Detecting and Avoiding Multiple Sources of Interference in the 2.4 GHz Spectrum

***Abstract*—Sensor networks operating in the 2.4 GHz band often face cross-technology interference from co-located WiFi and Bluetooth devices. To enable effective interference mitigation, a sensor network needs to know the type of interference it is exposed to. However, existing approaches to interference detection are not able to handle multiple concurrent sources of interference. In this paper, we address the problem of multi-source interference detection, i. e., identifying multiple channel activities impairing a sensor network's communication, such as simultaneous WiFi traffic and Bluetooth data transfers. We present SpeckSense, an interference detector that distinguishes between different types of interference using a unsupervised learning technique. Additionally, SpeckSense features a classifier that distinguishes between moderate and heavy channel traffic, and also identifies WiFi beacons. In doing so, it facilitates interference avoidance through channel blacklisting. We implement SpeckSense on common hardware and evaluate its ability to classify concurrent interference and its use in selecting less interfered channels. Additionally, we show how SpeckSense improves the performance of an existing multichannel data collection protocol by 30%.**

I. Introduction

Low-power wireless sensor networks (WSN) operating in the 2.4 GHz spectrum often face radio interference from other devices that share the same frequency band. Typically, IEEE 802.15.4-compliant sensor nodes compete for channel access with an increasing number of WiFi and Bluetooth devices such as laptops, smartphones and tablet PCs. This results in long contention delays and collisions which degrade sensor network performance [1], [2].

Several mitigation approaches [1]–[4] have been proposed to tackle the problem of external interference in sensor networks. Knowing the type of interference enables a sensor node to choose a suitable mitigation strategy [1], [5], [6]. In this regard, interference classification is a prerequisite step towards mitigation. Recent work on interference classification [6], [7] addresses the problem by mapping RSSI observations or patterns of corrupted packets to a known class of interference such as WiFi, Bluetooth or microwave ovens. Such designs are intrinsically constrained by a direct mapping of channel observations to a fixed number of interference classes. In particular, they do not address the predominant case of *multi-source* interference, i. e., multiple device types and instances that transmit on a channel. For example, a combination of WiFi and Bluetooth interference on a channel is likely to be reported as either WiFi or Bluetooth, depending on the dominant interferer. Furthermore, the classification approaches do not address the more difficult problem of channel selection and interference avoidance.

We present *SpeckSense*, a service that enables nodes to detect and classify multiple sources of interference in the 2.4 GHz band. In doing so, SpeckSense provides explicit recommendations on which channels are good for use. In contrast to earlier work [6], [7], SpeckSense performs an explicit interference detection step prior to classification. SpeckSense accounts for channel observations by parsing raw RSSI values as a temporal sequence of bursts. The burst sequence is fed to an unsupervised learning component, namely a clustering algorithm that detects multiple classes of RSSI bursts, each of which represents a distinct interference pattern. After this detection step, the identified classes are handed to a classification algorithm. Decoupling detection from classification thus allows for a generalized solution that identifies multiple interferers on the channel.

In addition to interference detection, SpeckSense also includes a classification algorithm that takes into account the time interval between recurring interference patterns. In this direction, SpeckSense performs two main operations—that of identifying the number of sources transmitting periodic signals (for example, WiFi access points) and that of distinguishing between different forms of data traffic (WiFi beacons, periodic and non-periodic channel traffic). We believe these two steps are crucial to selecting an interference-free channel for communication. Identifying beacons enables a sensor node to effectively avoid operating on a channel affected by WiFi interference. Towards this end, we exploit the knowledge of active WiFi zones to facilitate a blacklisting of channels that can potentially be interfered by WiFi traffic. Furthermore, the average time interval between data transfers provides a channel utilization measure that determines whether the channel is suitable for reliable communication.

We evaluate SpeckSense in an office corridor characterized by many interference sources that include several WiFi access points, a WiFi repeater, and Bluetooth data transfers. We show that SpeckSense distinguishes between the predominant sources of interference, and in particular, identifies multiple WiFi access points in the presence of data traffic with very high consistency. We demonstrate the usefulness of SpeckSense by adding it to a multichannel data collection protocol [2]. We evaluate the combined solution on a large-scale indoor testbed and observe a significant improvement in data yield facilitated by avoiding interfered channels.

In this paper we make the following contributions:

- We design and implement SpeckSense, a new approach to interference classification that decouples interferer detection from explicit classification. Our main contribution is a detector that is able to cluster multiple channel

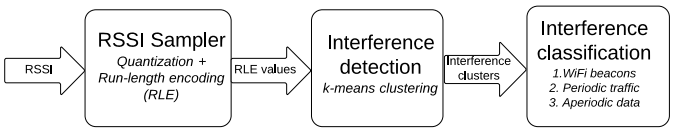


Fig. 1. SpeckSense reads RSSI values and performs a detection step prior to explicitly classifying multiple sources of interference.

activities.

- We augment interference detection with identification of periodic signals such as WiFi beacons. In addition, SpeckSense distinguishes the following predominant classes of 2.4 GHz data traffic: WiFi, Bluetooth, as well as 802.15.4 traffic from WSNs.
- We augment an existing multichannel protocol with SpeckSense to recommend channels that are unaffected by WiFi interference. We evaluate the combined solution on a large testbed comprising 85 nodes, and show how a data collection application benefits from the interference context provided by SpeckSense.

II. SpeckSense Design

Indoor environments such as offices or residential areas are witness to concurrent wireless activity across multiple standards such as WiFi, Bluetooth and IEEE 802.15.4 devices that operate in the 2.4 GHz spectrum. The resulting channel interference is therefore a combination of multiple transmissions that differ from each other in radio bit rate, message size, transmit power, channel attenuation and timing constraints [7]. As a result, their respective emissions exhibit characteristic patterns in intensity, duration, and timing. For example, emissions from a WiFi access point are distinctly different from a Bluetooth device's emissions. The central idea of SpeckSense is to disambiguate the concurrent emissions from the interferers so that the present interferers can be identified. To do so, SpeckSense accounts for collective emissions from the interferers by sampling the received signal strength (RSSI), i.e., the energy in the channel.

SpeckSense comprises two components, that perform *interference detection* and *classification* in sequence. The *interference detection* uses an *RSSI sampler* that captures the emissions from all interferers as a series of RSSI bursts. *Interference detection* involves an unsupervised learning approach, i. e., clustering, to distinguish the bursts from the different interferers. The output of the *interference detection* component is passed to a *classification* component that inspects each cluster for periodicity. Doing so enables SpeckSense to identify WiFi beacons on a given channel, as well as periodic traffic from other sources besides WiFi routers. Additionally, the classification component quantifies channel occupancy, which enables blacklisting of channels that are severely interfered.

Unlike earlier work [6], [7], SpeckSense decouples interference detection from explicit classification. This decoupling allows distinguishing the emissions from multiple interferers, and also classifying them in isolation. We now describe SpeckSense's components in more detail.

III. Interference Detection

SpeckSense's interference detection consists of an *RSSI sampler* and a *clustering process*, which are described in the following subsections.

A. RSSI Sampler

The RSSI sampler captures the energy in the channel due to the interferers' emissions, e.g., WiFi beacons or Bluetooth data packets. It continuously reads the RSSI register of the sensor nodes' radio chip. The readings are quantized, run-length encoded, and so-called bursts, i. e., contiguous sequence of high RSSI samples, are identified. The detected bursts are then processed by the clustering component.

Quantization is motivated by two observations. First, the emissions from a given interferer may vary slightly over time in their strength. These minor variations are not relevant to detecting the interferer, and hence they can be abstracted away by quantizing the RSSI reading. Second, storing raw RSSI readings is prohibitively memory-intense on a constrained sensor node. Storing quantized readings in memory is a simple means to reduce the memory requirement.

The number of quantization intervals represents a trade-off between the number of distinctly observable RSSI patterns and memory overhead. Using a higher number of intervals allows to capture more distinct channel activities, but requires more memory to store the observations. We establish power level 1 for RSSI values below -90 dBm, and divide the RSSI range above > -90 dBm evenly over the remaining number of levels. For example, using four quantization intervals would require defining the following power levels: power level 1 (RSSI ≤ -90 dBm), power level 2 (-90 dBm $<$ RSSI ≤ -60 dBm), power level 3 (-60 dBm $<$ RSSI ≤ -30 dBm), and power level 4 (-30 dBm $<$ RSSI).

The quantized RSSI readings are then run-length encoded to further reduce the memory overhead. Run-length encoding works by simply counting the number of subsequent occurrences of a power level. For example, consider the following RSSI sequence: -92, -91, -57, -58, -57, -29, -28, -59, -59, -59, -94. Quantization and run-length encoding produces the following sequence of 2D vectors: $(1,2),(3,3),(4,2),(3,3),(1,1)$. The first component of each vector denotes the power level, and the second component denotes the duration of the observation.

Finally, the RSSI sampler extracts *bursts* of activity from the quantized, run-length encoded vector sequence. A burst is defined by a contiguous subsequence where the channel is not idle, i.e., the power level is greater than 1. The RSSI sampler represents the burst by the weighted mean power level and the total duration of the subsequence. The previous example contains the non-idle subsequence $(3,3),(4,2),(3,3)$, which corresponds to the RSSI burst: $(\frac{3\times3+4\times2+3\times3}{3+2+3}, 3+2+3) = (3.25, 8)$.

SpeckSense's interference classification relies on the temporal patterns of an interferer's emissions, so it is important that processing a sample on a sensor node takes a constant amount of time. Otherwise, the duration value in an RSSI burst would

be misleading. In our implementation, processing an RSSI sample (reading it, quantizing it, and performing run-length encoding) takes 47 µs on average, giving a sampling rate of 21 KHz. This allows the detection of energy levels from WiFi beacons and Bluetooth data packets that have transmission times several magnitudes higher than 47 µs [7], [8]. More crucially, the variance in the processing delay is 0.04 µs, which is low enough to assume practically constant sampling speed. As per the suggestions by Boano et al., the RSSI sampler is implemented to avoid saturation in the radio transceiver's automatic gain control [9].

B. Clustering Algorithm

The clustering component groups together RSSI bursts that are likely to come from the same interferer. In a later step, the clusters can then be analyzed independently from each other to classify the interferer.

Prior to clustering, the RSSI bursts are normalized. Note that the mean power level of a burst can be at most 4, whereas the duration of a burst can take much larger values. Thus, normalization is required to avoid burst duration having a dominating influence on the clustering. Considering that the emissions could take 10 ms (microwave oven emissions), we scale up the average power level for all bursts by a factor of 16.

SpeckSense uses the k-means algorithm to group a set of normalized RSSI bursts B into clusters. k-means clustering is a general algorithm to group a set of observations into clusters such that similar observations belong to the same cluster [10]. We briefly describe the algorithm's operation.

Assume the bursts in B are to be grouped into k clusters. The cluster i is represented by a 2D vector μ_i called its cluster center. The vector's first component represents the average power level of bursts in the cluster, and the second component represents the average duration. Initially, the k cluster centers are chosen at random from the RSSI bursts in B. Then, the algorithm repeatedly assigns RSSI bursts to clusters and updates cluster centers until a termination condition is met.

Cluster assignment: Each RSSI burst is assigned to the cluster that has the closest center. More specifically, an RSSI burst $b_i \in B$ is assigned to the cluster j whose center has the minimal Euclidean distance to b_i. We denote the cluster center to which b_i is assigned by $m(b_i)$, defined as $m(b_i) = \operatorname{argmin}_{\mu_j} \|b_i - \mu_j\|$.

Cluster center update: After the cluster assignment, the cluster centers are recomputed. Let M_j be the set of bursts that were assigned to the jth cluster in the preceding step. Then, the cluster center μ_j is updated to be the average of all bursts in M_j. Specifically, $\mu_j = \frac{1}{|M_j|} \sum_{b \in M_j} b$.

Termination: The preceding two steps are repeated until a cost function (which is evaluated after each update step) converges, i. e., decreases by less than a fixed threshold. The cost function C describes how close the bursts are to the centers of their assigned clusters, and thus intuitively reflects the quality of the clustering: $C = \frac{1}{|B|} \sum_{b_i \in B} \|b_i - m(b_i)\|^2$. We have empirically found that a threshold of 0.001 gives good clustering performance.

The described algorithm groups the RSSI bursts into k clusters. However, the number of clusters k, which is related to the number of interferers, is not known a priori. Therefore, SpeckSense iteratively executes the algorithm for different values of k. Starting from $k = 1$, the cost function at termination is noted and k is increased by one. When the difference in cost at termination for k and $k + 1$ is less than 0.001, the algorithm terminates.

In summary, the clustering component arranges the RSSI bursts into groups such that bursts that are similar in duration and power level are assigned to the same group. The underlying intuition is that similar bursts are likely to come from the same interferer. The clustering component outputs the number of clusters k that yielded the best clustering, the center clusters $\mu_1, \ldots, \mu_k$, and which burst was assigned to which cluster.

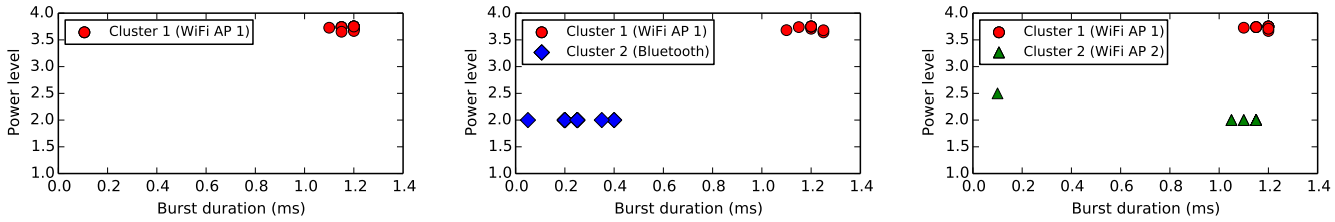
To validate SpeckSense's ability to cluster different interference patterns, controlled experiments were performed in an anechoic chamber. Figures 2(a), 2(b) and 2(c) show the different clusters detected by SpeckSense in a set of artificially induced interference scenarios. The specific cases comprise beacons from a WiFi Access Point AP1, a combination of WiFi beacons from AP1 and Bluetooth traffic between a pair of devices, and beacons from two WiFi access points AP1 and AP2. Each point in the figures represents a RSSI burst, and bursts belonging to a cluster have the same marker. The figures show that it is possible to disambiguate between different emissions based on average burst size (Figure 2(b)), as well as power level (Figure 2(c)).

Note that emissions from different sources may overlap in time, for example, microwave emissions overlapping with Bluetooth bursts. In such cases, the clustering algorithm detects only the *dominant* interferer (i. e., the microwave). SpeckSense addresses this concern by observing RSSI values over a longer duration (i. e., one second), thereby increasing the likelihood of detecting multiple interference sources.

IV. Evaluating Interference Detection

This section presents the preliminary results of SpeckSense's multi-source interference detection in a controlled RF setting. The underlying objective is to verify that SpeckSense distinguishes between the emissions from multiple interferers and that the clustering component correctly identifies one cluster of RSSI bursts per interferer. The experiments are carried out in an anechoic chamber to avoid external interference and multipath fading. Running experiments in the anechoic chamber allows a complete control which interferers are present, which is necessary to assess the validity of SpeckSense's output.

The experimental setup comprises five TelosB nodes, two WiFi access points, a laptop and a smartphone. The WiFi access points are used to create WiFi channel activity. The laptop and the smartphone are used to create Bluetooth channel activity. The sensor nodes record the interferers' emissions using the RSSI sampler as described in Sec. III-A. The



(a) Only WiFi AP 1 active

(b) WiFi AP 1 and Bluetooth active

(c) WiFi AP1 and AP2 active

Fig. 2. Clusters detected by SpeckSense in the anechoic chamber for different interference scenarios. Each marker represents an RSSI burst, and the marker's shape and color indicate which cluster the burst was assigned to. The number of clusters found by SpeckSense corresponds to the number of interferers.

RSSI bursts are then grouped into clusters by the clustering component described in Sec. III-B.

Multiple experiments were performed with different combinations of interferers, and each experiment was repeated five times. The sensor nodes were programmed to listen on the 802.15.4 channels 11, 13, 15, and 17, which overlapped with both the WiFi and Bluetooth interferers' channels. The following describes three representative runs of these experiments.

Experiment 1: Only one WiFi Access Point. In this experiment, only WiFi access point 1 (AP 1) was active and sent beacons, while the sensor nodes were running SpeckSense. The output of the clustering component for this experiment is shown in Fig 2(a). Each marker represents an RSSI burst. The marker's color and shape indicate which cluster the burst was assigned to. The figure shows that SpeckSense correctly identifies that there is one cluster and thus assigns all RSSI bursts to the cluster. As no other interferer was active during this run, it can be asserted that the bursts in the cluster correspond to channel activity from AP 1.

Experiment 2: Concurrent WiFi and Bluetooth Interference. In the next experiment, the smartphone transferred a 20 MB file to the laptop via Bluetooth, while the WiFi access point AP 1 was active.

Figure 2(b) shows the output of the clustering component. The RSSI bursts are grouped into two clusters. The first cluster is similar to the cluster detected in the previously described experiment. Thus, it is safe to conclude that it represents the emissions from AP 1. The bursts in the second cluster have a lower power level. These are likely to come from the Bluetooth devices. Each cluster thus maps to a specific type of channel traffic.

SpeckSense's interference detection can distinguish the emissions from Bluetooth and WiFi because they differ in both power level and duration. Bluetooth emissions are weaker in signal strength and shorter in duration than the emissions from the WiFi access point. The following experiment considers the case in which the interferers' emissions are similar in burst duration.

Experiment 3: Two WiFi Access Points. In this experiment both WiFi access points were active. This case represents a common scenario in indoor environments, where multiple WiFi access points are visible.

The detected RSSI bursts and cluster assignments are shown in Fig. 2(c). SpeckSense detects two distinct clusters. Cluster 1 is similar to the cluster detected in the first experiment, and therefore the RSSI bursts in the cluster are likely to represent channel activity from AP 1. The second cluster contains channel activity from AP 2. While most RSSI bursts in the second cluster have a duration of around 1.1 ms, note that there is an outlier with much shorter duration ($\approx$ 0.1 ms). Manual inspection of this burst revealed that it arose due to an artifact in sampling: the emission from a WiFi access point was only partially captured, because it was ongoing when the sensor nodes started recording channel activity.

Even though emissions from both access points are similar in burst duration (since they both represent beacons), SpeckSense distinguishes the emissions from the interferers due to their difference in power level.

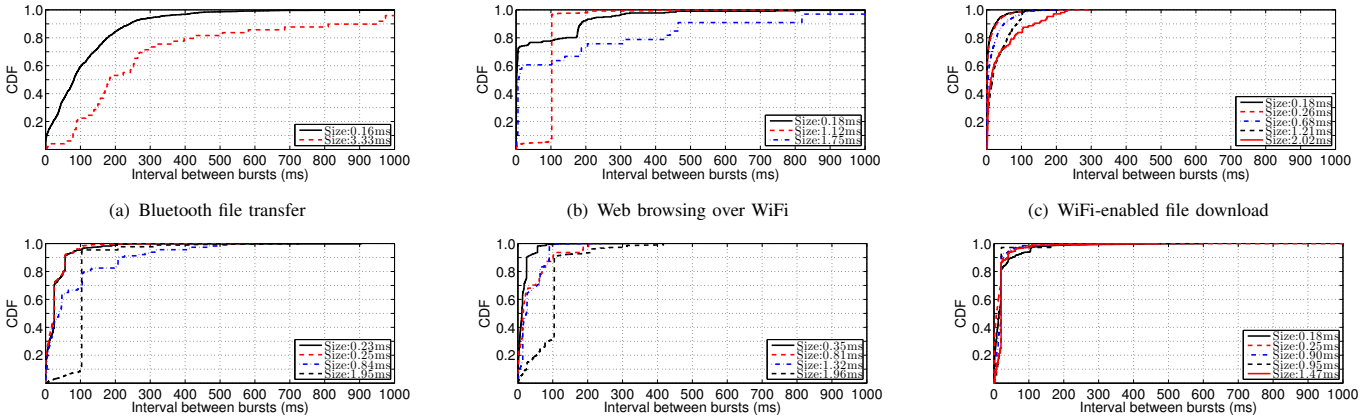
The experiments in the anechoic chamber show that it is possible to detect and distinguish multiple types of interference at the physical level. The following section presents a detailed classification scheme of SpeckSense that is based on the temporal patterns of channel activity exhibited by different interference clusters. The classification algorithm of SpeckSense can identify periodic channel traffic, and also distinguish bursty traffic from sporadic channel activity.

V. Interference Classification

SpeckSense classifies interference by inspecting each detected cluster for temporal patterns in RSSI bursts. In doing so, SpeckSense informs link-layer protocols whether the observed channel activity is periodic, bursty or a combination of both. This facilitates a meaningful assessment of channel quality and enables nodes to make informed decisions on channel selection. In this regard, SpeckSense deviates from earlier classification work such as SoNIC [6] that maps channel observations to specific labels such as WiFi, Bluetooth and microwave. This section elaborates on two aspects of interference classification, namely distinguishing different extremes of prevalent 2.4 GHz data traffic and identifying periodic signals such as WiFi beacons.

A. Distinguishing Channel Traffic

Interference in the 2.4 GHz spectrum is largely attributed to concurrent traffic over WiFi and Bluetooth, as well as electromagnetic emissions from microwave ovens. The impact



(a) Bluetooth file transfer

(b) Web browsing over WiFi

(c) WiFi-enabled file download

(d) WiFi-enabled video streaming

(e) WiFi repeater traffic

(f) Microwave

Fig. 3. Empirical CDFs of the inter-burst separations per detected cluster, for different interference scenarios. SpeckSense distinguishes between different extremes of channel traffic, using a 100 ms threshold on the observed average inter-burst separation. SpeckSense can also identify periodic emissions in the form of WiFi beacons (Figures 3(b), 3(d) and 3(e), as well as microwave bursts (Figure 3(f).

from channel interference on a wireless network application is determined by several factors such such as device usage patterns, application data requests as well as the underlying communication protocol. Therefore, it is reasonable to expect that certain applications contribute to a greater degree towards channel interference than others – for example, a file download over WiFi causes more channel interference than web browsing. SpeckSense distinguishes between diverse applications at the physical layer based on their characteristic contribution to channel traffic. Specifically, SpeckSense computes the average inter-burst separation for each interference cluster, and checks whether it is below a predetermined threshold. If so, the channel is said to be severely interfered and hence blacklisted for sensor network operation.

To empirically determine the threshold inter-burst separation, we conduct experiments involving controlled interference, in which SpeckSense gathers RSSI samples for different scenarios that included a Bluetooth file transfer, WiFi file download, WiFi web browsing, video streaming over WiFi, WiFi repeater traffic, and microwave oven emissions. Figure 3 shows the cumulative distribution of the inter-burst separation for different clusters for the aforementioned cases. We observe that for cases where bursty traffic is involved, such as in Figures 3(c), and 3(e), 80% of the inter-burst separations are below 100 ms. Note that channel activity bursts owing to Bluetooth transfers and WiFi-enabled web browsing are not as frequent as WiFi file download and repeater traffic. This is attributed to factors such as Bluetooth frequency hopping that effectively schedules packet transmissions over non-overlapping channels, as well as temporally sparse patterns in web browsing. Further, a reduced average inter-burst separation is correlated to an increase in the number of detected clusters. Table I lists the mean inter-burst separation values, showing that channel activity bursts owing to Bluetooth transfers and WiFi-enabled web browsing are not as frequent as WiFi file download and video streaming.

Based on these observations, SpeckSense uses an average inter-burst separation threshold of 100 ms, which has shown good results in distinguishing conditions of light channel traffic (cf. Figures 3(a), and 3(b)) from severe interference (cf. Figures 3(e), and 3(f)).

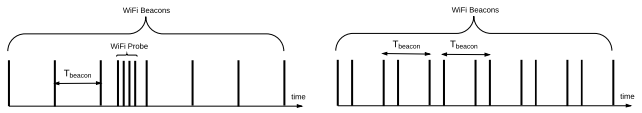
B. Identifying Periodic Beacons

Concurrent traffic over WiFi constitutes a major part of cross-technology interference in the 2.4 GHz ISM band [1]. Therefore it is necessary that a sensor node avoids operating on channels that overlap with WiFi activity. While usage patterns of WiFi may vary over time depending on varying user needs, there is a stable pattern in control signaling on the WiFi channels. Predominant IEEE 802.11 control messages include WiFi beacons, probe responses from access points, and probe requests from WiFi clients. Particularly, beacon messages are sent at a default periodic interval of 100 ms. Identifying them can thus be regarded as an indication of WiFi presence. Towards this end, SpeckSense uses the results from its multi-source interference detector, and classifies a clustered sequence of periodically recurring RSSI bursts as WiFi beacons. This is, however, a non-trivial problem and entails addressing the following challenges. WiFi control messages such as probe requests and probe responses may have similar on-air transmission times as beacons, and are also transmitted over non-periodic intervals (see Figure 4(a)). Moreover, beacons from multiple WiFi access points within interference range may have similar on-air transmission times and RSSI values (see Figure 4(b)), and get clustered together. The random occurrences of WiFi probes and beacons from multiple APs collectively represent a challenge in identifying periodic patterns.

Accounting for these challenges, SpeckSense employs an algorithm (see Algorithm 1) that is run once for each cluster obtained from the interference detection outlined in Section III-B. In every run, the input to the algorithm is a temporal sequence of RSSI bursts from a cluster. Let t_i denote the time

	Bluetooth file transfer	WiFi file transfer	WiFi web browsing	WiFi video streaming	WiFi repeater traffic	Microwave
Avg. inter-burst separation (ms)	253	23	146	63	50	32
Avg. number of clusters	1.4	3.6	2.5	3.5	5	5

TABLE I
SPECKSENSE DISTINGUISHES BETWEEN DIFFERENT FORMS OF CHANNEL ACTIVITY, BASED ON THE AVERAGE INTER-BURST SEPARATION AND THE NUMBER OF DETECTED INTERFERENCE CLUSTERS.



(a) Periodic WiFi beacons interspersed by probe messages.

(b) Periodic WiFi beacons from two access points

Fig. 4. WiFi beacons may be interspersed by probe messages or beacons from other access points, making their identification non-trivial.

Algorithm 1 Algorithm to detect periodic bursts

Inputs
▷ n is the number of RSSI bursts over time T
▷ $d_T = (d_t^1, d_t^2 \ldots d_t^{n-1})$ is the sequence of inter-burst separations
Outputs
▷ $P(d_\tau)$ is the confidence value for every $d_\tau \in L$
▷ t_p is the detected periodicity of the sequence

$L \leftarrow \emptyset$
for $d_t^i \in d_T$ ADDTOSET(L, d_t^i) **end for**
for $d_t^i \in (d_t^1, d_t^2 \ldots d_t^{n-1})$ **do**
    $s \leftarrow d_t^i$
    for $d_t^j \in (d_t^{i+1}, d_t^{i+2} \ldots d_t^{n-1})$ **do**
        $s \leftarrow s + d_t^j$
        UPDATESET(L, s)
    end for
end for
for each $d_\tau \in L$ **do**
    $n_\tau \leftarrow \lfloor \frac{T}{d_\tau} \rfloor$
    $P(d_\tau) = 2C(d_\tau)/(n_\tau(n_\tau + 1))$
end for
$t_p = \text{argmax}_{d_\tau} P(d_\tau)$

at which the ith burst in the cluster was recorded by the node, where $i \leq i \leq n$. The inter-burst separation is denoted by the sequence $d_T = (t_1 - t_0, t_2 - t_1, \ldots, t_n - t_{n-1})$.

The algorithm populates a set L with values denoting time periods at which RSSI bursts are captured. This is performed by inspecting every inter-burst separation value in the sequence d_T, and checking to see whether they are already included in the set L (Procedures 2, line 2 in *AddToSet*). Specifically, the check takes the form of a modulus operation, such that an inter-burst separation of kd_τ is not added to L, if d_τ has already been included. The modulo operation allows a certain variance ϵ_δ to account for factors such as clock speed variations of the node recording RSSI, as well as channel backoffs by the interfering source. Setting ϵ_δ to 7 RSSI sampling intervals allows a jitter of $2\epsilon_\delta \approx 0.4$ ms, which we have found to empirically give good results.

After populating L, the algorithm maps every $d_\tau \in L$ to a counter value $C(d_\tau)$. $C(d_\tau)$ is a measure of how periodic the RSSI sequence is in d_τ. Intuitively, the algorithm checks over a time window T, whether there are RSSI bursts at times $d_\tau, 2d_\tau, 3d_\tau \ldots kd_\tau$, where $k = \lfloor \frac{T}{d_\tau} \rfloor$. Since the entries in L are determined from d_T, this step is performed by scanning every value $d_t^i \in d_T$ in sequence. For every d_t^i, the algorithm adds the inter-burst separations from d_t^{i+1} to d_t^{n-1}, and checks at each step, whether the partial sum is periodic in any $d_\tau \in L$ (Procedures 2, line 8 in *UpdateSet*). If not, the sum is added to the list, and its count is set to 1 (Procedures 2, lines 11–12 in *UpdateSet*). In general, if n_τ denotes the number of RSSI bursts that are periodic in d_τ over time T, then $n_\tau = \lfloor \frac{T}{d_\tau} \rfloor$. This results in a maximum of $\frac{1}{2}n_\tau(n_\tau + 1)$ summations that are periodic in d_τ, or equivalently, $C(d_\tau) \leq \frac{1}{2}n_\tau(n_\tau + 1)$. Therefore, the fraction $P(d_\tau) = 2C(d_\tau)/(n_\tau(n_\tau + 1))$ represents a normalized confidence measure for periodicity in d_τ. Possible values for $P(d_\tau)$ range from 0 and can also exceed 1, especially when multiple RSSI bursts occur with the same periodicity, as in Figure 4(b). The periodicity check in *UpdateSet* is allowed a greater threshold, i. e., $\epsilon_\Delta > \epsilon_\delta$, in order to to account for accumulated variance over summing up inter-burst separations. We find that setting ϵ_Δ to 30 RSSI sampling intervals, or approximately 1.5 ms, gives good results. SpeckSense uses $\text{round}(P(d_\tau))$ as a measure for the number of different RSSI subsequences that are periodic in d_τ.

The period t_p of the RSSI sequence is determined to be $\text{argmax}_{d_\tau} P(d_\tau)$, with the additional constraint, $\text{round}(P(d_\tau)) \geq 1$. The value of t_p is approximately 100 ms for WiFi beacons, which is the default beaconing interval on most WiFi access points. Algorithm 1, however, is also generally applicable to detect RSSI bursts of any period, in contrast to other approaches [8], [11] that explicitly check for predetermined values. This makes it a viable option to detect and classify other forms of interference that include periodic transmissions in 802.15.4 networks [12] as well as microwave bursts.

VI. IMPLEMENTATION

Our implementation of SpeckSense is based on the Contiki operating system and the TMote Sky hardware featuring a CC2420 radio transceiver. There are, however, no special features that prevent porting SpeckSense to other sensor node hardware platforms that allow fast RSSI sampling.

The code for SpeckSense including the Contiki operating system fits within 21 KB of program memory, and the overall RAM usage is contained within 6 KB. Of these, the clustering algorithm, which forms a major part of SpeckSense takes only

Procedures 2 Updating entries in candidate set L

```
1: procedure AddToSet(L, d_t)
2:     if ∀d_τ ∈ L, d_t (mod d_τ) ∈ (ε_δ, d_τ − ε_δ) then
3:         L ← L ∪ d_t
4:         C(d_t) ← 0
5:     end if
6: end procedure
7: procedure UpdateSet(L, d_t)
8:     if ∃d_τ ∈ L|d_t (mod d_τ) ∉ (ε_Δ, d_τ − ε_Δ) then
9:         C(d_τ) ← C(d_τ) + 1
10:    else
11:        L ← L ∪ d_t
12:        C(d_t) ← 1
13:    end if
14: end procedure
```

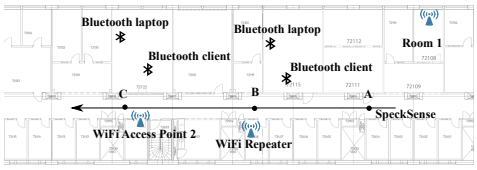


Fig. 5. Experimental setup in the office corridor. We evaluate SpeckSense at locations A, B and C in the presence of WiFi and Bluetooth interference.

about 4 KB of program memory and a total of less than 800 bytes of RAM.

VII. Evaluation

We implement SpeckSense on the Tmote Sky hardware featuring a CC2420 radio transceiver. There are, however, no special features that prevent porting SpeckSense to other sensor node hardware platforms that allow fast RSSI sampling. The code for SpeckSense is implemented using the Contiki operating system and fits within 21 KB of program memory. The overall RAM usage is contained within 6 KB, of which the clustering algorithm takes only about 4 KB of program memory and a total of less than 800 bytes of RAM.

We evaluate SpeckSense's ability to distinguish between multiple sources of interfering traffic, and its ability to identify the presence of WiFi access points in the 2.4 GHz band. We conduct our experiments in two indoor environments: an office corridor and a 85-node indoor testbed that spans three floors. These environments represent challenging conditions for SpeckSense because they induce strong multipath fading. We present our results in the following order. First, we showcase the multi-source interference detection results of SpeckSense from the office corridor. Then, we show how SpeckSense improves the data gathering performance of a multichannel protocol [2] on a 85-node testbed.

A. Detecting Concurrent Interferers

Indoor environments represent challenging conditions for SpeckSense due to non-line of sight between nodes that causes multipath fading effects. The extent of these effects may also vary over time, e.g., due to people moving, thereby increasing the variance in received signal strength on a sensor node. SpeckSense relies on RSSI observations to detect interference, so it is important to characterize its performance in such an environment.

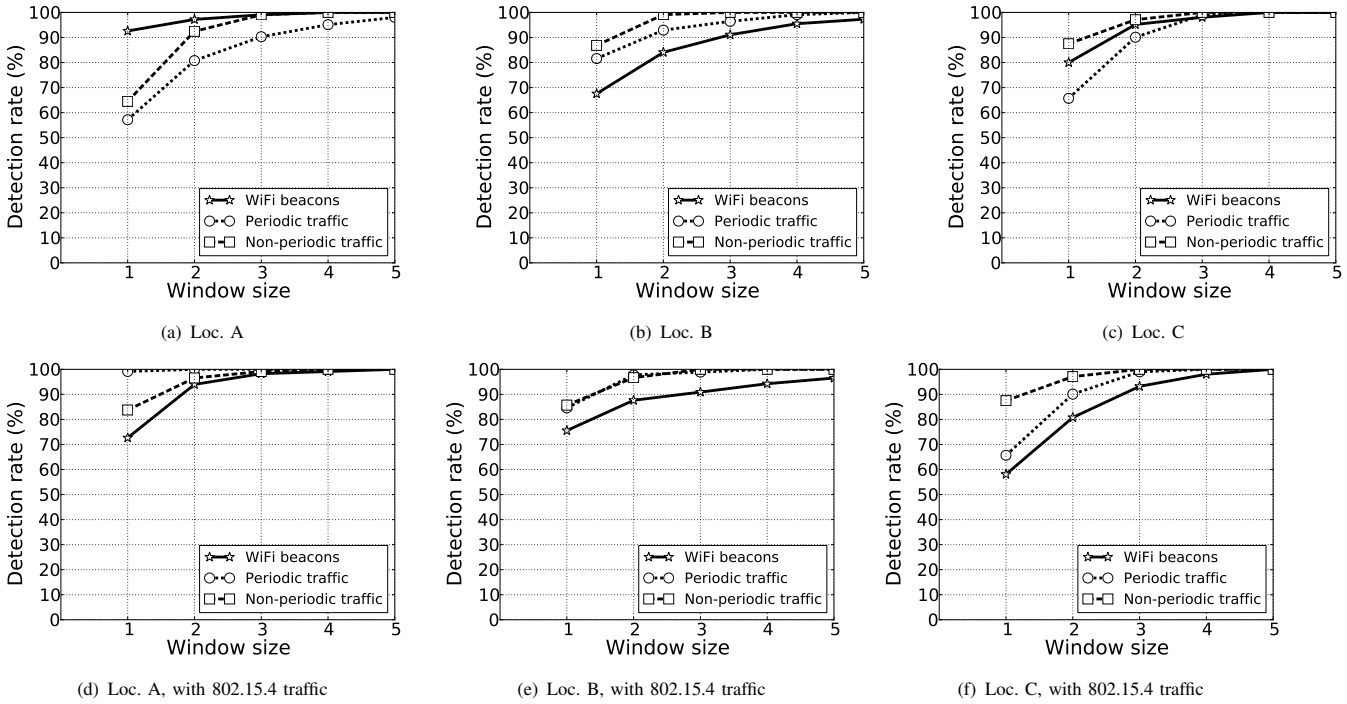
Experimental Setup. The setup in the office corridor is shown in Fig. 5. There are two WiFi access points (operating on WiFi channel 1 and 11, respectively) a WiFi repeater (operating on channel 1), as well as four Bluetooth devices. Sensor nodes run SpeckSense at locations A, B and C. Nodes at location A face interference from WiFi AP 1 and the WiFi repeater, as well as sporadic Bluetooth interference. Nodes at location B operate on a different channel and are exposed to Bluetooth interference as well as beacons from WiFi AP 2. Nodes at location C face interference from Bluetooth and WiFi data transfers.

We perform over 100 experimental runs in sequence. In each run, nodes perform RSSI sampling for 1 second, followed by interference detection and classification. The RSSI sampler uses four power levels to quantize signal strength information, as described in Sec. III-A. Each detected interference cluster is classified as follows: *(i)* WiFi beacons that have a period of 100 ms, *(ii)* periodic traffic and *(iii)* non-periodic traffic. To quantify SpeckSense's performance, we define a *detection rate* for every interference class. The *detection rate* for an interference class is measured as the percentage number of runs in which SpeckSense identifies it.

Data traffic from IEEE 802.15.4 compliant sensor nodes also contributes to co-channel interference in the 2.4 GHz spectrum. To validate that SpeckSense can classify multiple interferers even in the presence of WSN activity, we perform our experiments under two scenarios, namely with and without 802.15.4 traffic. To generate the channel traffic, we add two sensor nodes to the setup – one node sends packets every 125 ms, while the other receives them. In every setup, the sender node is co-located with the node running SpeckSense, and the receiver node is placed 6 m away from the sender. We refer to these nodes as the 802.15.4 sender and the 802.15.4 receiver.

Results. Figure 6 shows the detection rates for SpeckSense at different locations, both in the presence and absence of 802.15.4 traffic. Accounting for multipath fading effects that inhibit a seamless classification, we aggregate the detection rates over a window representing a sequence of runs. An interference class is detected when it is observed at least once over the window. The plots show the detection rate of SpeckSense for different window sizes. SpeckSense achieves a detection rate of over 90% in all cases when using a window size of 3 or greater. Depending upon the specific interference context described in the experimental setup, *non-periodic* and *periodic* traffic relate to different sources of channel activity. For example, *periodic traffic* in Figures 6(a), 6(b), and 6(c) represents periodic TCP bursts in WiFi data transfers. In contrast, *periodic traffic* in Figures 6(d), 6(e), and 6(f) also comprises additional 802.15.4 traffic, which has a period of 125 ms. *Non-periodic traffic* at location A relates to WiFi data transfers, and at locations B and C, relates to a combination of WiFi and Bluetooth data traffic.

Channel activity in the office corridor also includes beacons from additional WiFi APs outside of our control, such as the university's WiFi. Table II shows the 50th and 90th percentile



(a) Loc. A (b) Loc. B (c) Loc. C

(d) Loc. A, with 802.15.4 traffic (e) Loc. B, with 802.15.4 traffic (f) Loc. C, with 802.15.4 traffic

Fig. 6. Detection rates for the three locations in the office corridor. For window sizes of three and larger, SpeckSense's detection rate exceeds 90%.

802.15.4 traffic	Number of detected WiFi access points (percentile)					
	Location A		Location B		Location C	
	50th	90th	50th	90th	50th	90th
No	3	4	1.5	4	1	3
Yes	1	3	2	4	1	2

TABLE II
50TH AND 90TH PERCENTILE OF WIFI ACCESS POINTS IDENTIFIED BY SPECKSENSE AT DIFFERENT LOCATIONS. THE VALUES INDICATE THAT SPECKSENSE CAN DETECT WIFI ACTIVITY EVEN IN THE PRESENCE OF AMBIENT 802.15.4 TRAFFIC.

of WiFi access points that SpeckSense identifies at different locations. In general, SpeckSense identifies fewer access points in the presence of 802.15.4 traffic. We attribute this to an artifact of our experimental setup – the periodic 802.15.4 acknowledgement frames from the 802.15.4 receiver have burst durations similar to WiFi beacons. SpeckSense therefore detects a cluster that has multiple, yet distinct periods, which our approach (see algorithm 1) does not handle at present. We plan to address this issue in future work. Nonetheless, the results show that SpeckSense identifies multiple access points, even in the presence of Bluetooth and 802.15.4 traffic.

B. Improving Data Collection Performance

Data collection applications for indoor WSN deployments suffer from degraded performance on account of WiFi interference. To mitigate the effects of external interference, multichannel protocols [2] coordinate node communication on different radio channels. These approaches achieve resilience against interference by either hopping through a fixed sequence of channels [13], or by switching channels when interfered [2]. However, they do not address the problem of finding a relatively interference-free channel.

As a solution, we run SpeckSense independently on every node to perform a deployment-time assessment of WiFi-free radio channels. We evaluate SpeckSense as a link-layer service for Chrysso [2], a multichannel protocol that adaptively switches radio channels on interfered nodes. Sensor nodes independently run SpeckSense at network bootstrap and blacklist channels in which SpeckSense detects WiFi beacons or interfering channel activity with an average inter-burst separation less than 100 ms.

We compare SpeckSense's results against three other strategies that differ on channel selection policy, namely Chrysso *default*, Chrysso *best channels*, and Chrysso *threshold*. Chrysso *default* employs a random channel selection scheme over all 16 channels, whereas Chrysso *best channels* performs a random selection over a restricted set of channels, namely 15, 20, 25 and 26. The channels are chosen such that they empirically exhibit the best packet reception rates among all other channels on the testbed [14]. they do not overlap with commonly used WiFi channels 1, 6 and 11. Chrysso *threshold* is closest in design and objective to SpeckSense on interference avoidance, and ranks channels based on their utilization. The channel utilization is simply computed as a ratio of the number of channel *idle* RSSI samples (RSSI $\leq$ -90 dBm) over the total number of RSSI samples, as suggested by Musăloiu-E. et al. [15]. In our implementation, Chrysso *threshold* uses the best four channels in decreasing order of channel utilization.

We experimentally evaluate the aforestated strategies on

	Data collection performance		
Protocol	Data yield	Duty cycle	Energy per delivered packet
Chrysso *default*	73.3 %	2.9 %	4.22 mJ
Chrysso *best channels*	95.3 %	2.3 %	2.6 mJ
Chrysso + *threshold*	91.4 %	2.4 %	3.1 mJ
Chrysso + SpeckSense	94.8 %	2.3 %	2.9 mJ

TABLE III
DATA COLLECTION PERFORMANCE (AVERAGED OVER SIX RUNS) ON A 85-NODE TESTBED, HIGHLIGHTING THE ADVANTAGES DERIVED FROM INTERFERENCE AVOIDANCE. SPECKSENSE WITH CHRYSSO OUTPERFORMS THE CONVENTIONAL RSSI THRESHOLDING TECHNIQUE ON AVOIDING INTERFERED CHANNELS.

the Indriya WSN testbed [16], using a network of 85 nodes including the sink. Every node generates one packet per minute over a two-hour duration, and duty cycles its radio wakeup over an interval of 125 ms, using the X-MAC protocol [17]. We perform six experimental runs for each variant of Chrysso described above.

Table III contrasts data collection performance of the revised Chrysso variants against its original implementation, Chrysso *default*. In general, avoiding interfered channels improves both the average data yield and the energy per transmitted packet for Chrysso. Specifically, running SpeckSense with Chrysso increases the average data yield (packets received by the sink) by 30% over Chrysso *default*. This improvement is mainly attributed to avoidance of WiFi-interfered channels by SpeckSense. To validate our claim, we find that SpeckSense blacklists 802.15.4 radio channels that overlap with commonly used WiFi channels 1, 6 and 11, in more than 80% of the nodes. For the same reason, Chrysso SpeckSense performs comparably with Chrysso *best channels* that explicitly avoids the aforesaid WiFi channels. The 95% confidence intervals for both Chrysso SpeckSense and Chrysso *best channels* overlap on all three performance metrics. The overlap indicates that neither variant outperforms the other, in accordance with rules of analysis in [18]. However, SpeckSense presents a more general solution that applies to indoor environments wherein co-located WiFi networks may operate on channels other than 1, 6 and 11. Lastly, SpeckSense outperforms *rssi threshold* on average data yield and duty cycle. This suggests that for the same energy cost in RSSI sampling (334.6 mJ on average per node), SpeckSense is more effective at avoiding WiFi-interfered channels than a simple approach that computes channel utilization using a threshold. In conclusion, the results show that existing multichannel protocols such as Chrysso benefit from the interference classification output provided by SpeckSense.

VIII. RELATED WORK

As the number of wireless devices operating in the license-free frequency bands is steadily increasing, the problem of interference is receiving more attention. A few other approaches are similar to ours in that they sample the RSSI. Zacharias et al. [7] classify interference based on a fixed set of simple conditions. In contrast to SpeckSense, their classification includes processing of computationally expensive tasks such as FFTs and execution on a PC rather than on motes. Also Boers et al. [19] sample the spectrum for interferer classification but they only target interference occurring at regular intervals. Likewise, Zhou et al. [8], [11] propose an algorithm that is restricted to detecting WiFi beacons from RSSI traces. Another approach based on spectrum sampling is by Bloessl et al. [20]. In contrast to SpeckSense, their approach is limited to the detection of single interference sources. Ansari et al. [21] propose an approach to detect WiFi networks by using a synchronized pair of nodes to scan adjacent channels. In contrast, SpeckSense bases its observations of multiple interferers on a single node. Rayanchu et al. [22] detect WiFi access points and other non-WiFi devices using commodity WiFi hardware. However, their approach relies on device-specific WiFi features and involves computationally intensive processing, making it infeasible for resource-constrained sensor nodes. Hermans et al. [6] present SoNIC interference classification without spectrum sampling relying only on the information provided by corrupted packets. As their approach does not rely on spectrum sampling it is less energy-consuming than SpeckSense but it does not provide higher level information such as the number of WiFi access points.

There are efforts for channel selection that use the average energy in a channel [15], [23], [24], or packet reception counts [25] as selection criteria. In contrast to these approaches, we take the source of interference into account.

IX. CONCLUSION

In this paper we have presented SpeckSense, a detection and classification scheme for multi-source interference for wireless sensor networks. Experiments in a real setting have shown that SpeckSense detects multiple interferers with good consistency, i. e., over 90% of the cases. We have also implemented SpeckSense as a low-layer service that recommends relatively interference-free channels to use for WSN data collection. Experimental results from running a multichannel protocol along with SpeckSense have shown a significant improvement in data yield at lower duty cycle.

REFERENCES

[1] C. M. Liang, N. B. Priyantha, J. Liu, and A. Terzis, "Surviving Wi-Fi interference in low power ZigBee networks," in *ACM SenSys*, 2010.

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

[3] J. Hauer, A. Willig, and A. Wolisz, "Mitigating the effects of RF interference through RSSI-based error recovery," in *EWSN*, 2010.

[4] C. A. Boano, T. Voigt, N. Tsiftes, L. Mottola, K. Römer, and M. A. Zúniga, "Making sensornet MAC protocols robust against interference," in *EWSN*, 2010.

[5] K. R. Chowdhury and I. F. Akyildiz, "Interferer classification, channel selection and transmission adaptation for wireless sensor networks," in *ICC*, 2009.

[6] F. Hermans, O. Rensfelt, T. Voigt, E. Ngai, L.-Å. Nordén, and P. Gunningberg, "SoNIC: classifying interference in 802.15.4 sensor networks," in *IPSN*, 2013.

[7] 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 *ICWMC*, 2012.

[8] R. Zhou, Y. Xiong, G. Xing, L. Sun, and J. Ma, "ZiFi: wireless LAN discovery via ZigBee interference signatures," in *ACM MobiCom*, 2010, pp. 49–60.

[9] C. A. Boano, T. Voigt, C. Noda, K. Romer, and M. Zúñiga, "JamLab: Augmenting sensornet testbeds with realistic and controlled interference generation," in *ACM IPSN*, 2011.

[10] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 281-297. California, USA, 1967, p. 14.

[11] Y. Gao, J. Niu, R. Zhou, and G. Xing, "Zifind: Exploiting cross-technology interference signatures for energy-efficient indoor localization," in *IEEE Infocom'13*, pp. 2940–2948.

[12] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with Glossy," in *IPSN*, 2011, pp. 73–84.

[13] L. Tang, Y. Sun, O. Gurewitz, and D. B. Johnson, "EM-MAC: a dynamic multichannel energy-efficient MAC protocol for wireless sensor networks," in *In ACM MobiHoc'11*, p. 23.

[14] B. A. Nahas, S. Duquennoy, V. Iyer, and T. Voigt, "Low-Power Listening Goes Multi-Channel," in *IEEE DCOSS*, Marina Del Rey, CA, USA, 2014.

[15] R. Musaloiu-E and A. Terzis, "Minimising the effect of WiFi interference in 802.15. 4 wireless sensor networks," *IJSN*, vol. 3, no. 1, pp. 43–54, 2008.

[16] M. Doddavenkatappa, M. C. Chan, and A. L. Ananda, "Indriya: A low-cost, 3d wireless sensor network testbed," in *TriDentCom'12*. Springer, pp. 302–316.

[17] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," in *ACM SenSys'06*, pp. 307–320.

[18] R. Jain, *The art of computer systems performance analysis*. John Wiley & Sons, 2008.

[19] N. M. Boers, I. Nikolaidis, and P. Gburzynski, "Sampling and classifying interference patterns in a wireless sensor network," *ACM TOSN'12*, vol. 9, no. 1, p. 2.

[20] B. Bloessl, S. Joerer, F. Mauroner, and F. Dressler, "Low-Cost Interferer Detection and Classification using TelosB Sensor Motes," in *ACM MobiCom*, 2012, pp. 403–406.

[21] J. Ansari, T. Ang, and P. Mähönen, "WiSpot: fast and reliable detection of Wi-Fi networks using IEEE 802.15.4 radios," in *ACM MobiWac'11*, pp. 35–44.

[22] S. Rayanchu, A. Patro, and S. Banerjee, "Catching whales and minnows using WiFiNet: deconstructing non-WiFi interference using WiFi hardware," in *Proc. of USENIX NSDI*, 2012.

[23] J. Ansari and P. Mähönen, "Channel selection in spectrum agile and cognitive MAC protocols for wireless sensor networks," in *ACM MobiWac*, 2010.

[24] C. Noda, S. Prabh, M. Alves, C. Boano, and T. Voigt, "Quantifying the channel quality for interference-aware wireless sensor networks," *ACM SIGBED Review*, vol. 8, no. 4, pp. 43–48, nov 2011.

[25] M. Doddavenkatappa, M. C. Chan, and B. Leong, "Improving link quality by exploiting channel diversity in wireless sensor networks," in *IEEE RTSS'11*, pp. 159–169.