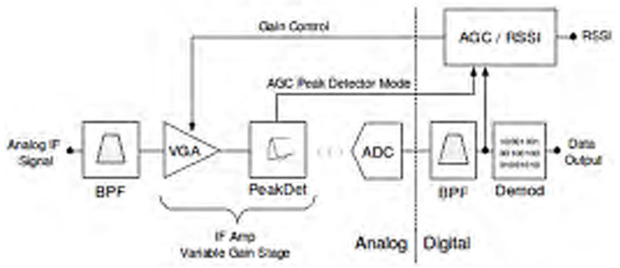


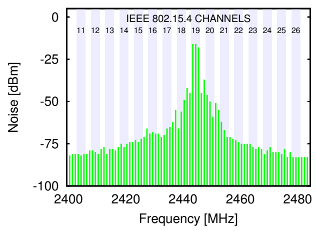
Figure 3: Simplified diagram of the CC2420 AGC loop.

(ADC). More specifically, the signal is maintained within the ADC dynamic range, despite large variations in the input signal from the antenna. For this purpose, the AGC loop uses a digital sample of the final IF signal amplitude and adjusts the gain of the VGA stage accordingly (see Figure 3). If a narrowband signal is present near the cut-off frequency of the combined IF chain, the resulting sampled signal amplitude may be remarkably lower than the partially unfiltered one at the ADC, as a consequence of the digital filtering. Since the AGC uses the final value to set the gain of the amplifier chain, there is no guarantee that the ADC is not saturating. In the event of ADC saturation, the receiver is no longer linear and the RSSI values are incorrect.

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

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

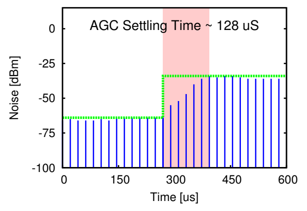
The register also includes flag bits to activate peak detectors among fixed gain stages in the IF chain and at the ADC itself [4].



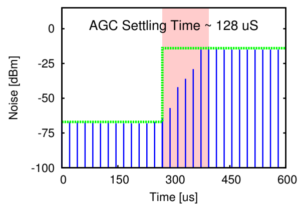
(a) Sensor Mote

(b) Anritsu MS2711D

Figure 4: Single tone excitation obtained running the *frequency scanner* operating across the band (a), and in the Anritsu Spectrum Analyzer, with a frequency span of 2 MHz (b). Notice the correct readings despite the very narrow pulse used, as compared to Figure 2(c).



(a) -25 dBm



(b) 0 dBm

Figure 5: Evolution of RSSI readings over time to different RF tone step signals. The accuracy of our RSSI scanner is high enough to show the moving average used by the CC2420 to compute the RSSI over the last 8 received symbols.

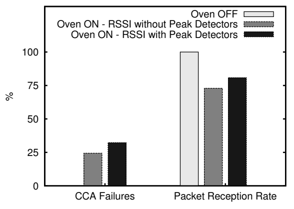
3.3 Validation of the Experimental Setup

We validate our RSSI noise floor measurements both in time and frequency with the help of a professional Anritsu MS2711D spectrum analyzer [10]. In these experiments, we connect the RF ports of the transceivers or the analyzer directly via a 50 Ohms matched impedance RF pigtail. This isolates the signals of interest from external noise sources and eliminates the medium pathloss, so that the amplitude of the tone and the spectral footprint can be compared.

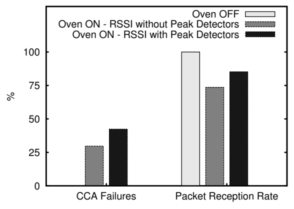
Firstly, we verify the correctness of the *frequency scanner* readings, using the unmodulated test signal available in the CC2420 radio. In order to do this, we program another mote to transmit an unmodulated tone tuned at 2445 MHz, the center of IEEE-802.15.4 channel 19, at maximum power. Figure 4(a) shows the correct operation of the receiver and the linearized IF amplifier chain while scanning the RSSI values across the band using the peak detectors. The same test signal can be seen in the spectrum analyzer (Figure 4(b)). This worst case scenario shows that we have linearized the receiver, thus avoiding wrong RSSI noise floor readings.

Secondly, we measure the evolution of the RSSI readings over time to an RF tone step signal in order to evaluate the accuracy with which we can effectively measure RSSI values. We use our *time scanner* with two different power levels (-25 and 0 dBm), and obtain the results shown in Figure 5. The frequency of the scanner is sufficiently high to show how the CC2420 internally averages the RSSI over the last 8 received symbols, or 128 μs, as defined by the IEEE 802.15.4 standard. Such settling time is shown to be independent of the height of the step signal.

Impact on Clear Channel Assessment (CCA). Activating the peak detectors in-between the amplifier stages also improves the reliability of the CCA operation commonly used in MAC protocols [9]. Due to wrong RSSI readings, the CCA returns a clear channel when a narrow unmodulated signal is transmitted. As a result of this, the application would generate a transmission that is very likely to fail, thus wasting some of the limited energy budget.

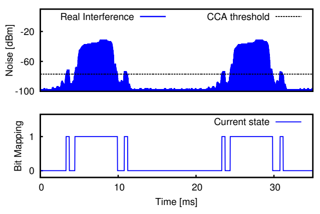


(a) Channel 23

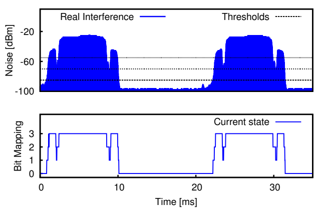


(b) Channel 25

Figure 6: Avoiding wrong RSSI readings improves the CCA accuracy and packet reception rate under interference.



(a) 1-bit precision



(b) 2-bit precision

Figure 7: Encoding techniques to save memory resources.

A typical example of this would happen when transmitting packets in presence of an active Bluetooth device or a microwave oven in the neighborhood. Our approach significantly improves the CCA accuracy, leading to a higher Packet Reception Rate (PRR).

Figure 6 shows the amount of "channel busy" outcomes of CCA before and after activating the peak detectors. The absolute gain in terms of PRR depends on the microwave oven model, on the channel of interest, and on the data rate. We experimentally collect data at the receiver side of a couple of sensor nodes communicating periodically at a rate of 128 packets/second in presence of an active Lunik 200 microwave oven in the neighborhood. The nodes are placed at 1 meter distance and use a transmission power of -25 dBm. As shown in Figure 6, the PRR increases by up to 12% when activating the peak detectors and avoiding wrong RSSI readings.

4. (RE)GENERATING INTERFERENCE

With the techniques to accurately measure interference introduced in the previous section, we can now proceed to record and replay those patterns. We describe first how to compress and store traces on motes and then how to playback those recordings.

4.1 Recording Interference Traces

When used in *regeneration* mode, HandyMote records interference traces that are later played back accordingly. Those traces can be either stored on the mote in RAM or Flash memory, or – if the HandyMote is connected to a testbed during recording – can be streamed over a wired backchannel to a base station. In any case, the data rate of 480 kbps generated by sampling RSSI with a resolution of 8 bits to hold values between 0 and -100 dBm at 60 kHz is too high to store it directly in memory or to stream it over the backchannel. The very efficient Coffee Flash file system supports a peak write bandwidth of only 376 kbps [11], the MSP430 UART supports a maximum data rate of 460 kbps for writing to the USB backchannel, and the limited 4 kB RAM of the MSP430 could just record a trace of less than 70 milliseconds duration.

While we need a high compression ratio, the compression method has to be efficient enough to allow sampling of RSSI at 60 kHz. Therefore, we use a simple Run-Length Encoding strategy and a quantization of the samples to a few bits per sample. We store a stream of pairs (v, o), where v is a sample and o is the number of consecutive occurrences of this sample. This method is very effective, as RSSI values typically change slowly over time. The quantization is justified by the fact that the CC2420 only supports 11 distinct output power levels in the range [-55,0] dBm by setting the PA_POWER register to the values we derived and listed in Table 4.1. To obtain the highest possible output resolution, four bits per sample with an appropriate non-linear quantization are hence sufficient. For example, for two-bit resolution one can use thresholds -55, -70, and -80 dBm (or register values 31, 7, and 3) with a spacing of 15 and 10 dBm, respectively, for quantizing the RSSI range into four regions.

Figure 7(b) shows how original RSSI readings (top) are mapped into 2 bits (bottom): the two-bit quantization of a 35 ms interference recording reduces the amount of data from 2076 Bytes to 84 bytes – a compression ratio of $\frac{1}{25}$. A single bit per sample is enough to support binary interference regeneration. This corresponds to the outcome of a continuous CCA operation, in which the outcome busy/idle channel is mapped to a binary number [12]. Figure 7(a) shows the outcome of a one-bit quantization of 35 ms of interference. The amount of data is reduced from 2076 Bytes to 20 Bytes – a compression ratio of less than $\frac{1}{100}$. This reduces the raw data rate of 480 kbps to less than 5 kbps (depending of course on the values of the raw samples), a data rate that can be handled by Flash and USB, and allowing us to store several seconds of recording in RAM. In our current implementation, we store traces in RAM.

Recording interference traces is energy demanding, as both CPU and radio need to be constantly active while scanning the radio medium. Using software-based on-line energy estimation [13], we obtain an average power consumption of 65.4 mW for Tmote Sky motes, which allows for a lifetime up to 4 days when powered using primary AA batteries.

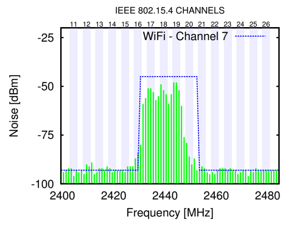
PA_POW.	dBm	PA_POW.	dBm	PA_POW.	dBm
31	0	15	-7	2	-45
27	-1	11	-10	1	-50
23	-3	7	-15	0	-55
19	-5	3	-25	-	-

Table 1: Discrete output power levels of the CC2420 radio.

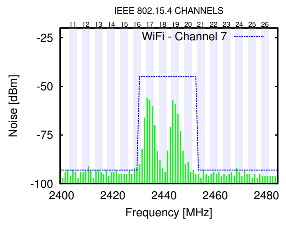
4.2 Generating Interference Patterns

We have recently shown how the CC2420 test modes can be used to generate controllable and repeatable interference [14, 15] by transmitting a modulated or unmodulated carrier signal that is stable over time. This approach is superior to common jamming techniques based on packet transmissions, as the emitted carrier signal is independent from packet sizes and inter-packet times.

In order to generate an interference pattern, the interferer has to be enabled and disabled and its output power has to be set according to the compressed recorded trace in regeneration mode or according to the output of models in emulation mode, as described in Section 5. When enabling the transmitter using the STXON command, the radio oscillator first has to stabilize before a transmission is possible, resulting in a latency of 192μs or a maximum playback frequency of only 5 kHz. Therefore, we leave the transmitter on and just change the output power level to 0 (or -55 dBm) instead of disabling the transmitter. At level 0 the RF output power is so small that even a receiver at a distance of only few centimeters can hardly notice the signal. The advantage of this approach is that the latency for changing the output power is dominated by the SPI access time. We optimized the SPI driver in Contiki, resulting in a latency of only few microseconds – allowing us to to playback at the same frequency of 60 kHz that was also used during recording.



(a) HandyMote (ch. 17,18,19)



(b) HandyMote (ch. 17,19 only)

Figure 8: Compared to wideband interferers, the HandyMotes jam only selected channels, preserving connectivity in others.

Besides the sampling and playback rate, also the jitter during playback of the individual samples needs to be minimized in order to ensure an accurate reconstruction. At 60 kHz, the playback time between two consecutive samples is just 17μs, hence the duration of the execution of a sequence of microcontroller instructions is no longer negligible. In particular, different execution paths in the program to uncompress samples in regeneration mode lead to different execution times and jitter. Therefore, we add NOP instructions to make all execution paths equally long.

4.3 Multiple Channels

Sensor motes are designed to transmit in only one of the 16 IEEE 802.15.4 channels. As most of the existing sensornet protocols only use a single channel, a single HandyMote is sufficient to interfere with this channel. However, there is an increasing trend to use multiple channels in order to increase robustness and bandwidth. In this case, we use multiple HandyMotes, each one interfering with one channel, as depicted in Figure 8(a). Using this approach, the interfered channels can be carefully selected as shown in Figure 8(b), and one can therefore avoid to jam other sensor networks installed in the same building. In contrast, a wideband interference source such as WiFi always jams at least 4 adjacent 802.15.4 channels.

For the synchronization we assume that all HandyMotes are connected to a basestation computer via USB cables and hubs as it is common in existing testbeds, such as TWIST and MoteLab. The synchronization algorithm is inspired by [16] and works as follows. For every HandyMote, the basestation sends a stream of N packets to the HandyMote. Just before sending the packet, the basestation reads its clock and includes the send timestamp in the message. Upon reception, the HandyMote reads its local clock to obtain the receive timestamp. It then computes the difference between send and receive timestamps for each of the N messages, chooses the message with the smallest difference (this is the message that had the smallest latency), and corrects its local clock by this difference. The synchronization error then depends on the variability of the observed minimum latencies across different HandyMotes. As the minimum latency is rather stable across different HandyMotes, the HandyMotes will be synchronized accurately among each other (but not necessarily with the basestation).

To estimate the accuracy of synchronization, we measure the clock offset between two HandyMotes synchronized in that way by having them trigger one of their digital output pins when their synchronized clock reaches a given time. By connecting the output pins of the two HandyMotes to an oscilloscope we measure an average offset of 8.44μs with a standard devitation of 6.94μs for a stream of N=10 packets. Increasing N further does not substantially reduce error. This accuracy is enough to ensure synchronization of motes playing back interference at 60 kHz.

5. MODELING INTERFERENCE SOURCES

In this section, we describe how we can use an HandyMote to emulate three major sources of external interference on the 2.4 GHz ISM band: WiFi and Bluetooth devices, as well as microwave ovens. We present models that capture the temporal characteristics of these interference sources. A key requirement is the simplicity and efficiency of models, as they need to be executed in real-time on the HandyMotes to generate interference. We are not concerned about the intensity of the generated interference, since when running a HandyMote in *Emulation Mode*, we can decide to adjust the output power of the CC2420 according to different schemes. For example, the output power can be kept fixed or chosen randomly, as shown in Figure 9 (emulation of the interference generated by a Whirlpool M440 microwave oven).

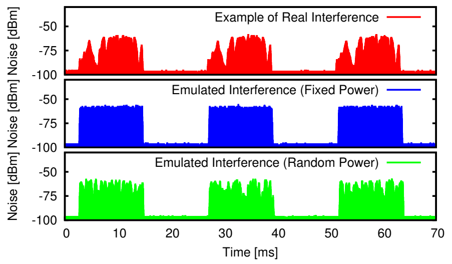


Figure 9: Emulation of microwave oven interference (top) with fixed (middle) and random power (bottom).

5.1 WiFi Emulation

Modeling Wi-Fi traffic is challenging, as it depends on several factors such as the number of active users, their activities, the protocols they use (UDP or TCP), the traffic conditions in the backbone, etc. Under some reasonable assumptions, several theoretical studies have analyzed the performance of 802.11 [17, 18, 19, 20]. However, based on the empirical data we collected, we observed that the models for saturated sources provide a better approximation than the models for non-saturated sources (saturated sources always have data to send). Hence, in order to re-create a realistic representation of interference patterns, we implement an analytical model for saturated traffic sources, and for non-saturated traffic we derive models from empirical data.

Non-Saturated Traffic: Empirical Model. The empirical model for non-saturated traffic is obtained in the following way. Let us denote a random variable X as the *clear* time between two consecutive *busy* times. We obtain the probability mass function $p(x) = P_r\{X = x\}$ from the empirical sampling of the channel, where x is the time in number of slots (each slot is 1 ms). The length of the *busy* times is represented by the transmission delay of packets, which is a rather deterministic variable (for a fixed packet size). Following the methodology described on the previous paragraph, we obtained the $p(x)$ for the scenarios presented on Table 2. Figures 10(a) and 10(b) show the probability mass function $p(x)$ for two WiFi scenarios: an audio-stream application and the download of a large file.

Scenario	Users	Scenario	Users
Radio Str.	1	Video Str.	1
File Transfer	1	File + Radio	1

Table 2: Scenarios.

Saturated Traffic: Analytical Model. There exist several analytical models for the Distributed Coordination Function (DCF) mode of 802.11. Among them, the model proposed by Bianchi [18] has been one the most influential. Bianchi modeled the DCF mode of 802.11 as a discrete Markov process, where the back-off and retransmission mechanisms are represented as discrete states. Based on this model, Garetto and Chiasserini [19] developed a simpler Markov process by merging back-off states. For details, we re-

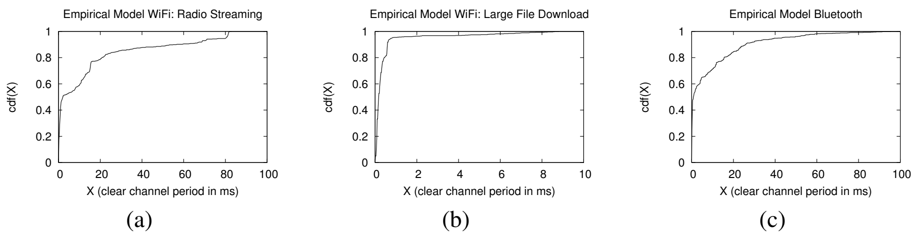


Figure 10: Empirical Models for WiFi and Bluetooth.

fer the reader to their paper [19]. In our work, we use Garetto and Chiasserini's model to emulate WiFi interference for saturated sources: whenever there are transmissions of frames in the model, the HandyMote activates the carrier.

5.2 Bluetooth Emulation

IEEE 802.15.1, better known as Bluetooth, specifies 79 channels, spaced 1 MHz, in the unlicensed 2.4 GHz ISM band. Bluetooth stack implementations apply an Adaptive Frequency Hopping (AFH) mechanism to combat interference, which does not permit to anticipate the frequency at which the interference will take place. Bluetooth hops 1600 times/sec., which means that it remains in a channel for at most 625μs. Note that Bluetooth channels are 1 MHz-spaced, while the resolution of our scanner is 2 MHz, which implies that consecutive time slots may eventually coincide within this frequency window and result in a larger interference period. We model Bluetooth using the same method as for non-saturated traffic in WiFi, that is, we obtain the probability density function $p(x)$ for the *clear* periods of the channel, and the transmission time of Bluetooth packets for the *busy* periods. Figure 10(c) shows the probability mass function $p(x)$ for the Bluetooth scenario. The Adaptive Frequency Hopping characteristic of Bluetooth leads to a smoother *cdf* curve compared to WiFi, because the *clear* periods are independent of the application run.

5.3 Microwave Oven Emulation

Microwave ovens are a kitchen appliance used to cook or warm food by passing non-ionizing microwave radiations to heat water and other polarized molecules within the food, usually at a frequency of 2.45 GHz. Therefore, they are a potential source of interference for sensornets operating in the 2.4 GHz spectrum.

The detailed characteristics of the interference patterns emitted by domestic microwave ovens depend on the model; nevertheless they all present the same basic properties. Firstly, on a spectral basis, our experiments show that microwave ovens tend to interfere all the 802.15.4 channels, with a higher impact on channels 20-26. It is not possible to state with certainty which channel will be mostly affected, as our experiments confirm that the peak frequency of the ovens depends on multiple factors, including the oven content, the amount of water in the food, and the position within the oven, as all these parameters affect the temperature of the magnetron [21]. Secondly, on a temporal basis, the generated noise is rigorously periodic. Figure 11 shows the temporal pattern of the interference caused by a Lunik 200 microwave oven retrieved experimentally. In one period of approximately 20 ms, there is an 'on' and 'off' cycle, whose duration is roughly 10 ms each. This matches the observations in [22], confirming the correctness of our results.

For all the above reasons, microwave oven interference is the simplest dynamic to model, as it follows a deterministic on/off sequence. Defining the period of the signal τ, the duty cycle λ (fraction of time the oven is 'on'), and hardcoding these two parameters into the HandyMote, we can generate interference patterns such as the ones shown in Figure 9.

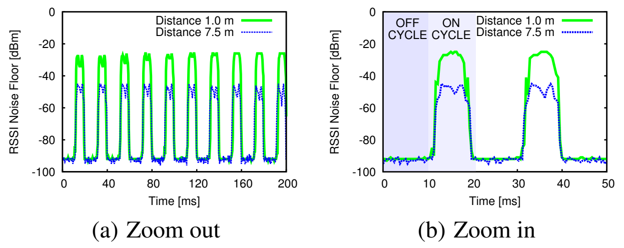


Figure 11: Temporal characteristics of the interference caused by microwave ovens. The ovens emit frequencies with a periodic pattern with period $T \approx 20$ ms.

6. TESTBED CONFIGURATION

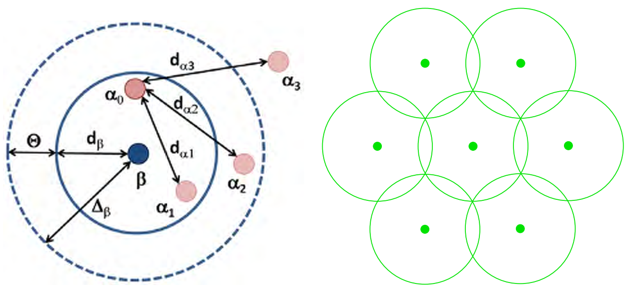
As outlined in Section 2, we partition the area of a testbed into different cells to deal with the limited RF output power of the HandyMotes compared to interference sources such as WiFi or microwave ovens. In this section we explain how to configure the testbed, i.e., how to place the HandyMotes and how to control their RF output power level such that the motes in the testbed are partitioned. This implies that every mote should be covered by a cell and cross-talk between neighboring cells should be minimized (i.e., a HandyMote does not interfere with motes outside of its cell).

6.1 Coverage and Cross-Talk

A key issue we need to understand is under which conditions the packet reception of a mote is actually affected by an interference signal generated by a HandyMote. The impact of interference on reception in the CC2420 radio is closely dependent on the modulation scheme used, namely OQPSK (Offset Quadrature Phase Shift Keying) and DSSS (Direct Sequence Spread Spectrum). With these modulation schemes, the interference signals generated by two HandyMotes do not simply "add up" at the receiver as it would be the case for ASK (Amplitude Shift Keying) used in older radios, but the receiver will pick the stronger of the two signals if their strength differs by a certain minimum. This is called co-channel rejection: according to [4], the CC2420 is able to receive a signal at -82 dBm if the second signal is at least 3 dB weaker.

In order to enable a HandyMote to interfere with the motes in its cell, we therefore need to make sure that a mote belonging to the cell will receive interference signals from that HandyMote with a signal strength at least 3 dB higher than the maximum strength of other signals that mote may receive. To minimize cross-talk between neighboring cells, we need to make sure that motes outside of the cell will receive that interference signal with a signal strength that is at least 3 dB weaker than the minimum strength of other signals that this mote may receive. Finally, we need to make sure that all testbed motes are covered by the cells.

In practice, an ideal configuration without cross-talk and with complete coverage typically does not exist. Also, due to environmental dynamics, the amount of cross-talk and coverage may vary over time. We can only try to find a configuration that maximizes coverage and minimizes cross-talk. Note that there is a tradeoff



(a) (b)

Figure 12: JamLab's division in cells.

between the size of the cells and the accuracy of the spatial distribution of generated interference: the smaller the cells, the higher is the spatial sampling resolution and the smaller are the cross-talk regions. However, smaller cells also implies that more HandyMotes are needed to cover the testbed.

6.2 A Theoretical Model

In this section we develop a theoretical model that allows us to estimate the radius of a cell such that a HandyMote can still interfere with all nodes in the cell. We will also model the amount of cross-talk between neighboring cells. Finally, we develop a model that allows us to estimate how many HandyMotes are at least needed to cover a testbed deployed over a geographical area A.

In order to derive the models, we need to make a number of practical assumptions. Firstly, we assume that the minimum distance between a pair of motes in the testbed equals D_{min} with typical values in the order of few meters. For example, it is common practice to place a mote in each room on an office floor. Secondly, we assume that we can reduce the RF output power level of the testbed motes to a value P_{mote} below the maximum of 0 dBm (e.g., -10 dBm) without loosing connectivity. In practice, this is often done to obtain multi-hop topologies with a large diameter even on the constrained space of an office floor. Thirdly, we assume that a mote is only able to receive a packet with a certain minimal signal strength P_{min}, with typical values in the order of -90 dBm. Finally, we assume the signal propagation can be modeled with the widely used log-normal model [23, 24, 25]:

$$P(d) = P_T - P_L(d_0) - 10 \cdot \eta \cdot \log_{10} \frac{d}{d_0} + \chi_\sigma \tag{1}$$

where $P_L(d_0)$ is the path loss measured at reference distance d_0, η is the path loss exponent, χ_σ is a zero-mean Gaussian random variable with standard deviation σ that models the random variation of the RSSI value due to shadowing. We use the well-known $P_L(2) = 46$ dBm, and the typical path loss exponent for indoor environments $\eta = 6$ without accounting for shadowing.

Consider the scenario in Figure 12(a) with a HandyMote β and several motes α_i. We are interested in computing the cell radius d_β such that HandyMote β can block the reception of any message by motes contained in its cell (i.e., α_0 and α_1 in the figure). Further, we are interested in the radius Δ_β of the cross-talk region. The cross-talk region is defined as the region where the reception of a message by a mote (i.e., α_2 in the figure) may but need not be blocked by HandyMote β.

Knowing output power P_{handy} of the HandyMote and P_{mote} of the mote, as well as the minimum distance D_{min} between motes, we can compute the maximum RSSI P_{max} a mote can receive from another mote:

$$P_{max} = P_{mote} - P_L(d_0) - 10 \cdot \eta \cdot \log_{10} \frac{D_{min}}{d_0} \tag{2}$$

Using that value and the output power P_{handy} of the HandyMote, we can compute the radius of the cell d_β as follows:

$$d_\beta = 10^{\frac{-P_{max}+P_{handy}-P_L(d_0)+10\cdot\eta\cdot\log_{10}(d_0)}{10\cdot\eta}} \tag{3}$$

Knowing the minimum RSSI P_{min} at which a mote can still receive a message, we can compute the radius of the cross-talk region Δ_β as follows:

$$\Delta_\beta = 10^{\frac{-P_{min}+P_{handy}-P_L(d_0)+10\cdot\eta\cdot\log_{10}(d_0)}{10\cdot\eta}} \tag{4}$$

From that we can compute the difference between the cell radius and the radius of the cross-talk region as $\Theta = d_\beta - \Delta_\beta$.

Knowing the cell radius d_β, we now derive a simple model to estimate the number of HandyMotes needed to cover a given area A. As illustrated in Figure 12(b), we consider the sparsest-possible coverage of an area with disks. Ignoring border effects, the area covered by a single cell can be estimated with the area of the hexagon defined by the intersection points of one circle with the six adjacent circles. Dividing A by the area of the hexagon, we can estimate the number of HandyMotes N needed to cover area A:

$$\mathbf{N} = \frac{A}{\frac{3*\sqrt{3}}{2} * d_\beta^2} \tag{5}$$

We now illustrate those model with concrete examples. If we have a sparse testbed with a distance between nodes of $D_{min} = 4$ meters and transmission powers $P_{mote} = -15$ dBm, $P_{handy} = 0$ dBm, we derive $P_{max} \approx -80$ dBm and the radius of our cells $d_\beta = 8$ meters. This configuration would imply that the size of the cross-talk area $\Theta \approx 4$ meters when using $P_{min} = -90$ dBm.

This cell size is obviously very large, and the consequence would be that in theory only $N = 6$ HandyMotes would be needed to cover a testbed area $A = 750m^2$.

However, with this configuration the cross-talk area Θ is quite large. The accuracy of the regenerated interference may therefore be low as all nodes contained in cross-talk areas are potentially interfered by multiple HandyMotes in neighboring cells with different interference traces. To gain more accuracy, we need to decrease the size of the cross-talk area Θ. This can be achieved by reducing the radius of the cells by means of reducing P_{handy}, which requires more cells and HandyMotes to cover the testbed area. To obtain $\Theta \approx 2$ meters, using the same parameters as above, one would need to use a cell radius of $d_\beta \approx 4$ meters, which would imply that to cover the same testbed area $A = 750m^2$, we would need at least $N = 19$ HandyMotes.

6.3 Practical Testbed Configuration

In this section we describe in a practical procedure how to configure the testbed, i.e., how to select the HandyMotes and how to set their power levels such that mote coverage is maximized and cross-talk between cells is minimized.

1. In a first step, we empirically obtain D_{min}, P_{min}, and P_{max} from the testbed as introduced in the previous section. D_{min} can be obtained directly from the layout of the testbed. P_{min} and P_{max} are measured by having the motes in the testbed sequentially broadcast a message and all others nodes record the maximum and minimum RSSI value.
2. Knowing these parameters, we can compute the maximum cell radius according to Equation 3. We overlay a hexagonal grid as depicted in Fig. 12(b) with cells of the computed radius over the testbed layout, place HandyMotes at the center of the overlay cells (or select testbed motes close to the center of the cells to become HandyMotes), and allocate motes to the HandyMotes based on the cell overlay.
3. Next, we sequentially trigger the selected HandyMotes to generate an interference signal at maximum output power

and check if every mote in the cell of a HandyMote is covered. For this, the motes measure the RSSI noise floor and check if it is larger than $(P_{max} + 3)$ dB.

4. If there are any uncovered motes, we select additional HandyMotes in the vicinity of those motes and return to step 3.
5. In order to reduce cell cross-talk, we reduce the output power levels of the HandyMotes to the minimum value that still guarantees coverage using the same approach as in step 3. If the selected power level is not the maximum power levels, then the power levels higher than the selected one can be used to realized different levels of interference strength. Otherwise, only binary interference can be generated.
6. Finally, one may estimate the quality of the generated configuration by counting the number of motes contained in cross-talk region as follows. The HandyMotes sequentially generate an interference signal at the selected output power. All motes outside of the cell of that HandyMote measure RSSI. If the measured value is larger than $P_{min}+$ 3 dB, then the mote is contained in a cross-talk region. If the number of motes in cross-talk regions is too high, one may start over with a different initial selection of cells.

This procedure is supported by a program running on the testbed motes during the setup phase. After the configuration is completed, the motes may be programmed with the test application. As part of future work, we plan to further automate this procedure.

7. EVALUATION

In this section, we first evaluate the accuracy with which a HandyMote can regenerate a previously recorded interference trace in the time domain. We then augment an existing sensornet testbed infrastructure with JamLab, and evaluate the accuracy with which the augmented testbed can regenerate a previously recorded interference trace in the spatial domain. Finally, we use JamLab to characterize the performance of MAC protocols under interference.

7.1 Temporal Accuracy

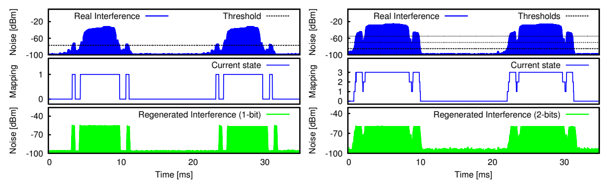
We evaluate the accuracy with which a HandyMote can regenerate a previously recorded interference in the time domain. We run a HandyMote in regeneration mode in proximity of an active Lunik 200 microwave oven warming a bowl of tea. The HandyMote is placed at 1 meter distance from the oven, and records a trace of channel 24 at a sampling rate of 60 kHz. Figure 13(a) (top) shows the interference generated by the microwave as measured by the HandyMote. Next, the trace is quantized to single-bit resolution (middle). Finally, once the microwave oven stopped operating, the HandyMote plays back the recorded binary interference (bottom) using transmission power 0 dBm. As we can notice from the figure, the regeneration is quite accurate in the time domain.

We quantify the accuracy of the regenerated signal with respect to the originally recorded signal using the the *cross-correlation* coefficient (**c**). We represent original and regenerated signals by the series $x(i)$ and $y(i)$, respectively, where $i = 1, \ldots, N$. These series are binary, and take 0 (clear channel) or 1 (busy channel) values. Considering this representation, **c** is given by:

$$\mathbf{c} = \frac{\sum_{i=-\infty}^{\infty} x(i)y(k-i)}{rms(x)rms(y)} \tag{6}$$

where $rms()$ denotes the root mean square value of a signal. We tested eight pairs of original and regenerated samples and the maximum value of **c** was selected for each pair:

$$\mathbf{c}_{xy} = \max_{k\in[-(N-1),(N-1)]} \{\mathbf{c}\} \tag{7}$$



(a) 1-bit Mapping (b) 2-bit Mapping

Figure 13: Regenerated interference of a microwave oven.

The average correlation $\mathbf{c}_{xy}$ is 0.93 with a standard deviation of 0.065. Hence, our implementation does a commendable job with respect to the cancellation of the jitter between sampled and regenerated interference and hence regenerates interference with a fairly high accuracy.

We carry out the same experiment using 2-bit quantization with thresholds -55, -70, and -80 dBm, and we then regenerate the interference using transmission power register levels 31, 7, and 3 (i.e., 0, -10, -25 dBm), respectively. The results match the above ones with binary interference. Figure 13(b) shows the regeneration process when using a two-bit quantization.

7.2 Impact on Packet Reception Rate

In this section we experimentally study the impact of interference on Packet Reception Rate (PRR), comparing the PRR for original, emulated, and regenerated interference signals. We use the same Lunik 200 microwave oven as in the previous experiment, and collect data at the receiver side of a pair of sensor nodes at about 1 meter distance, with the sender transmitting packets at a rate of 128 packets/sec. The sensor just transmits the packet without any clear channel assessments or duty cycling. We place an HandyMote between the two nodes and we run it both in emulation and regeneration mode, once the microwave oven stopped being active.

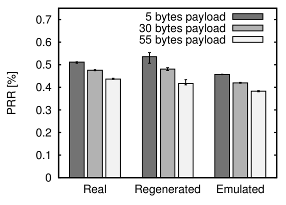
We carry out different experiments with different payload sizes, and we run the HandyMote using transmission power 0 dBm in both emulation and regeneration mode, such that the generated interference signal blocks communication between the sensor nodes.

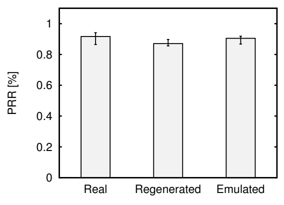
Figure 14(a) shows the results. The PRR collected when the microwave oven is active decreases when the payload size increases as the probability of periodic microwave interference hitting a packet increases with increasing payload size. The PRR obtained for regenerated interference differs by 5.6% from the original one, hence showing a reasonable accuracy. For emulated interference, the PRR differs from the original one by 12.83%, the reason for that being the noisy amplitude of the original interference signal as depicted in Figure 9, such that occasionally the interference is too weak to block the transmission. In contrast, the emulated interference signal is binary and always blocks communication. Accuracy could be improved in this case by randomly varying the transmission power of the HandyMote as discussed in Section 5.3.

We repeat the experiments in presence of Bluetooth interference. We first measure PRR during a Bluetooth file transfer between a laptop and a mobile phone. We place the HandyMote between the 2 communicating motes and we measure the PRR obtained with original, emulated, and regenerated interference. We run the HandyMote in emulation mode using the models derived in Section 5.2.

Figure 14(b) shows that the packet reception rate obtained under regenerated interference differs by 5.02% from the the original one, while in emulation mode it differs by only 1.31%.

Finally, we repeat the experiment with WiFi interference. Using the same setup as above, we run the HandyMote in emulation mode using the models derived in Section 5.1 while generating WiFi traffic from a laptop according to the scenarios presented on Table 2.


(a) Microwave Oven


(b) Bluetooth

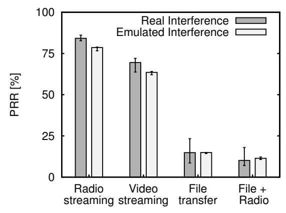

(c) WiFi

Figure 14: Impact of real, emulated, and regenerated interference on packet reception rate.

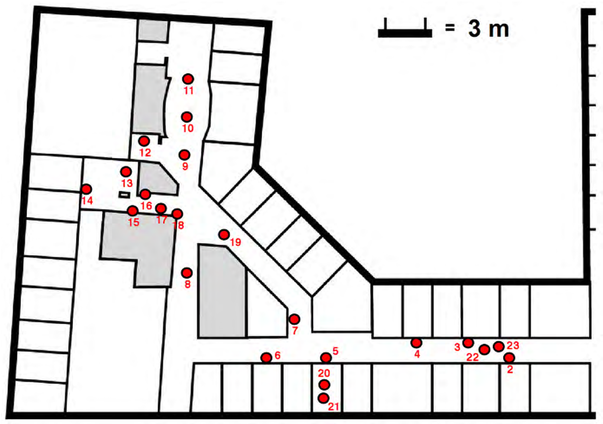


Figure 15: Map of the testbed used for our experiments.

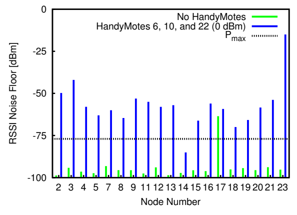
Figure 14(c) shows the results. Also in this case the HandyMote generates interference quite accurately, and the difference between the PRR obtained under real interference and the one obtained under emulation varies between 0.25% and 8.56%. The reason for this difference is that emulation repeats the same pattern over and over, while actual WiFi interference might change in time, due, for example, to TCP adaptation mechanisms.

7.3 Testbed Configuration

Next we want to study the accuracy of the spatial distribution of interference generated by a testbed that has been augmented with JamLab, and therefore configured as described in Section 6.3. Figure 15 shows the topology of the testbed we use, which contains 25 Tmote Sky nodes deployed in an office environment.

As discussed in Section 6.2, there is a tradeoff between the accuracy of regeneration and the cell size, as larger cell size leads to larger cross-talk regions, where motes may be interfered by multiple HandyMotes in neighboring cells. We therefore want to investigate a worst-case scenario with respect to accuracy, where only a few large cells with large cross-talk regions are used. Contrary to the procedure outlined in Section 6.3 to compute the cell size, we therefore start with a cell radius of $d_\beta = 8$ meters, which equals the values in the first example in Section 6.3, where $P_{max} = -80$ dBm. With this cell size, we can cover the testbed with just three cells. We select nodes 6, 10, and 22 as HandyMotes. Next we sequentially trigger the selected HandyMotes to generate interference at maximum output power, and check that the RSSI at every mote is at least $P_{max} + 3dB = -77$ dBm.

Figure 16(a) shows that, with this configuration, node 14 would not be covered as RSSI is smaller than -77 dBm due to the remote location of the node. We therefore change the selection of the HandyMotes (instead of adding more cells) to motes 6, 9, and 23 as shown in Figure 1). With this configuration node 14 is covered, but node 10 is not covered by HandyMote 9 (Figure 16(b)), while in the original configuration node 10 covered node 9 (Figure 16(a)).


(a) HandyMotes 6, 10, and 22

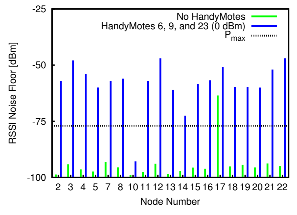

(b) HandyMotes 6, 9, and 23

Figure 16: Testbed augmented with JamLab. In the first configuration, nodes 6, 10, and 22 are selected as HandyMotes. In the second configuration, nodes 6, 9, and 23 are selected instead.

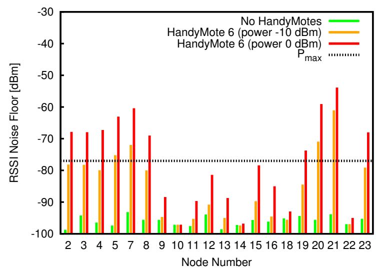


Figure 17: Impact of TX power of HandyMote 6.

This is an example of an asymmetric link – something our simple model in Section 6.2 does not capture. Figure 16 shows also another practical problem. Node 17 is apparently broken as it always returns RSSI readings higher than −67 dBm. We therefore ignore this node in the remainder of the experiments.

Finally, we need to reduce the output power of the HandyMotes to minimize the cross-talk area while still maintaining coverage. The cell controlled by HandyMote 6 is quite small and therefore it is possible to reduce its output power. We show the outcome of varying the transmission power of HandyMote 6 in Figure 17: power level 11 (-10 dBm) is the smallest that provides full cell coverage. Similarly, we obtain output power levels 31 and 7 for HandyMotes 9 and 23, respectively. Figure 18 shows that with such configuration, only the HandyMote controlling a cell can generate an interfering signal at the other motes in the cell exceeding P_{max}.

7.4 Spatial Accuracy

Using the testbed configuration obtained in the previous section, we now study how accurately we can regenerate the spatial distribution of interference. For this, we place a Whirlpool M440 microwave oven in the position marked as M in Figure 1, within the cell controlled by HandyMote 6. This case represents a worst-case scenario, as the oven can interfere over long distances due to its high (60 dBm) and highly varying output power.

Our goal is to record the spatial distribution of the interference patterns generated by the microwave oven in one of the most affected channels (23) all over the testbed. We then let the HandyMotes regenerate the recorded traces while the remaining nodes

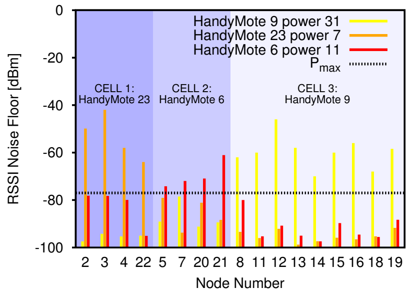
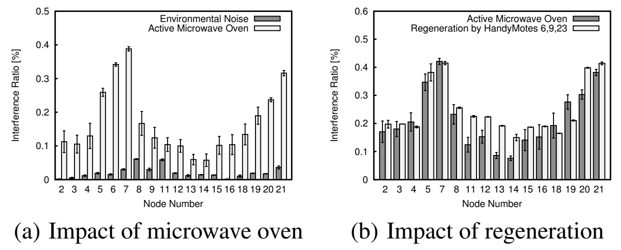


Figure 18: JamLab configuration with independent cells.



(a) Impact of microwave oven (b) Impact of regeneration

Figure 19: Comparison between the interference generated by an active microwave oven and the one regenerated by JamLab in regeneration mode throughout the whole testbed.

record the regenerated interference and compare it with the original interference recorded while the microwave oven was active.

As we have already investigated the temporal accuracy of regeneration in Section 7.1, we now focus on the distribution of the intensity of interference. Instead of recording raw traces, every mote computes the *interference ratio* as the percentage of time in which interference is present (i.e., the percentage of RSSI noise floor readings higher than P_{max}). Figures 19(a) and 19(b) show the comparison of the interference ratio during the activity of the microwave oven, and during the regeneration using JamLab (HandyMotes 6, 9, and 23). Due to their different distances from the microwave oven, node 7 recorded the highest interference ratio when the oven was active, followed by nodes 6, 21, 20, and 5, respectively (Figure 19(a)). The regeneration within this cell is based on the trace recorded by HandyMote 6, therefore nodes 21, 20, and 5 will perceive a higher interference ratio, node 7 a lower one (Figure 19(b)). A similar reasoning can be applied to all other nodes in the testbed: node 14, for example, perceives a higher interference ratio, as recorded by HandyMote 9, which is closer to the oven. If a better spatial accuracy is required, a higher number of (smaller) cells needs to be configured, as discussed in Section 6.

It is important to remark that the environmental noise may play an important role in the quality of the (re)generation, as it will add-up to the interference (re)generated by the HandyMotes. Observing Figures 19(a) and 19(b), we can see how the interference received by node 8 is higher than the one recorded by HandyMote 9 due to a high environmental noise. In order to reduce the non-determinism caused by differences in ambient interference between recording and regeneration, the experiments should be run when the background noise is low, for example in the evening or during the night.

We finally investigate the accuracy of the regeneration with respect to PRR. We repeat the above experiment while nodes pairs (2,3), (5,21), and (18,19) transmit and receive packets with a payload of 5 bytes at a rate of 64 packets/second on channel 23. Figure 20 shows that PRR values are similar between original and regenerated interference for the first two pairs of nodes, while there is a larger error (31.6%) for pair (18,19). This is due to nodes 18 and 19 being much closer to the microwave oven than HandyMote 9, following the discussion made for Figure 19.

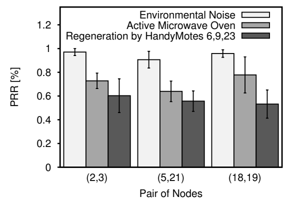


Figure 20: Comparison of the PRR obtained generating interference using a microwave oven and using JamLab.

7.5 Characterization of Protocol Performance

In this section we demonstrate the usability of JamLab by characterising the impact of interference on low-power MAC protocols. We show that using JamLab we can get important insights regarding protocol behaviour under *emulated but realistic* interference.

We perform our experiments in the testbed shown in Figure 15. Our setup consists of a sender (node 7), a receiver (node 5) and one (node 6) or two (nodes 6 and 21) HandyMotes, whose position and transmission power is carefully chosen to jam the communications between sender and receiver. The sender transmits 400 packets to the receiver at a rate of 1 packet/sec. We use 3 different MAC layers: NULLMAC, a simple layer that just forwards packets between the radio driver and the network layer, X-MAC [26], and X-MACQ [27], an enhanced X-MAC with a queue and the ability to rapidly drain the queue in absence of interference. A first HandyMote generates interference using the implementation of Garetto's 802.11 model presented in Section 5.1. We use the model with the RTS/CTS access mechanism and set the minimum and maximum contention window size to 32 and 1032, respectively, as these seem to be the most widely used parameter settings. We emulate saturated traffic from 20 stations (this amount was chosen to have an interference time similar to the one of an active microwave oven), where each station sends packets with a size of 1000 Bytes. A second HandyMote emulates a microwave oven, as in Section 5.1.

Table 3 shows our results. We depict the average results of three runs. With NULLMAC and microwave oven interference, the PRR is slightly lower than 50%, which confirms the results in Figure 14(a). The table also shows that under microwave oven interference, X-MAC performs better than NULLMAC with smaller payloads. As explained in [27], the reason for this is X-MAC sending strobes for a longer time than its off-time and hence the receiver has on average more than one chance to complete the handshake.

While it is known that the PRR decreases with increasing packet size, X-MAC's PRR decreases significantly, namely from almost 60% to less than 40%, as the packet size increases from 30 to 100 Bytes. Also in presence of WiFi interference X-MAC performs much worse for large packet sizes. The experimental results in Figure 14(a) are taken using NULLMAC. Combining the results in this figure with the ones in Table 3, we see a very modest decrease of NULLMAC's PRR with increasing packet size.

The difference between NULLMAC and X-MAC is that in order to receive a data packet, a receiver that employs NULLMAC needs to successfully receive 1 packet only, whereas X-MAC requires the completion of the handshake, i.e., the receiver needs to receive the sender's strobe and acknowledge it, before the sender can send the data packet to the receiver. In our experiments, this data exchange must happen within one time period without interference. This means that until the data packet itself is transmitted, a substantial fraction of a time period without interference has already been used for the handshake. Note that this time period without interference is short due to the bursty interference patterns created by both microwaves and Garetto's WiFi model as the latter emulates saturated traffic. This explains why the packet size is

Payload (Bytes)	Oven NULL	Oven X-MAC	WiFi X-MAC	Both X-MAC	Both X-MACQ
30	45.3%	59.2%	41.6%	20.9%	39.7%
100	43.6%	39.5%	23.8%	9.2%	15.6%

Table 3: Performance of different MAC protocols under emulated but realistic interference (average PRR in %)

more important for X-MAC than for NULLMAC. The table also shows that X-MACQ is more robust than X-MAC against interference, hence confirming the results in [27] using more realistic interference patterns generated using JamLab.

8. RELATED WORK

The study of interference sources in the ISM band has received significant attention from the research community, especially in the crowded 2.4 GHz band. Petrova et al. [1] perform measurements using 802.11g/n devices and quantify their impact on 802.15.4 networks. Sikora reports the impact of microwave ovens, 802.11, and Bluetooth on the packet reception rate of 802.15.4 networks [2]. Liang et al. present a careful analysis of the symmetric and asymmetric IEEE 802.15.4 and 802.11 interference patterns [28]. The high number of interference sources in the ISM band has motivated the study of solutions to overcome interference, in particular WiFi [17, 29]. While we evaluate the same sources of interference, the distinctive and most important contribution of our work is that we provide a low-cost tool to (re)create interference in sensornet testbeds, which goes beyond a one-time-evaluation approach, and enables a better study and debugging of communication protocols.

To the best of our knowledge, we are the first to develop such a low-cost testbed framework for the generation of controlled and realistic interference. Existing sensornet testbeds do not provide any capability for interference generation, or they are limited to static WiFi access points randomly placed in the testbed [3]. JamLab, instead, can seamlessly augment existing sensornet testbeds to study the robustness of protocols against interference.

We have recently discussed the idea of using the CC2420 test modes to generate interference [14, 15] in conjunction with a directional antenna to direct the interference towards a selected set of motes. Our present work goes significantly beyond this, by providing accurate RSSI readings, record-and-playback and emulation of interference capabilities, as well as the integration of these functions into a testbed for interference studies.

Several studies have evaluated the impact of interference on the performance of MAC protocols [27, 30], and a set of fair transmission schedules have been derived by synchronizing the transmission of neighboring nodes in the presence of interference [31]. This type of studies would definitely benefit from the realistic interference patterns that JamLab provides.

9. CONCLUSIONS AND FUTURE WORK

Interference has a strong impact on the performance of sensor networks. Hence, protocols need to be tested under realistic and controlled interference. We present JamLab, a tool to augment existing sensornet testbeds with a low-cost infrastructure for the creation of realistic and repeatable interference patterns. JamLab provides simple models to emulate the interference patterns generated by several devices, and a playback capability to regenerate recorded interference patterns. We demonstrate the utility of JamLab by showing its accuracy in both temporal and spatial domains.

Future work includes a further automation of the testbed configuration procedure, and an accurate study and modeling of new interference sources in the frequency bands used by sensornets.

Acknowledgments

We thank our Shepherd Dr. Yuvraj Agarwal for his constructive comments. This work has been supported by the European Commission with contracts FP7-ICT-224282 (GINSENG) and FP7-2007-2-224053 (CONET), by VINNOVA, by SSF and by the DFG-funded Cluster of Excellence 306/1 ”Inflammation at Interfaces”.

10. REFERENCES

[1] M. Petrova et al. Interference Measurements on Performance Degradation between Colocated IEEE 802.11g/n and IEEE 802.15.4 Networks. In *International Conference on Networking 2007*.

[2] A. Sikora and V.F. Groza. Coexistence of IEEE 802.15.4 with other systems in the 2.4 GHz-ISM-Band. In *IEEE Instrumentation and Measurement Technology*, May 2005.

[3] Divya Sakamuri. NetEye: a Wireless Sensor Network Testbed. Master's thesis, Wayne State University, 2008.

[4] Texas Instruments. *Smart RF CC2420 datasheet - 2.4 GHz IEEE 802.15.4 / ZigBee-Ready RF Transceiver*, March 2007.

[5] A. Dunkels and B. Grönvall and T. Voigt. Contiki - a Lightweight and Flexible OS for Tiny Networked Sensors. In *EmNetS'04*.

[6] The Contiki Projects Community. http://sourceforge.net/projects/contikiprojects.

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

[8] P. Verkaik, Y. Agarwal, R. Gupta, and A. Snoeren. SoftSpeak: Making VoIP Play Fair in Existing 802.11 Deployments. In *NSDI'09*.

[9] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *SenSys'04*.

[10] Anritsu MS2711D Spectr. Analyzer. http://www.anritsu.com.

[11] N. Tsiftes, A. Dunkels, Z. He, and T. Voigt. Enabling Large-Scale Storage in Sensor Networks with the Coffee File System. In *IPSN'09*.

[12] Q. Wang and T. Zhang. Source Traffic Modeling in WSN for Target Tracking. In *Proc. of the 5th ACM PE-WASUN*, 2008.

[13] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *Proc. of EmNets'07*.

[14] C.A. Boano et al. Controllable Radio Interference for Experimental and Testing Purposes in WSN. In *IEEE SenseApp 2009*.

[15] C.A. Boano, K. Römer, Z. He, T. Voigt, M. Zuniga, and A. Willig. Generation of Controllable Radio Interference for Protocol Testing in Wireless Sensor Networks. In *SenSys'09, demo session*.

[16] Flaviu Cristian. Probabilistic clock synchronization. *Distributed Computing*, 3(3):146 – 158, 1989.

[17] R. Musaloiu-E. and A. Terzis. Minimising the effect of WiFi interference in 802.15.4 WSN. *IJSNet'07*, 3(1):43–54.

[18] G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal on selected areas in communications*, 18(3):535–547, 2000.

[19] M. Garetto and C.F. Chiasserini. Performance analysis of 802.11 WLANs under sporadic traffic. In *Networking'05*.

[20] P. Rathod et al. Characterizing the exit process of a non-saturated IEEE 802.11 wireless network. In *MobiHoc'09*.

[21] M. Vollmer. Physics of the microwave oven. In *Physics Education 39/1*, pages 74–81. IOP Publishing Ltd, 2004.

[22] T. Taher, M. Misurac, J. LoCicero, and D. Ucci. Microwave oven signal modelling. In *WCNC'08*.

[23] D. Lymberopoulos, Q. Lindsey, and A. Savvides. An Empirical Characterization of Radio Signal Strength Variability in 3-D IEEE 802.15.4 Networks Using Monopole Antennas. In *EWSN'06*.

[24] M. Zuniga and B. Krishnamachari. Analyzing the Transitional Region in Low-Power Wireless Links. In *SECON'04*.

[25] E. Miluzzo, X. Zheng, K. Fodor, and A. Campbell. Radio characterization of 802.15.4 and its impact on the design of mobile sensor networks. In *Wireless Sensor Networks*, volume 4913, 2008.

[26] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-MAC: a short preamble MAC protocol for duty-cycled WSN. In *SenSys'06*.

[27] C.A. Boano, T. Voigt, N. Tsiftes, L. Mottola, K. Römer, and M. Zuniga. Making Sensornet MAC Protocols Robust Against Interference. In *EWSN'10*.

[28] C. Liang, N. Priyantha, J. Liu, and A. Terzis. Surviving Wi-Fi Interference in Low Power ZigBee Networks. In *SenSys'10*.

[29] J. Hauer, V. Handziski, and A. Wolisz. Experimental Study of the Impact of Wlan Interference on IEEE 802.15.4 BAN. In *EWSN'09*.

[30] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.M. Liang, and A. Terzis. Design and Evaluation of a Versatile and Efficient Receiver-Initiated Link Layer for Low-Power Wireless. In *SenSys'10*.

[31] Y. Yi, G. de Veciana, and S. Shakkottai. On optimal MAC scheduling with physical interference. In *INFOCOM'07*.