SoNIC: Classifying and Surviving Interference in 802.15.4-based Sensor Networks

Olof Rensfelt
IT Department
Uppsala Universitet, Sweden
olofr@it.uu.se

Frederik Hermans
IT Department
Uppsala Universitet, Sweden
frederik.hermans@it.uu.se

Thiemo Voigt
Swedish Institute of Computer Science
thiemo@sics.se

Edith Ngai
IT Department
Uppsala Universitet, Sweden
edith.ngai@it.uu.se

Lars-Åke Nordén
IT Department
Uppsala Universitet, Sweden
lln@it.uu.se

Per Gunningberg
IT Department
Uppsala Universitet, Sweden
perg@it.uu.se

ABSTRACT

Sensor networks that use 802.15.4 in the 2.45 GHz ISM band are prone to radio interference from devices such as microwave ovens, WiFi devices, and Bluetooth devices. Interference can cause packet loss and thus reduces network performance and lifetime. Online detection of interference in sensor networks is challenging, because nodes cannot decode foreign transmissions due to incompatibility between 802.15.4 and other technologies.

We present SoNIC, a system that enables sensor networks to detect and classify cross-technology interference. At the core of SoNIC lies an interference classification approach that assigns individual corrupted 802.15.4 packets to different interference classes. A voting algorithm fuses the classification results to detect the presence of interferers. The output of the voting can be used to select a mitigation strategy. In contrast to other solutions, SoNIC does not require active spectrum scanning or specialized hardware. It runs on ordinary TelosB sensor nodes using Contiki.

This technical report describes the core idea of SoNIC, selected system aspects of it (including three mitigation strategies), as well as initial experiments in an office environment.

1. INTRODUCTION

The 2.4 GHz ISM band is available worldwide for unlicensed radio operation and consequently, a vast number of devices using a range of technologies operate in this frequency band—for example household microwave ovens, WiFi laptops, Bluetooth game controllers, and baby monitors. Sensor networks often use 802.15.4 for radio communication, commonly also operating in the 2.4 GHz band. Therefore, sensor networks suffer from cross-technology interference, when their communication overlaps in time, frequency and space with other wireless networks or interference from other electronic equipment such as microwave ovens. Cross-technology interference incurs packet loss if multiple devices using incompatible technologies access the medium at the same time [16, 22]. This can severely hamper the operation of sensor networks. Furthermore, because communication is commonly the most energy-intense operation in sensor networks, packet loss due to interference can cause retransmissions that reduce the lifetime of a sensor network.

In this paper, we consider the problem of interference classification and interference mitigation in 802.15.4-based sensor networks. We describe an approach that uses information collected from corrupted 802.15.4 packets to detect the presence of interferers. We establish that different types of interferers leave unique patterns in how they cause corruption in 802.15.4 packets, and these patterns can be used by sensor nodes to classify the source of interference. The classification accuracy of the approach was evaluated on a dataset collected in a controlled environment in previous work [11].

Unlike prior work, our approach of inspecting individual packets does not rely on costly spectrum scanning to detect interference [3, 23], or on additional hardware [15]. In contrast, it only incurs low energy overhead: a packet has to be completely received to decide whether it is corrupted, and our approach extracts interference information only from corrupted packets that are received during a sensor network's regular operation. Thus, the nodes' radio transceiver (commonly the most energy-hungry component) only has to be turned on when actually required by the duty-cycling protocol used in the sensor network.

We implement our approach in SoNIC, a SensOr Network Interference Classification system that classifies interferers, and selects a suitable mitigation strategy to reduce the effect of interference. SoNIC is sufficiently light-weight for operation on resource-constrained sensor nodes, as demonstrated through our prototype implementation for the Contiki OS on TelosB sensor nodes.

This technical report documents the current state of our ongoing development of SoNIC. It is organized as follows: Sec. 2 describes the technical background of radio technologies operating in the 2.4 GHz ISM band. Sec. 3 gives a description of our approach to interference detection and classification. The SoNIC system that implements the classification and combines it with interference mitigation strategies is described in Sec. 4. We describe initial experiments in Sec. 5. Related work is surveyed in Sec. 6, and the paper is concluded in Sec. 7.

2. TECHNICAL BACKGROUND

We briefly summarize the technical aspects of 802.15.4, 802.11b/g, Bluetooth, and microwave ovens that are relevant to our goal of interference classification.

2.1 802.15.4

The 802.15.4 standard defines a physical layer and a MAC layer for low-power, low-rate wireless networks [13]. We consider 802.15.4 at 2.4 GHz in this paper, because most interference is faced in this

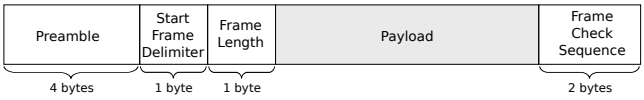


Figure 1: Format of an 802.15.4 packet

popular ISM band. At 2.4 GHz, 16 channels of 2 MHz width are defined with an inter-channel spacing of 3 MHz. A maximum transmission power of 0 dBm is common. The standard implements direct sequence spread spectrum by mapping each four-bit symbol to be transmitted to a pseudo-random 32-chip sequence. Offset quadrature phase-shift keying is used for modulation. The data rate is 250 kbps, the symbol period is 16 µs.

The format of an 802.15.4 PHY packet is shown in Fig. 1. Each packet begins with a preamble, which consists of four zero-bytes, followed by a one-byte start frame delimiter (SFD) field with a fixed value. The frame length field contains the number of the packet's payload bytes, which may be up to 127. The length includes the two-byte frame check sequence (FCS) field which trails the packet payload. The FCS field contains a checksum which is calculated over the length field and the payload bytes. A receiver synchronizes to incoming zero-bytes; after receiving four zero-bytes, it scans for an SFD. Only after correctly receiving the SFD, it reads the following payload field and then reads the specified number of payload bytes. If no SFD is received after four zero-bytes, the receiver synchronizes to incoming zero-bytes again. A receiver can detect transmission errors by comparing the FCS against the checksum calculated for the received packet.

2.2 802.11b/g

The amendments b and g describe the two most prevalent physical layer implementations of the 802.11 standard for wireless local area networks [14]. Fourteen channels of 22 MHz width are defined, of which eleven are available worldwide. Nominal transmission power is between 15 dBm and 20 dBm, and bit rates range from 1 MBit/s to 54 MBit/s using a variety of modulations. The MAC layer is a variant of CSMA/CA and mandates a minimum interframe spacing.

2.3 Bluetooth

Bluetooth is a standard for short-range radio communication within personal area networks [12], and is often used for connecting peripherals. Bluetooth divides the 2.4 GHz band into 79 bands of 1 MHz each and employs frequency hopping over these bands. Devices in a network synchronize on a hopping sequence and stay on each frequency for a slot duration of 625 µs. The most common class of Bluetooth devices has a maximum transmission power of about 4 dBm.

2.4 Microwave oven

Residential microwave ovens are kitchen appliances that are used to heat food using non-ionizing microwave radiation in the 2.4 GHz band. Usually the whole band is affected by the emissions, with ovens exhibiting a frequency sweeping behavior. Minor variations in the affected frequencies can be observed for different models. The ovens' emissions alternate between on and off phases every $\frac{1}{2f}$ s during the heating period, where f is the mains frequency. The equivalent isotropically radiated power (EIRP) has been reported to range between 16 dBm and 33 dBm [7].

3. CLASSIFICATION APPROACH

This section gives a general overview of our interference classification approach. Our earlier paper describe the approach in more detail [11], but it does not include decision tree classifiers.

In previous work, we have studied the characteristics of different types of interferers on 802.15.4 traffic using an extensive dataset of interfered packets collected in an anechoic chamber. More specifically, we considered the influence of interference on individual 802.15.4 packets. It emerged that different types of interferers corrupt 802.15.4 packets in different ways. This observation can be explained by the characteristics of the different interfering technologies, such as when they access the channel, or how long an emission is.

To train classifiers to classify sources of interference using the information available when packets are received, we collected reference data in an anechoic chamber. In our measurements, we logged *corrupted packets*, i.e., packets that have been received, but that failed the CRC check, because the packet's FCS field did not match the checksum calculated over the packet's payload. For these packets, a receiving node knows that there has been a transmission (because it was able to decode the packet's preamble), but it also knows that the reception was erroneous (because the CRC failed). For each corrupted packet, we logged the following information: signal strength at the receiving node during packet reception, link quality indication (LQI), and what parts of a corrupted packet (i.e., which symbols) have been incorrectly received. Based on this information, which can be obtained by the sensor nodes themselves, we defined a set of features that capture the different patterns we identified for the interferers. These features are then fed into a classification algorithm which assign the packet an interference class, or indicates that the packet was corrupted not due to interference, but because the sender's transmission power was too low.

Classifying interference by assessing individual corrupted 802.15.4 packets has two salient advantages. First, it incurs limited overhead. A node has to receive and decode a packet completely in order to establish whether the packet has been received correctly. Thus, exploiting information from corrupted packets does not incur an additional sampling cost. This is in contrast to other approaches that try to detect interference by sampling the spectrum for characteristic emissions from interferers [3, 23]. It also does not require any additional hardware; the node's regular 802.15.4-compatible radio transceiver is sufficient. Second, by only inspecting corrupted packets, the sensor network will only capture foreign emissions that actually affect its communication. By contrast, if the nodes would sample the spectrum even when no sensor network communication is ongoing, they may detect (and react to) interferers that actually do not influence the sensor network communication, e.g., because they back off when detecting transmissions from the sensor network.

Collecting signal strength of a received packet requires a straightforward adaption of the radio driver in Contiki. The LQI of a packet is available from any 802.15.4-compatible radio transceiver at no extra sampling cost. To detect what parts of a packet are corrupted, we rely on link-layer retransmissions. If a node receives a corrupted packet, it stores it in a buffer; if it receives a correct copy of the same packet later on (due to link-layer retransmissions), it can detect the corrupted symbols by comparing the correct to the corrupted packet, as detailed in Sec. 4.1.

3.1 Features

The data that nodes collect for each corrupted packet is unsuitable for direct input into a classification algorithm because it consists of raw time series. Thus, we define a number of features that can be extracted from the time series that can be calculated on-line.

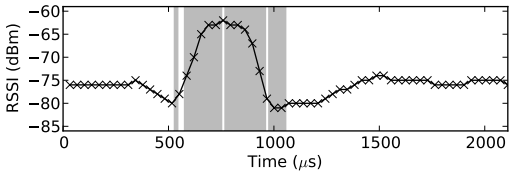


Figure 2: RSSI during reception of a packet interfered by a microwave oven. The gray area is where bits are corrupted in the packet.

LQI threshold. We define a binary feature that indicates whether the *LQI of a corrupted packet is higher than 90*. LQI can be considered to represent the chip error rate over the first two bytes of a packet. If a packet is received with a high LQI, but the packet fails the CRC check, we take this as an indication that channel conditions were good when reception started, but then deteriorated. Such a sudden change in channel conditions can be observed when an interferer starts emission during packet reception. In contrast, packets that cannot be decoded correctly due to insufficient signal strength usually have a low LQI value, because the channel conditions are poor over the whole packet reception time. The threshold value 90 was chosen empirically from our measurement data.

RSSI-related features. We define a binary feature to indicate whether the *range of RSSI values is greater than 2 dB*. For packets that are corrupted due to insufficient signal strength, the RSSI values often contain little variation, whereas distinct peaks in signal strength can usually be seen for interfered packets.

Before calculating other RSSI-related features, we normalize the RSSI series to values between 0 and 100. This normalization masks effects of distance and transmission powers, while preserving the series' shape. We define the features *mean normalized RSSI* to be the average value of the smoothed, normalized RSSI readings, and similarly, we use the *standard deviation of normalized RSSI*. These features are a simple means of characterizing the shape of a series of RSSI values, which is often indicative of the interferer. Though obviously two different shapes can have the same mean and standard deviation, we observe distinct distributions of these values for different types of interferers. Finally, we define a feature in terms of the *difference between the maximum normalized RSSI value and the most common RSSI value*, to capture a pattern that we observed in packets interfered by microwave ovens: the signal strength slightly drops, then peaks, and drops again (Fig 2).

Number of corrupted symbols. If we know the true payload of a packet, we can count the number of symbols that have been decoded incorrectly. We normalize this number by the total number of symbols in the payload. Radio technologies such as Bluetooth or 802.11g specify bit rates, minimum and maximum packet lengths, as well as inter-packet delays for medium access control. These specifications put a constraint on the amount of damage that can be done to a 802.15.4 packet, and hence the normalized number of corrupted symbols gives an indication of the type of interferer.

Error bursts. We define an error burst in a corrupted packet to be a sequence of corrupted symbols that may contain subsequences of at most four consecutive correct symbols. The feature *mean burst length*, *standard deviation of burst length*, and *bursts spanned* (the latter represents the number of symbols between the first burst's start and the last burst's end) further help capturing timing aspects related to the interferers' different radio technologies. Similarly, the feature *mean burst spacing* captures inter-burst spacings. An overview of all features can be found in Tab. 1

LQI-based	$\mathrm{LQI} < 90$
RSSI series-based	$\max(\mathrm{RSSI}) - \min(\mathrm{RSSI}) > 2\mathrm{dBm}$
	$\mathrm{avg}(\mathrm{RSSI}_{\mathrm{normed}})$
	$\mathrm{stddev}(\mathrm{RSSI}_{\mathrm{normed}})$
	$\max(\mathrm{RSSI}_{\mathrm{normed}}) - \mathrm{mode}(\mathrm{RSSI}_{\mathrm{normed}})$
Error burst-based	Number of corrupted symbols/payload length
	avg(burst lengths)
	stddev(burst lengths)
	End of last burst − start of first burst
	avg(burst spacings)

Table 1: Overview of features used for classification

3.2 Classification

In this paper, we describe two classifiers that are used in SoNET. First we have used a feed-forward artificial neural network (ANN) that has high classification accuracy. To reduce the computational complexity of classification, we have also used a decision tree classifier that has slightly lower classification accuracy.

3.2.1 Artificial Neural Network

The artificial neural network used for classification is trained offline on a dataset of corrupted packets collected in an anechoic chamber. Because the anechoic chamber is a controlled environment shielded from external interference, the corrupted packets can be tagged with the correct interferer. This tagged dataset is split into two parts where one part is used for training and the other part is used to evaluate how well the classifier performs. The datasets contain packets that are interfered by hardware from different vendors to reduce the risk that the ANN is trained on hardware specific characteristics.

The ANN is a feed-forward artificial neural network that is trained with a back-propagation trainer. We have evaluated support vector machine (SVM) classifiers, but the ANN was selected as it had good accuracy and could still fit on a resource-constrained node. The accuracy of the ANN and the SVM are presented in our previous work [11].

3.2.2 Decision Trees

We also investigated the use of decision trees [5], a conceptually very simple statistical classifier. Decision trees are interesting in the context of this work because they are computationally cheap to evaluate in comparison to support vector machines or neural networks.

In its simplest form, a decision tree is a binary tree; each leaf is labeled with a class name, and each inner node is labeled with a feature name and a value. To classify an input, the tree is traversed from the root until a leaf is reached. At each inner node, the input's feature is compared with the value stored in the inner node, and traversing continues at the left or right child depending on the result of the comparison. This process continues until a leaf is reached, which represents the classification result. Several algorithms exist to induce a decision tree from a given training set. We chose the widely-used C4.5 algorithm [19].

The decision tree classifier was induced from the same training set that was used to train the ANN and SVM. With the currently selected feature set, it reaches a mean classification accuracy of 78.5 %. The accuracy is around 70 % for packets interfered by WiFi, Bluetooth, or microwave, whereas packets corrupted due to a weak link are correctly classified with an accuracy of 99.2 %. We are still investigating the choice of features for the decision tree classifier, so these numbers may be subject to change in the future.

However, we expect future accuracy to be reasonably close to the accuracy presented here.

4. THE SONIC SYSTEM

After having laid out the general approach to interference classification, we now describe how this approach can be realized within sensor networks.

We are implementing SoNIC based on Contiki OS [6] version 2.4. SoNIC comprises feature calculation code, two classifiers (an artificial neural network and a decision tree), a packet storage and matching module, and an interference mitigation module that includes different mitigation strategies.

The high-level operation of SoNIC is as follows. When a corrupted packet is received, its payload, LQI and RSSI samples are stored by the packet storage module. When a packet is received correctly, the matching module is used to determine whether the received packet has the same payload as any previously received, corrupted packet in storage. If a corrupted version of the correctly received packet is found, the patterns of corrupted nibbles are extracted by a comparison of the payloads. The corrupted nibbles, RSSI samples, and LQI value are then forwarded to the feature calculation module. After the features are calculated, the classifier uses them to assign the corrupted packet to an interference class. The classification result is fed into the voting module, which outputs an estimated interference state. This estimation state is used to decide which, if any, mitigation strategy to enable.

We will now describe the individual modules in more detail.

4.1 Packet storing and matching

Our approach leverages retransmissions to identify the corrupted nibbles in packets that have failed the CRC check. To this end, each node stores corrupted packets in a FIFO buffer that holds recently received corrupted packets. When a packet that passes the CRC check is received, SoNIC matches the correct packet against the corrupted packets in the buffer. If a match is found, the corrupted regions can be easily identified by performing a nibble-wise comparison to the correctly received packet.

It is necessary to ensure that two distinct packets that have highly similar payloads are not accidentally matched to one another. We use two techniques to avoid accidental matches.

First, all transmitted packet payloads are randomized as follows. When a node sends a packet, it draws a random 8-bit bitmask. Prior to transmission, the node performs a bitwise exclusive or (XOR) operation of the bitmask and the payload. The XOR'ed payload is then transmitted along with the bitmask. A node that correctly receives the XOR'ed payload and the bitmask can simply restore the original payload by again performing the XOR operation, since XOR is its own inverse. These operations are performed in the radio driver and are thus transparent to MAC and higher layers. Importantly, the bitmask remains constant for any retransmissions of the packet. The randomization (and de-randomization) of the transmitted payload as just described helps to avoid matching payloads that are almost identical. If two similar packets have different bitmasks, they will have different representations on the radio channel, and hence are unlikely to be mistaken for one another.

Second, the matching process, which decides if a correctly received packet matches a stored, corrupted packet, gives a higher matching score if contiguous regions of two packets match. Matching works as follows: Two variables, *weight* and *score* are initialized to zero. The algorithm then iterates over the payloads' nibbles. If the *i*-th nibbles match, then *weight* is increased by one and *score* is increased by *weight*. If the *i*-th nibbles do not match, then *weight* is reset to zero and *weight* remains unchanged. If the final value of *score* is larger than 200, the two packets are considered to match. This simple procedure avoids accidental matching of packets that are distinct, yet share a periodic similarity (e.g., two packets that by chance match in every other payload byte). The threshold score of 200 was established empirically. We have found it to give good matching performance for the packet sizes of interest ($\geq$ 64 bytes).

4.2 Feature calculation

The calculation of most features is straightforward and computationally cheap. For example, features such as the mean of a series can be calculated sample by sample using few extra variables. The calculation of the mode is the most complex feature to be calculated. The mode is the most common value of a series, which means that all values need to be included individually when it is calculated. In the feature module, the mode is calculated using a 100 byte array in which each element represents a counter for a possible value in the series; the 100 byte array is sufficient, as RSSI samples are normalized between 0 and 100. Calculation of the mode is then possible in linear time by iterating over the series and keeping track of the maximum value. A more memory-efficient approach would be to sort the RSSI samples in place and then traverse the samples and to count the occurrences of each value. This alternative approach would increase CPU usage for saving memory usage and hence may be suitable for extremely memory-constrained node platforms.

4.3 Classification: neural network

We implemented a fixed point feed-forward neural network for classification on the sensor nodes. The training of the network is done on a desktop machine using the Python machine learning library PyBrain [18]. We have modified PyBrain to use a less complex neuron activation function. Instead of the default logistic sigmoid function in PyBrain (Eq. 1), we use Eq. 2 to train and evaluate the neural network because it is more suited for implementation using fixed-point arithmetics. This change is an optimization as exponential functions are expensive to implement in resource-constrained systems. The accuracy is not significantly affected when the approximated sigmoid function is used.

$$y = 1/(1 + e^{-x}) \tag{1}$$

$$y = -abs(x)/(k - abs(x) + 1), k = -1.4 \tag{2}$$

After the network error has converged to sufficiently small values during the training phase, the floating point weights are extracted and converted to fixed-point representation and then exported to C code source files. These auto-generated source files are then compiled into Contiki.

Classification on the sensor nodes is done by activating the neural network on a feature vector produced by the feature calculation module. The activation is implemented as a series of matrix multiplications and additions, which use fixed-point arithmetics. The current implementation takes approximately 300 ms to classify a corrupted packet, but we have not made an effort to optimize the classifier yet. Potential optimizations are to reduce the number of hidden neurons and optimize the fixed-point multiplications.

4.4 Classification: decision tree

Alternatively to the ANN classifier, SoNIC can use a decision tree classifier. The decision tree consists of 749 nodes, of which 374 are inner nodes and 375 are leaves. A textbook implementation of the tree would require several kilobyte of RAM for storing

comparison values and child pointers, and therefore is infeasible on a platform like TelosB, which provides only 10 kB of RAM in total. We apply three optimizations to reduce the memory consumed by the decision tree. First, fixed-point numbers of 16-bit width are used for comparison values instead of floating point. The fixed-point implementation gives a different classification result from the floating-point implementation for only one out of about 11,000 sample inputs in our testing set, indicating that the accuracy loss from using 16-bit width fixed-point values is negligible. As a second optimization, we do not store child pointers for each node, but instead store one parent pointer for each node. As a consequence, we can traverse the tree only by sequentially inspecting the list of nodes in $O(n)$ time (where n is the number of nodes) as opposed to $O(h)$ time for a representation using child pointers (where h is the height of the tree). Due to the low number of nodes, we believe this to be a reasonable trade-off between classification time and memory overhead. Finally, we do not store leaf nodes explicitly, but store them within the data structures for inner nodes.

These optimizations allow us to represent the decision tree with only 1870 bytes of RAM. On average, classification of a packet takes 1.137 ms, which is roughly 250 times as fast as classification with the neural network. While the neural network implementation is not optimized for speed, we expect the decision tree to be about two orders of magnitude faster; however, it also requires about 400 bytes more RAM than the neural network. Because both the neural network and the decision tree give similar classification accuracy, the choice between the two classifiers boils down to a trade-off between speed (decision tree is faster) and memory (neural network uses less memory).

These specific benchmark numbers are dependent on the feature set that the classifier is trained on, and thus, as we are still evaluating features, they may slightly change. Yet, as mentioned earlier, we believe the number to be indicative of performance for other feature sets, too.

4.5 Voting

Because the classifier sometimes assigns a corrupted packet to the wrong interference class as described in our earlier paper [11], making a mitigation decision on one individual classification might give poor performance. To increase the classification confidence, and thus reduce the risk of using unsuitable mitigation strategies, a history of recent classification results is used. This is done by keeping a window of classified packets that can be inspected when a mitigation decision is made.

A mitigation decision is triggered by a decrease in packet reception rate (PRR). The reasons to wait until PRR drops are, first, to ensure that mitigation is not initiated unless there are actual problems. Second, a decrease in PRR is often correlated with an increase in corrupted packets, which increases the chance of a correct mitigation decision. The length of the time window is used as a tuning parameter to configure how quickly SoNIC reacts to interference.

Because it is the receiving nodes that suffer from destructive interference, classifications of interfered packets from all links can be combined when the voter make its decision.

4.6 Mitigation strategies

When interference is detected and the type of interference is classified, a mitigation strategy may alleviate the effect of interference by reducing packet loss. We now describe three mitigation strategies that are suitable for use with our system. However, SoNIC is not bound in a specific way to any of these strategies.

Mitigating WiFi interference with channel adaption. WiFi interference occurs if the frequencies of a sensor network and a co-located WiFi network overlap. Thus, if a sensor network detects that it is being interfered by WiFi traffic, it may adapt its channel frequency. Channels in 802.11b and g are 20 MHz wide, whereas 802.15.4 channels are 2 MHz wide and any two adjacent channel center frequencies are separated by 5 MHz. A simple means of mitigating WiFi interference thus is for the sensor network to move four channels up, i.e., increase its channel center frequency by 20 MHz. In this way, the sensor network's channel will no longer overlap with the WiFi that caused the interference. Of course there is no guarantee that no other WiFi network or another interferer will operate in the new channel. Thus, the sensor network should continue to perform interference detection after switching to the new channel.

Mitigating WiFi interference with forward error correction. Because WiFi networks are ubiquitous in office environments, the previously described channel adaption strategy may not be able to find an 802.15.4 channel that is not used by another interferer. Thus, in some cases, the sensor network has to operate with a co-located WiFi network at a given frequency. Liang et al. describe an approach to combat WiFi interference in 802.15.4 sensor networks [16]. Their approach relies on two mechanisms; first, it uses repeated PHY headers to mitigate the effect of interferers that are close (< 1 m) to a receiving node. Second, it uses forward error correction (FEC) to facilitate the reconstruction of corrupted packets at receiving nodes. Liang et al. investigated different FEC algorithms and conclude Reed-Solomon coding with a redundancy of 30 bytes to give the good results. This allows receiving nodes to correctly reconstruct corrupted packets with up to 15 corrupted bytes. We have integrated FEC into SoNIC, using the parameters described by Liang et al. Thus, a sensor network that detects that it is exposed to WiFi interference may use FEC to reduce packet loss.

During initial experiments we found that the computational overhead of encoding and decoding packets using FEC is high. Encoding a packet that has 64 bytes of payload takes around 33 ms; the decoding time is dependent on the amount of corrupted bytes in the received payload. For example, if a packet contains corrupted bytes, decoding will take approximately 70 ms. Due to this high overhead, the FEC is only of limited use at the moment. For example, the FEC cannot currently be combined with a MAC that uses link layer acknowledgments, because the 802.15.4 standard mandates that these are sent $12 \cdot 16\ \mu\text{s} = 0.192\text{ms}$ after packet reception. This timespan is too short to decide whether a corrupted packet can be reconstructed. Even with MAC layers that do not require link layer acknowledgments, the long duration of packet decoding and encoding may induce further packet loss due to RX buffer overflows in the radio transceiver if packets are received during the decoding or encoding process.

We note, however, that this drawback could be overcome if the FEC-related functionality was implemented in hardware, for example in a separate field-programmable gate array (FPGA).

Mitigating microwave oven interference with transmission timing. A key characteristic of microwave interference is that it is very regular in time. Due to the operating principle of microwave ovens, the ovens emit energy in the 2.45 GHz band periodically during a 10 ms on period, followed by a 10 ms off period. (The actual period durations are dependent on the mains frequency, and hence differ in different part of the world. The periods reported here are valid for Europe, where the mains frequency is 50 Hz.) This periodicity can be exploited to facilitate communication in presence of interference from a microwave oven.

Our simple microwave oven mitigation works as follows. If a node that uses the strategy sends a packet and receives a link layer

acknowledgment from the receiver, it will schedule its next retransmission 20 ms later than the transmission of the previous successful packet. The reasoning for this approach is that since the previous transmission was successful, it occurred during a time instance at which the microwave was not active. Due to the regularity of microwave interference, it is expected that the microwave will not be active 20 ms after the last successful transmission. If a node sends a packet and does not receive a link layer acknowledgment, it will schedule its next transmission to a point in time chosen uniformly at random within the next 20 ms. The next transmission time is randomized (as opposed to transmitting in 10 ms, when a microwave emitter would be expected to be turned off) because packet loss is not guaranteed to be caused by a microwave oven. Randomization helps to avoid accidental synchronization between nodes in case that packet loss is not due to microwave interference, but caused by two sensor nodes transmitting at the same time.

5. EXPERIMENTS

We present some initial experimental results for the collection of corrupted packets, the voting output during interference, and different parts of SoNIC.

5.1 Microwave oven mitigation

We performed an experiment to find out whether the microwave mitigation approach described in Sec. 4.6 is feasible. We set up two TelosB sensor nodes at a distance of around 0.5 m; one of the nodes continuously sent PING messages, to which the other node responded with a PONG message. At a distance of about 1.5 m from the nodes, a microwave oven was placed with a bowl of water inside. Because the goal of this experiment was to test the mitigation approach (and not to test the classification approach), the nodes were explicitly configured prior to each run either to use the timing-based mitigation or not to use it.

The nodes used the Contiki nullmac MAC layer, which is a simple non-duty-cycling MAC that performs a clear channel assessment (CCA) prior to transmissions. While state-of-the-art MAC protocols are far more sophisticated than nullmac, we used it because it allows us to exclude the impact of interference on a specific MAC protocol. Furthermore, rescheduling transmissions, as the mitigation strategy does, may have implications for the duty-cycling approach of more sophisticated MAC protocols; by using nullmac, we can ignore this additional complexity for the moment.

Figure 3 exemplifies the correct operation of the mitigation approach. The figure shows the signal strength recorded by a nearby listening node. Different types of elevations from the noise floor (which is around -95 dBm) are visible, which correspond to emissions from the two nodes and the microwave. The first peak represents a PING packet, which is immediately followed by a link-layer acknowledgment. Then, an emission from the microwave oven follows, which is repeated every 20 ms as expected from the oven's principle of operation. Emissions from the microwave oven are marked with dotted arrows. The peak following the first microwave emission cannot be attributed to either microwave or TelosB nodes of our experiment; it is most likely due to another device. Then, a PONG message is visible (slightly shorter than a PING), which is again followed by an immediate link-layer ACK, which in turn is followed by a microwave emission. Because both PING and PONG were sent successfully, the nodes each schedule subsequent transmissions after 20 ms, and the pattern (PING, microwave, PONG, microwave) is repeated. Thereby collision between the 802.15.4 traffic and microwave emissions is avoided.

The experiment results are summarized in Tab. 2.We define a successful transmission to be a transmission for which (i) the channel was clear at transmission time (i.e., the CCA was successful), and (ii) the link-layer ACK has been correctly received. The table shows that our simple microwave mitigation leads to a significant increase in successful transmission, confirming its general feasibility. Most transmission failures are due to CCA failures. This is because in our setup, nodes are placed close to the microwave and hence will defer from sending while it is active. With the mitigation strategy, nodes are less likely to attempt sending while the microwave is active.

	W/o mitigation	With mitigation
Successful transmissions	87.29 %	97.46 %
CCA failed	12.46 %	2.45 %

Table 2: Results from experiment on microwave mitigation. The mitigation strategy increases the number of successful transmissions by about 10 %.

We conclude from this experiment, that mitigating microwave interference by scheduling transmissions is generally feasible. However, to make the approach applicable to real deployments, it needs to be integrated with an energy-efficient, duty-cycling MAC layer. We leave this integration to future work.

5.2 Robot measurements

To investigate how the interference classification is affected by distance to an interferer and how many corrupted packets are received by sensor nodes given various distances from an interferer, we set up an experiment with a mobile robot using our Sensei-UU testbed [21]. The scenario is that a mobile sensor node passes a stationary interferer while communicating with a stationary sensor node. A robot in the testbed carried a TelosB node along a 32 m straight track. The stationary sensor node was placed close to the beginning of the track. We used two different types of interferers: A microwave oven that heated a bowl of water, and a WiFi access point that streamed constant bit rate UDP traffic to a WiFi station. The WiFi devices used a data rate of 54 MBit/s at a transmission power of 1 dBm; the traffic saturated the WiFi link. In each experiment run, only one interference type (microwave or WiFi) was active for the whole duration of the run.

The mobile node periodically polled the stationary node for data packets. All received packets, correct or corrupt, were logged and we calculated the rate of total received packets (including corrupted packets) and the rate of correctly received packets (excluding corrupted packets). These rates are shown in Fig. 4 over the time of the experiment; the mobile node started moving at the beginning of the experiment (0 s), passed the interferer roughly halfway through (at 160 s), and stopped moving at the end of the track (270 s). In the case of microwave interference, the microwave interferes with the mobile node all along the 32 m track (Fig. 4a), whereas the WiFi devices only interfere heavily while the mobile node is in close proximity, due to lower transmission power (Fig. 4b). The important information to extract from Fig. 4 is that there are corrupted packets to use for classification of interferers that are in the proximity of a sensor node.

In the next experiment, the mobile node classifies broken packets and inputs the classification results into the voting module (Fig. 5). In the run when the microwave is active, the voter outputs microwave as the dominant interferer over the duration of the run (Fig. 5a). This matches Fig. 4a where there was a steady rate of broken packets to use for classification. In the case of WiFi interference the output is less distinct (Fig 5b). WiFi has the highest output a majority of the time but the distinction to other interference

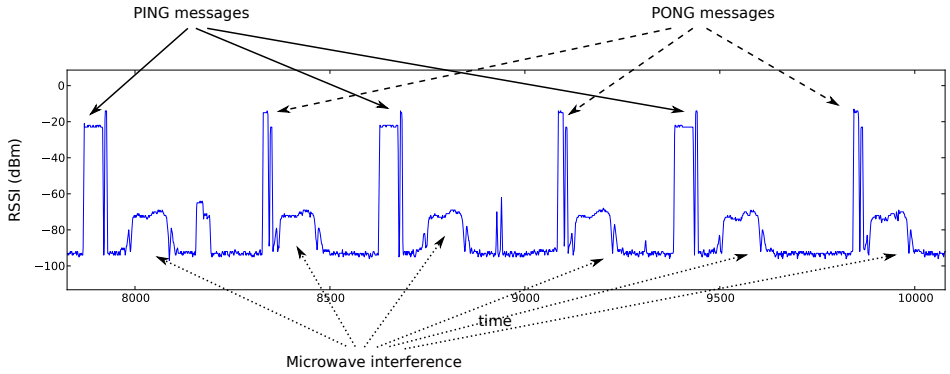
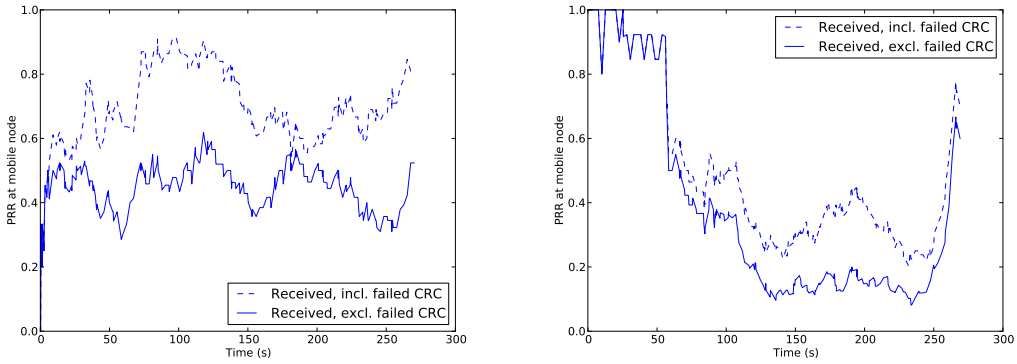


Figure 3: This plot of signal strength demonstrates the operation of the microwave mitigation strategy. Transmissions are scheduled in between the regular bursts of microwave emissions.



(a) In presence of a microwave interferer

(b) In presence of WiFi interferers

Figure 4: Total rate of received packets (including corrupted packets) and rate of correctly received packets when the mobile node passes different interferer types.

classes is less pronounced. While the mobile node passes the WiFi interferer, the voting output gets more distinct which is promising for selection of mitigation strategies.

The lower output values in the WiFi run is caused by a smaller amount of corrupted packets, and thus the confidence is lower. The lower rate of corrupted packets can also be seen in Fig 4b.

6. RELATED WORK

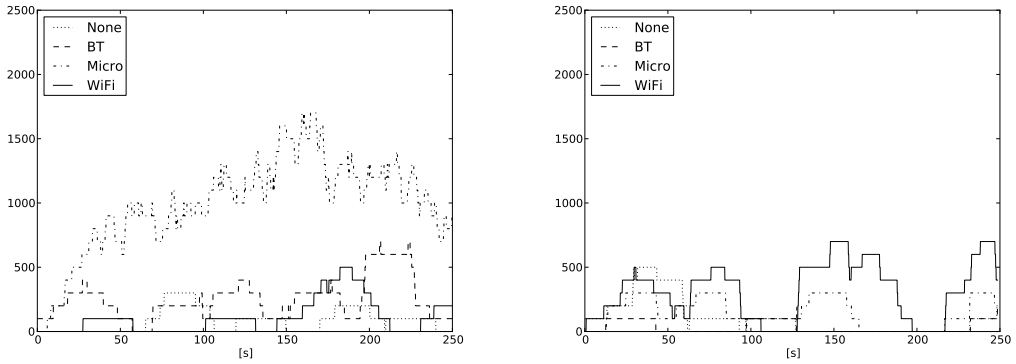
Relevant research in the WiFi domain includes Airshark [20], a system that uses standard 802.11 cards to sample the spectrum. The sampled data is analyzed using cyclostationary process methods to detect transmission patterns, which are then used to classify interferers. Another example is RFdump [15], which uses a software-defined radio to detect which devices are accessing the medium. RFdump aims to provide a tcpdump-like tool for the wireless communication. Gollakota et al. describe how antenna diversity in 802.11n can be exploited to reconstruct interfered signals [8]. All these approaches have in common that they require advanced signal processing capabilities that are usually not available in sensor networks. Cisco has developed a spectrum analyzer for network analysis that is capable of classifying radio devices [4].

In sensor networks, interference detection can help mitigation, which in turn increases the network lifetime by reducing unsuccessful communication attempts. Boer et al. have collected RSSI traces and manually assigned them to link quality classes. A Bayesian classifier is trained on the dataset with the goal to enable WSNs to quickly select the best channel [2]. Unlike in our approach, the classification can not be used to select a mitigation strategy but only helps to select a channel.

Chowdhury et al. describe an approach to interference classification by actively scanning channels for characteristic spectrum usage [3]. In contrast to our work, this approach comes at a higher energy cost, because the radio needs to be turned on even when no sensor network communication is ongoing. Their work is also concerned with interference mitigation. Hauer et al. describe how detection of WiFi interference can be used for interference mitigation [10]. Similar to our approach, they also consider RSSI during packet reception for identifying interference. Their work is concerned with selectively retransmitting parts of a packet that are suspected to be interfered, without having certain knowledge about what caused the corruption.

A large body of works considers the effect of interference on sensor networks in terms of high-level metrics such as packet reception rate. We selectively refer to [9, 17, 22] for an overview. How the effects of interference on protocol performance can be included in testbed experiments has also been studied [1].

7. CONCLUSION AND FUTURE WORK



(a) Microwave interferer

(b) WiFi interferer.

Figure 5: Voting output at mobile node during interference. The node passes the interferer at approximately 160 s.

We have presented our ongoing work on SoNIC, a system for 802.15.4-based sensor networks that detects and mitigates cross-technology interference. At the core of SoNIC lies an interference classification approach that classifies interferers only using information available when packets that have failed the CRC check are received. The output from the classification is fed into a voting module, and its output is used to choose a suitable mitigation strategy. We described three mitigation strategies in this paper, but SoNIC is not bound to these strategies; further strategies can be integrated with SoNIC.

SoNIC is a work in progress, and while its core ideas have been developed and implemented, some issues still need to be addressed in future work. Currently, we use the Contiki nullmac protocol with SoNIC to prevent interactions of a specific MAC protocol with interferers from leaking into experimental results. When SoNIC is to be used in production systems, it needs to be integrated with a state-of-the-art, duty-cycling MAC protocol. The mitigation strategies (especially the timing-based microwave mitigation) may need to be adapted to the chosen MAC layer.

The experiments that were described in Sec. 5 and in our previous paper [11] are concerned with the individual components of SoNIC. A full system evaluation is still to be done. To this end, we plan to evaluate the system in the following scenario: a mobile sink travels along a path to collect data from a number of stationary nodes. Along the path, the sink is exposed to different types of interferers, which it has to detect and mitigate. We believe that this scenario would be well suited to investigate the performance of SoNIC because of two properties. First, it is challenging because the sink is entering and leaving various zones of different types of interference, requiring the classification and mitigation to be sufficiently swift. Second, as the sink travels closer the interference, its influence increases; thus, the scenario lets us study the performance of SoNIC at different distances to interferers.

So far, we have focused on three types of interference: WiFi, Bluetooth, and microwave. Another possibility for future work is to add further interference classes to the neural network classifier. A related open question is how to handle interference that SoNIC is not trained to classify. In such a case, the system would react in either of two ways. If the interfered packets yield inconsistent classification results, the voting module will decide not to active a mitigation strategy, which is desirable. However, if the interfered packets are similar to a known interference class, SoNIC may erroneously activate a mitigation strategy. The associated cost (cost of the strategy plus cost of interference) again cannot be generally assessed. It may well be that if two different interferers have similar interference patterns, the same mitigation strategy applies to both of them. We believe that by training SoNIC to detect the most common types of interference, it can give robust performance increases in many deployment scenarios.

Another related problem is how to decide when to stop mitigation and when to transition between different strategies. For example, consider a set of nodes that detect that they are interfered by a microwave oven, and thus mitigate the interference. When should these nodes stop mitigation? Stopping mitigation prematurely may expose the nodes to interference, while stopping too late may waste transmission opportunities, because the microwave is still assumed to be emitting. The factors that influence this decision are related to the cost of the specific mitigation strategies used, and hence there is no general solution to the problem.

8. REFERENCES

[1] C. A. Boano, T. Voigt, C. Noda, K. Römer, and M. A. Zúñiga. JamLab: Augmenting sensornet testbeds with realistic and controlled interference generation. In *Proceedings of the* 10^{th} *IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 175–186, apr 2011.

[2] N. Boers, I. Nikolaidis, and P. Gburzynski. Patterns in the rssi traces from an indoor urban environment. In *Proceedings of the 15th International Workshop on Computer Aided Modeling, Analysis and Design of Communication Links and Networks (CAMAD)*, page 61–65, 2010.

[3] K. Chowdhury and I. Akyildiz. Interferer classification, channel selection and transmission adaptation for wireless sensor networks. In *Proc. of ICC '09.*, pages 1 –5, June 2009.

[4] Cisco Inc. *Cisco Spectrum Expert Wi-Fi*, 2012.

[5] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2 edition, Nov. 2001.

[6] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, pages 455–462, Washington, DC, USA, November 2004. IEEE Computer Society.

[7] P. E. Gawthrop, F. H. Sanders, K. B. Nebbia, and J. J. Sell. Radio spectrum measurements of individual microwave ovens – volume 1 & 2. NTIA Report TR-94-303-1, NTIA Report TR-94-303-2.

[8] S. Gollakota, F. Adib, D. Katabi, and S. Seshan. Clearing the RF smog: making 802.11n robust to cross-technology interference. In *Proc. of SIGCOMM '11*, pages 170–181, 2011.

[9] J.-H. Hauer, V. Handziski, and A. Wolisz. Experimental Study of the Impact of WLAN Interference on IEEE 802.15.4 Body Area Networks. In *Procs. of EWSN '09*, 2009.

[10] J.-H. Hauer, A. Willig, and A. Wolisz. Mitigating the Effects of RF Interference through RSSI-Based Error Recovery. In *Wireless Sensor Networks*, volume 5970 of *Lecture Notes in Computer Science*, pages 224–239. Springer, 2010.

[11] F. Hermans, O. Rensfelt, L. Åke Larzon, and P. Gunningberg. A Lightweight Approach To Online Detection and Classification of Interference in 802.15.4-based Sensor Networks. *ACM SIGBED Review, Special Issue on the Third International Workshop on Networks of Cooperating Objects*, 9(3), 2012.

[12] IEEE Computer Society. *Local and metropolitan area networks, specific requirements, part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs)*, 2005.

[13] IEEE Computer Society. *802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, 2006.

[14] IEEE Computer Society. *Local and metropolitan area networks, specific requirements, part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 2007.

[15] K. Lakshminarayanan, S. Sapra, S. Seshan, and P. Steenkiste. RFDump: An Architecture for Monitoring the Wireless Ether. In *Procs. of CoNEXT '09*, Dec. 2009.

[16] C.-J. M. Liang, N. B. Priyantha, J. Liu, and A. Terzis. Surviving Wi-Fi Interference in Low Power ZigBee Networks. In *Procs of SenSys '10*, pages 309–322, 2010.

[17] R. Musaloiu-E. and A. Terzis. Minimising the effect of WiFi interference in 802.15.4 wireless sensor networks. *Int. Journal of Sensor Networks*, 3:43–54, 2008.

[18] Pybrain. `http://www.pybrain.org/`.

[19] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[20] S. Rayanchu, A. Patro, and S. Banerjee. Airshark: detecting non-wifi rf devices using commodity wifi hardware. In *Procs. of IMC '11*, pages 137–154, 2011.

[21] O. Rensfelt, F. Hermans, P. Gunningberg, L.-Å. Larzon, and E. Björnemo. Repeatable experiments with mobile nodes in a relocatable wsn testbed. *The Computer Journal*, 2011.

[22] A. Sikora. Compatiblity of IEEE 802.15.4 (ZigBee) with IEEE 802.11 (WLAN), Bluetooth and Microwave Ovens in 2.4 GHz ISM-Band. Technical report, University of Cooperative Education Loerrach, 2004.

[23] S. Zacharias, T. Newe, S. O'Keeffe, and E. Lewis. Identifying sources of interference in rssi traces of a single ieee 802.15.4 channel. In *Procs. of ICWMC 2012*, 2012.