Ad Hoc 802.11-802.15.4 Crosstalk-Based Communication in Practice

Inga Rüb
Faculty of Mathematics, Informatics, and Mechanics
University of Warsaw
inga.rub@mimuw.edu.pl

Szymon Acedański
Faculty of Mathematics, Informatics, and Mechanics
University of Warsaw
accek@mimuw.edu.pl

Konrad Iwanicki
Faculty of Mathematics, Informatics, and Mechanics
University of Warsaw
iwanicki@mimuw.edu.pl

Abstract

802.11-802.15.4 crosstalk-based communication (CTC) has been heralded as a potential solution to many interoperability problems posed by low-power wireless networking. Among the issues that can be addressed by CTC, the one that has drawn our attention is configuring 802.15.4-based micro-devices (e.g., low-power wireless sensors) in a convenient way. Such devices have become an intrinsic part of our lives. Yet, they often lack any interfaces suitable for users with no technical background. Consequently, the idea of utilizing an ordinary smartphone as a remote controller for wireless and user-interface-less micro-devices is appealing in particular.

However, despite the promising results on 802.11-to-802.15.4 crosstalk, some important elements are missing to realize this vision. Above all, state-of-the-art CTC solutions require access to low-level subsystems (e.g., firmware) of both sides of communication. In contrast, in real-world applications the communicating devices need not always be fully controllable. On the other hand, the considered sensor configuration use-case allows for ignoring some of the difficulties posed by implementing general-purpose CTC. This propels us to attempt to verify whether a less restrictive version of CTC can be deployed in an ad-hoc manner for micro-device configuration. We answer this question by presenting results of experiments carried out in variety of settings.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*wireless communication*

General Terms

Experimentation, Measurement

Keywords

crosstalk-based communication, non-overlapping channels, configuration of devices, WiFi, ZigBee

International Conference on Embedded Wireless Systems and Networks (EWSN) 2018
14–16 February, Madrid, Spain

ISBN: 978-0-9949886-2-1

1 Introduction

A major problem in the commercial deployments of 802.15.4 wireless low-power sensors in which we have participated was a high cost of configuring the sensors on-site, in particular, selecting their radio channels in the target environments or uploading customers' cryptographic keys. This is because the sensors usually lack any user interfaces and thus have to be configured by external devices. However, such auxiliary devices are not commodity products, and hence typically they are not in possession of the customers. Likewise, using them requires technical background, which the customers normally need not have. As a result, any maintenance work may involve staff of the sensor provider, often visiting the customer on-site. This is expensive and problematic for the customers, especially if they are forced to reveal sensitive data. Therefore, enabling the customers to perform sensor configuration themselves using commodity off-the-shelf devices would be a far superior solution.

The problem is that 802.15.4 is not supported by the popular off-the-shelf products, notably smartphones and tablets. Instead, these devices provide 802.11, which is incompatible with 802.15.4, despite using similar frequency bands. Recently, however, it has been shown that interference—so-called *crosstalk*—between devices based on different radio technologies can be utilized in a constructive manner, to communicate data [1, 13, 11, 4], a technique dubbed *crosstalk-based communication* (CTC). Nevertheless, despite promising results on CTC, which we survey in the next section, we are not aware of any reports on real-world applications of this technique to sensor configuration.

In this light, here we investigate whether 802.11-to-802.15.4 CTC can be employed to address the considered sensor configuration problem. Focusing on this particular application of CTC presents both challenges and opportunities, the combination of which, to the best of our knowledge, has not been explored to date. For example, on the one hand, prior approaches to CTC usually assume extensive control over the operation of communicating devices, such as the possibility of precisely timing transmissions. In contrast, unless their firmware is modified, which we must not assume in our case, smartphones offer virtually no such low-level control for applications. On the other hand, sensor configuration usually involves transferring only little information and just in one direction—from a smartphone to a sensor—as for any further setup after the initial configuration, the sensors can

be reached normally, without CTC. Moreover, it may be assumed that the communicating devices are physically close to each other. Our study thus involves implementing CTC for the considered application and evaluating the resulting solution. We illustrate the encountered problems and highlight some unanticipated, albeit useful properties of CTC itself.

The rest of the paper is organized as follows. Section 2 gives a background and surveys past work on CTC. Section 3 is a record of our efforts to implement CTC for the considered problem. Section 4 evaluates the proposed solutions. Finally, Section 5 highlights the outcomes of our work.

2 Background

CTC between 802.11- and 802.15.4-based radios relies on the fact that these two standards occupy roughly the same frequencies within the 2.4 GHz band. In general, this is disadvantageous because sharing the spectrum by different technologies may lead to interference. To prevent unwanted transmission collisions, many transceivers are thus capable of detecting and quantifying the activity on a radio channel. This is also the case for typical 802.15.4-based sensors, which can measure so-called *received signal strength indicator* (RSSI), that is, the power of signals accumulated over a given range of frequencies at a specific moment. This functionality is commonly used for clear channel assessment or dynamic channel assignment. However, it is also a prerequisite for CTC. In a nutshell, the idea behind CTC consists in encoding data as timings of transmissions in one standard (in our case, 802.11), which can later be sensed by radios of the other standard (in our case, 802.15.4). The details of this technique differ between various algorithms, though.

2.1 Related Work

More specifically, the subject of direct communication between seemingly incompatible radio technologies has already attracted considerable attention from the research community. Apart from several attempts to create CTC-like solutions [1, 12, 11, 4, 2, 9, 5, 3], there are also proposals of protocol suites for such communication [13] and attempts to make existing CTC frameworks more reliable and efficient [10, 8, 7]. Analyzing those works, one can distinguish three main methods of transferring information from 802.11 to 802.15.4. In the first one [1, 11, 3], a 802.11-based device sends packets of certain sizes, which result in energy bursts of known durations. The role of a 802.15.4 radio is then to detect the interval of increased RSSI and map it to a predefined symbol. The second technique [12], rather than sending dedicated packets, relies on appending preambles of energy patterns to regular ones. This approach involves modifications to the firmware of the transmitting device, and hence is not suitable for our use-case. Finally, the third technique [4] does not generate any additional traffic at all. Instead, it shifts transmission times of consecutive beacons, which are regularly sent by 802.11 access points. In our scenario this would require commodity smartphones to control the timings with a precision that, again, cannot be realized without changing the firmware.

After rejecting the approaches that are not practically implementable on customers' smartphones, our possible choices are limited to just one method.

2.2 Base Algorithm

The particular technique that does not violate any of our assumptions is Esence [1] with further improvements proposed as part of HoWiES [13]. In this method, symbols of an alphabet used to transfer information are encoded as durations of packets transmitted by a 802.11 device and then decoded by a 802.15.4 device based on RSSI readings. However, neither Esence nor HoWiES has a publicly available implementation and their published descriptions lack several crucial details.

To start with, to employ the considered method, one has to define an energy threshold (a RSSI level) that discriminates between *neutral* (interpreted as background noise) and *positive* (interpreted as information) samples sensed by a 802.15.4 radio. Only then is it possible to detect the number of consecutive positive (high-energy) samples and translate the sequence length into a certain letter of the alphabet. Choosing such a threshold is not trivial in practice, though.

Likewise, selecting proper packet sizes that result in sample sequences of specific lengths has a significant impact on the performance of CTC. First, the sequences should be distinguishable from each other given a certain 802.15.4 RSSI sampling frequency, so that they can be unambiguously translated into letters. Second, they should also differ in length from "typical" network packets, so that normal 802.11 communication is not confused with CTC.

For their prototypes, in turn, the past works assume that the communicating devices are fully controllable, which allows for optimizing CTC by tweaking low-level parameters. For example, to generate energy bursts of the widest variety of unique lengths, 802.11 devices are made to transmit packets at the possibly lowest speed: 1 Mbps [13]. This, however, is not an option for smartphones that, unless their firmware is modified, do not offer such a level of control to applications. Furthermore, both sides of communication are assumed to use overlapping frequency bands. To this end, either they are preconfigured to operate on certain channels or one of them loops through different channels. Again, this is impossible or inefficient or both for our scenario.

From these and similar observations, we conclude that the past solutions, dedicated for wireless access points and laptops, may not be suitable for customers' mobile devices and it is necessary to rethink certain aspects of CTC with respect to our use-case of sensors commissioning.

3 Adapting CTC for Sensor Configuration

So far the idea of crosstalk-based communication, a reliable way to send data directly to transceivers based on a different radio technology, has been designed, deployed and evaluated solely for situations in which all of involved devices are properly set up beforehand. We attempt to adapt this mechanism to the scenario where our control over one side of communication is strictly limited. Since we want to provide users with a *simple* and *convenient* way to configure sensors with a smartphone or another 802.11 device:

1. CTC-related parameters of this device can be modified only via a dedicated mobile application,
2. the effort expected from end-users should be reduced to running the program,

3. the program itself must not require the device to be rooted.

Throughout the process of deployment we need not only to overcome these constraints but also to derive possible advantages from the specificity of our use-case. Therefore, our implementation should rest upon the following assumptions:

1. software of 802.15.4 devices can be entirely customized prior to the communication process,
2. no feedback from 802.15.4 is needed: the communication does not need to be bidirectional,
3. sensors are configured only from within a short distance (up to 1 m),
4. the bit rate does not need to be high: the amount of configuration data is already limited due to sensor parameters,
5. the process of sensor commissioning takes place in locations with moderate noise, that is, settings typical for smart-home devices.

With the above remarks in mind, we begin our work on the CTC-like framework.

3.1 Preparing the 802.15.4 Side

The first step is to implement an 802.15.4 application that detects RSSI at a particular moment on a selected channel. In our case sampling can be performed every 250 µs, due to finite hardware capabilities and associated software operations. The program works on our own TinyOS-based operating system dedicated for motes and, for the purpose of experiments, runs on a CC2650 MCU with 802.15.4 transceiver connected to the SmartRF06 Evaluation Board. The radio sensitivity equals −100 dBm, which means that RSSI measured by the device has the range from 0 dBm (the strongest signal) to at least −100 dBm (the weakest signal).

3.2 Selecting the RSSI Level

Our next goal is to distinguish RSSI of samples that may indicate CTC activity from background interferences. For this purpose we analyze various surroundings in terms of the present noise. Places, where we collect data include two apartments in a block of flats (with and without a 802.11 access point) and two rooms at a university department (with and without a 802.11 access point). The highest signal levels given by `iwlist` program on our laptop, ASUS K455L, for each of these environments are consecutively: −60 dBm, −79 dBm, −46 dBm, −63 dBm. Concerning our task, we are interested mainly in measurements taken by a 802.15.4 transceiver in the department room with the access point (AP) inside, that is, the one with the highest interference. An exemplary sequence of registered signals is presented in Figure 1.

We notice that in all settings most of the impulses sensed by the radio belong to a light background noise (a noise floor), while packets transmitted by the nearby 802.11 devices result in rare but intensive, a few samples long bursts of energy. Secondly, average strength of both kinds of noise depends on the channel that the radio listens on: if the channel overlaps with occupied frequencies (especially with commonly used 802.11 channels 1, 6, 11), levels of detected noise are higher. For this reason, it seems sensible to define

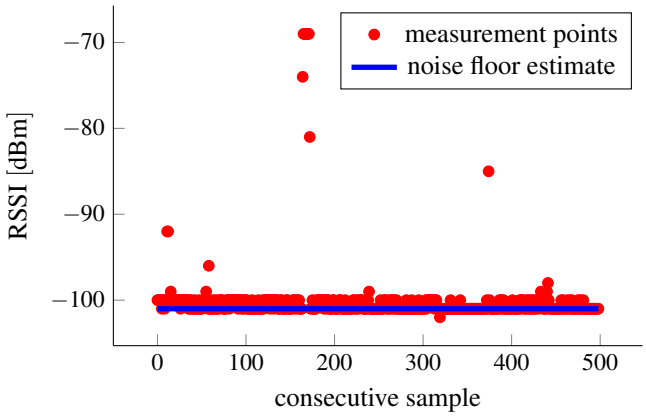

Figure 1. RSSI values recorded by the 802.15.4 device on channel 19 in the room with a 802.11 AP.

the sought RSSI threshold, $E_{threshold}$, in relation to the energy of the noise floor, E_{noise}. Consequently we say that a sample is low-energy if its energy E_{sample} does not exceed the threshold over the noise floor: $E_{sample} \leq E_{noise} + E_{threshold}$. To that end we compute the noise floor estimate as an exponential moving average of detected low-energy samples. Our algorithm uses every 80th low-energy sample to update E_{noise} in accordance with the formula:

$$E_{noise} := \min(\gamma,\ E_{sample} \cdot \alpha + E_{noise} \cdot (1-\alpha))$$

Based on preliminary experiments, we set, $\alpha = \frac{1}{256}$ and $\gamma = -98$ dBm.

With the method of estimating the noise floor we set the 802.15.4 radio to record the surrounding noise and identify *chirps*, sequences of high-energy samples, for a range of possible thresholds. Since in case of this experiment observed chirps result solely from interference we want to choose a threshold that is high enough to eliminate most of this unwanted network traffic. Figure 2 presents the outcome of our measurements carried out in a department room with a 802.11 AP: the average percentage of chirps "eliminated" by setting the energy threshold at certain level.

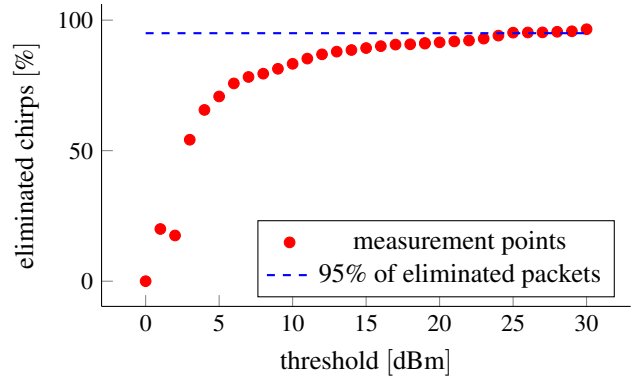

Figure 2. The average percentage of chirps below a chosen threshold. The reference number of chirps is the one measured for $E_{threshold} = 0$.

The data suggests that if the threshold $E_{threshold}$ is set at

the level of 25 dBm, less than 5% of interfering chirps are considered by the 802.15.4 radio as potential CTC messages (in quiet environments, with no AP in the same room, similar percentage of chirps is sieved out by $E_{threshold}$ equal to 15 dBm). Hence, as for now, we decide to set $E_{threshold}$ at the value of 25 dBm and later on we make sure that for this threshold typical mobile devices are powerful enough to transmit our CTC letters as high-energy bursts.

3.3 Preparing the 802.11 Side

The next step requires us to prepare a 802.11 device, i.e. the transmitting side of unidirectional CTC. For this role we choose Samsung Galaxy S4 and use it for almost all experiments, including the one where it is compared with other models of smartphones in terms of performance. At first, since the device is not rooted, we are not able to control which 802.11 channel is used: to our best knowledge, neither Android nor iOS users are allowed to change a default 802.11 channel via the application interface.

Likewise, it is not possible to set a specific bit rate of transmission, which in our case should be very low: the lower is the bit rate, the more chirp lengths can be reserved for CTC ([13]). To address this need, we leverage the fact that configured access points on mobiles ubiquitously use 1 Mbps as its lowest 802.11 basic rate parameter for maximum compatibility. This means that if we turn a smartphone into a wireless AP, its broadcast packets have fixed, 1 Mbps, transmission rate. Consequently, our 802.11 application dedicated for Android operation system (API 19 and above) enables tethering immediately after its start. Then a user is able to send a specified number of packets of a custom size.

The interval between sending two consecutive packets is another of relevant parameters: if CTC packets are transmitted too quickly low-power radio might see them as one longer chirp, but if they are unnecessarily delayed the rate of communication decreases. Figure 3 shows the number of correctly received chirps for various intervals in the quietest environment. Packet sizes range from 200 B to 1400 B while the applied threshold equals 25 dBm. As a result, we decide to transmit each packet within a window of 20 ms: to ensure that for the chosen settings 99% packets can be decoded with a sensible margin of safety.

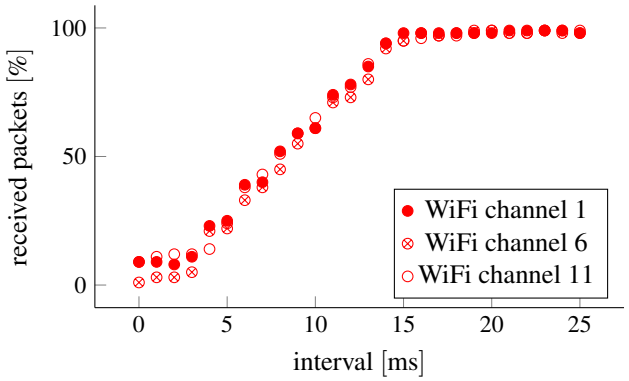

Figure 3. The percentage of correctly decoded chirps measured for various transmission intervals, three different 802.11 channels (1, 6, 11) and 802.15.4 channel 17.

3.4 Selecting the Alphabet

Now that we have implemented both applications, the last missing element of our CTC mechanism is the alphabet: a set of packet sizes that, during the transmission, generate chirps of certain lengths, which the 802.15.4 radio interprets as symbols. Yet before choosing the CTC letters, we carry out additional measurements to find out what chirp lengths are most frequent among interfering signals. The results are presented in Figure 4.

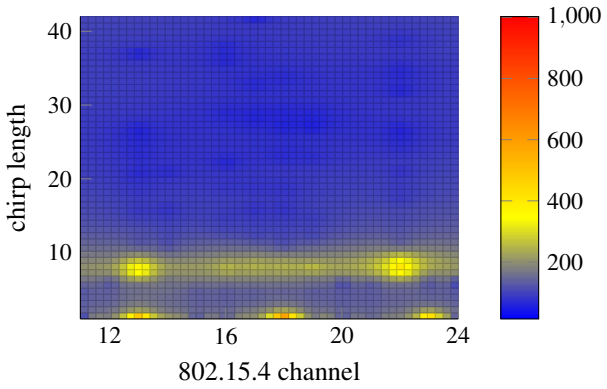

Figure 4. Count of chirps of a given length for various 802.15.4 channels. For channels 25 and 26 no chirps above the threshold (25 dBm) **were recorded.**

We observe that single samples and 8-sample packets dominate, whereas longer transmissions take place occasionally. Hence it is reasonable to build the CTC alphabet out of these packet sizes that correspond to unusually long chirps. In practice this conclusion means that the selected packet size cannot be smaller than 200 B. The other limit is imposed by the default IP MTU, which does not exceed 1500 B and can be changed only on a rooted smartphone.

Despite the two-fold constraint, we aim to determine as many distinct CTC letters as possible so that the amount of information represented by a single letter is maximized. To accomplish this goal we analyze chirps generated by various packets within the mentioned size limits. It turns out that each packet size can be sensed by 802.15.4 as chirps of four different lengths. For example, in 95% of cases, a 200 B message is detected as 10 or 11 consecutive high-energy samples. The remaining 5% refers to situations where 802.15.4 records 17 or 18 high-energy samples. This secondary length shifted from the basic one by 7–8 samples can be observed for all packets and the bigger the packet size is, the more frequently the radio senses lengthened signals (up to 15% of cases). The reason behind this phenomenon are 8-sample beacons that are regularly sent by the smartphone AP: sometimes, a beacon needs to be transmitted immediately after broadcasting a CTC message, while imperfect accuracy of sampling makes the radio merge these two transmissions into one.

After taking into account this disadvantageous side-effect, we determine a 5-letter alphabet out of the following packet sizes: 200 B, 500 B, 800 B, 1100 B, 1400 B. The last letter can be represented by a chirp of 56 high-energy samples.

4 Evaluation

At last, having prepared the software we can move on to tests, which aim to verify if crosstalk-based communication could be deployed to provide a user-friendly way to configure sensors.

4.1 Various Smartphones

In the first step we want to find out whether 802.15.4 radio is able to notice any difference between transmissions carried out by distinct smartphones. For the sake of the experiment we install our Android application on three devices, which then broadcast 1500 letters. The smartphones send data one at a time and in the same environment: a department room with no 802.11 AP. For the currently active smartphone its distance from the sensor mote (15 cm), 802.11 hotspot channel (6) and 802.15.4 channel (17) remain the same throughout all measurement series. The transmitted packets are collected and analyzed to make a comparison presented in Table 1. We conclude that devices may differ significantly in the signal strength, but the CTC mechanism itself (broadcasting messages of certain lengths and at the lowest bit rate) works as expected regardless of the smartphone model.

Table 1. Transmissions by various smartphones.

smartphone	average RSSI	received packets	time taken
Samsung Galaxy S4	−71 dBm	97.8%	29 s
Moto 3rd Gen	−78 dBm	99.5%	30 s
HTC Desire	−63 dBm	98.4%	30 s

Also, time taken by the radio to receive the chirps reveals a weakness of the framework: a low bit rate (12 Bps if we reserve one of the letters for the preamble). This, however, is not problematic in case of our specific scenario.

4.2 Varying Channels

Next two experiments test CTC in situations, where the 802.15.4 channel does not overlap with the 802.11 AP frequency. We carry out measurements in the quietest of four environments and for all combinations of channels (we use a rooted smartphone to explicitly control the 802.11 channel). During the transmission the distance between the communicating devices equals 0 cm (the devices touch each other). Both experiments require the smartphone AP to broadcast the whole CTC alphabet hundreds of times.

As shown in Figure 5, in the described settings even if channels do not overlap, the 802.15.4 radio is able to sense activity of the CTC origin.

In the second experiment the settings remain the same with one exception: we put the smartphone even closer to the sensor (it is slipped underneath the radio). Surprisingly, what can be noticed in Figure 6, this small change results in the 802.15.4 device being able to decode chirps even if the channels occupy faraway frequencies. We explain this unexpected benefit mainly with modulation sidebands of transmitted signals and near field interference. Imprecise calibration of the receiver might contribute to this effect as well.

4.3 Varying Distance

The significant gain brought about by the close proximity of the devices inspires us to investigate CTC performance for

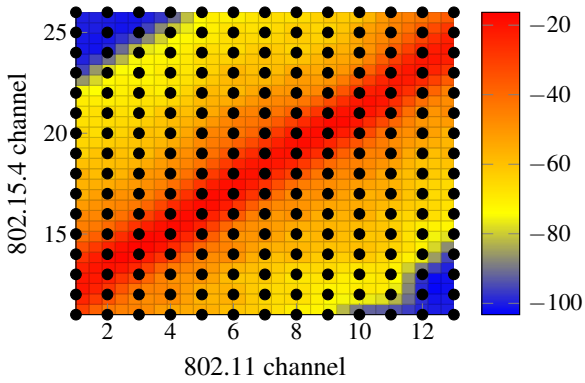

Figure 5. RSSI of decoded chirps for varying combinations of 802.11 and 802.15.4 channels.

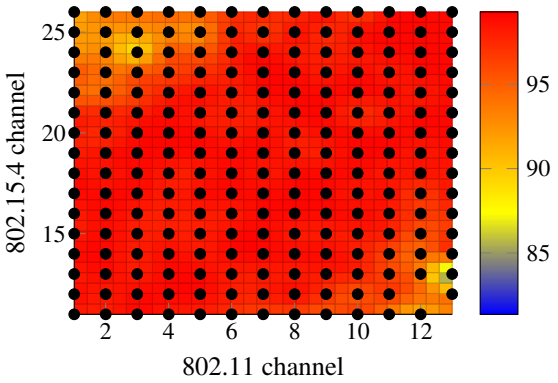

Figure 6. Percentage of correctly decoded packets for various 802.11 and 802.15.4 channels.

a variety of distances, which, in accordance with our initial assumptions (Section 3), should not exceed 1 m in the use-case scenario. For this purpose we carry out experiments in the quiet environment with the RSSI threshold of 15 dBm. As a consequence of these settings, if RSSI of CTC transmission is greater than −76 dBm more than 97% of chirps get correctly decoded, whereas for smaller RSSI values the reception rate rapidly decreases. Hence, in order to precisely observe quality of CTC, we require the 802.15.4 radio to compute the average RSSI of CTC transmission with respect to the distance.

In the first experiment the 802.15.4 channel is fixed while the 802.11 channel and the distance between the communicating devices vary (Figure 7).

In the other experiment we fix the value of the 802.11 channel whereas the 802.15.4 channel and the distance are variable (Figure 8). The collected data confirms that it is possible to receive CTC messages on a non-overlapping channel, however the devices need to be in close proximity.

5 Discussion

To sum up, to apply CTC in the selected sensor configuration scenario, one has to overcome several problems. After finding an appropriate RSSI threshold for noisy environments, choosing inter-packet intervals, and enforcing certain transmission rates for a user's smartphone, the last difficulty lies in the lack of control over the 802.11 channel

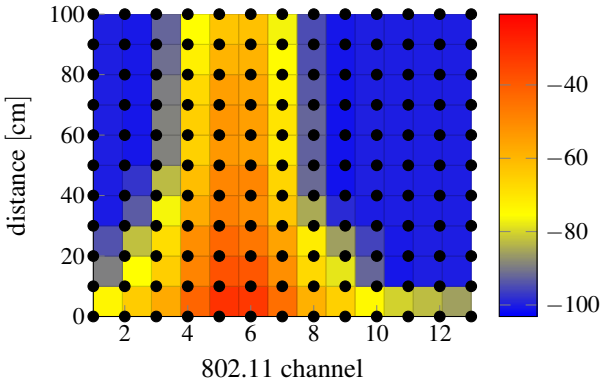

Figure 7. RSSI of decoded chirps for the fixed 802.15.4 channel (17) over varying 802.11 channels and distance.

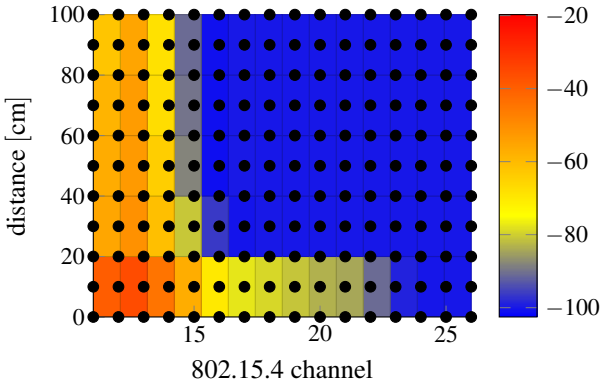

Figure 8. RSSI of decoded chirps for the fixed 802.11 channel (1) over varying 802.15.4 channel and distance.

the smartphone selects: the channel need not overlap with the frequencies on which the 802.15.4 sensor listens. The cited literature on CTC suggests that 802.15.4 should loop through its channels to find an optimal one. However, the results of our experiments, namely the fact that under some conditions an 802.15.4 radio listening on a central channel is able to decode 98% CTC chirps from all 802.11 frequencies of the 2.4 GHz band, motivate a different approach. The idea consists in the 802.15.4 device looping through RSSI thresholds instead of channels. The radio starts sampling with a maximal RSSI threshold and, as long as it fails to decode a specific CTC preamble, the threshold is decreased. Only after reaching the minimal threshold does the radio switch the channel. In the case of a quiet environment and small distance between devices, this approach saves the time needed for radio calibration and allows for applying an approximately optimal threshold. More importantly, it seems to work in all of the aforementioned environments and regardless of the selected 802.11 channel.

However, as mentioned previously, the considered solution offers extremely limited bit rates and is designed only for applications that meet certain restrictive conditions. Therefore, it would be interesting to study whether WEBee [6], an approach developed in parallel to our work and presented only recently, which promises an orders-of-magnitude better performance than any past CTC-based solutions, can indeed be applied in our scenario, notably without modifying the firmware of smartphones.

6 Acknowledgments

The presented work was partially supported by the National Center for Research and Development (NCBR) in Poland under grant no. LIDER/434/L-6/14/NCBR/2015.

7 References

[1] K. Chebrolu and A. Dhekne. Esense: Communication through energy sensing. In *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking*, MobiCom '09, pages 85–96, New York, NY, USA, 2009. ACM.

[2] Z. Chi, Y. Li, H. Sun, Y. Yao, Z. Lu, and T. Zhu. B2w2: N-way concurrent communication for iot devices. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, SenSys '16, pages 245–258, New York, NY, USA, 2016. ACM.

[3] D. Croce, N. Galioto, D. Garlisi, F. Giuliano, and I. Tinnirello. An inter-technology communication scheme for wifi/zigbee coexisting networks. In *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks*, EWSN ’17, pages 305–310, USA, 2017. Junction Publishing.

[4] S. M. Kim and T. He. Freebee: Cross-technology communication via free side-channel. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom '15, pages 317–330, New York, NY, USA, 2015. ACM.

[5] Y. Kondo, H. Yomo, S. Tang, M. Iwai, T. Tanaka, H. Tsutsui, and S. Obana. Energy-efficient wlan with on-demand ap wake-up using ieee 802.11 frame length modulation. *Computer Communications*, 35(14):1725 – 1735, 2012. Special issue: Wireless Green Communications and Networking.

[6] Z. Li and T. He. Webee: Physical-layer cross-technology communication via emulation. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, MobiCom '17, pages 2–14, New York, NY, USA, 2017. ACM.

[7] S. Tang, H. Yomo, and S. Obana. Dynamic threshold selection for frame length-based wake-up control. *IEEE Wireless Communications Letters*, 4(6):609–612, Dec 2015.

[8] S. Tang, H. Yomo, and Y. Takeuchi. Optimization of frame length modulation-based wake-up control for green wlans. *IEEE Transactions on Vehicular Technology*, 64(2):768–780, Feb 2015.

[9] S. Tang, H. Yomo, S. Yamaguchi, A. Hasegawa, and S. Obana. Exploiting frame length of 802.15.4g signals for wake-up control in sensor networks. In *2015 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1578–1583, March 2015.

[10] I. Tinnirello, D. Croce, N. Galioto, D. Garlisi, and F. Giuliano. Cross-technology wifi/zigbee communications: Dealing with channel insertions and deletions. *IEEE Communications Letters*, 20(11):2300–2303, Nov 2016.

[11] S. Yin, Q. Li, and O. Gnawali. Interconnecting wifi devices with ieee 802.15.4 devices without using a gateway. In *2015 International Conference on Distributed Computing in Sensor Systems*, pages 127–136, June 2015.

[12] X. Zhang and K. G. Shin. Gap sense: Lightweight coordination of heterogeneous wireless devices. In *2013 Proceedings IEEE INFOCOM*, pages 3094–3101, April 2013.

[13] Y. Zhang and Q. Li. Howies: A holistic approach to zigbee assisted wifi energy savings in mobile devices. In *2013 Proceedings IEEE INFOCOM*, pages 1366–1374, April 2013.