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

Venkatraman Iyer[1], Frederik Hermans[1], and Thiemo Voigt[1,2]

[1] Uppsala University, Sweden
{venkatraman.iyer, frederik.hermans}@it.uu.se
[2] SICS Swedish ICT, Sweden
thiemo@sics.se

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 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 evaluate SpeckSense on common mote hardware and show how it classifies concurrent interference under real-world settings. We also show how SpeckSense improves the performance of an existing multichannel data collection protocol by 30%.

1 Introduction

Low-power wireless sensor networks (WSN) operating in the 2.4 GHz spectrum often face interference from other wireless technologies 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 that 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 prerequisite 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. In this regard, the detection of multiple interfering sources offers interesting insights on channel utilization. The number of distinct interfering sources on a channel has a marked influence on its utilization – for example, concurrent traffic over WiFi and Bluetooth traffic has a greater interference impact than either in isolation. Moreover, interfering channel traffic from multiple sources can be independently inspected for temporal patterns such as periodicity. This enables a wireless device to identify periodic control signals on an active WiFi channel, and blacklist it for sensor network operation. Lastly, multiple interference detection enables wireless devices to disambiguate external interference from in-network channel traffic. This provides a clearer context for motivating interference mitigation mechanisms as in [1, 2].

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 [6,8], SpeckSense performs an explicit interference detection step prior to classification. The detection step uses RSSI values to account for channel observations, and clusters them based on pre-determined RSSI intervals in which they belong and also the time duration for which a sequence of similar RSSI values persist. Each cluster thus represents a distinct interference pattern, which is handed to a classification algorithm.

SpeckSense is primarily designed for avoiding WiFi and other forms of severe interference in indoor WSN deployments. To this end, SpeckSense performs two main operations — distinguishing between different forms of data traffic (WiFi beacons, periodic and non-periodic channel traffic) and identifying the number of sources transmitting periodic signals – for example, WiFi access points. SpeckSense uses the average time interval between recurring RSSI patterns to distinguish between conditions of moderate (web browsing) and intense (bulk data transfer) channel traffic. In doing so, SpeckSense provides a channel utilization measure that determines whether the channel is suitable for reliable communication. Furthermore, identifying beacons enables a sensor node to effectively blacklist channels affected by WiFi interference.

We evaluate SpeckSense in an office corridor characterized by many interference sources that include several WiFi and Bluetooth-enabled devices. 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. We demonstrate the usefulness of SpeckSense by adding it to a multi-channel 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 develop SpeckSense, a new approach for detecting and classifying *multiple concurrent sources of interference* in the 2.4 GHz spectrum.

– We facilitate interference avoidance by distinguishing between different extremes of channel traffic (web browsing vs. file transfers), and identifying periodic WiFi beacons.
– We show how an existing data collection protocol can benefit from using SpeckSense to recommend WiFi-free channels. Our experimental evaluation on a large testbed comprising 85 nodes shows a 30% improvement in data yield when using SpeckSense.

2 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 [8]. 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, 8], 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.

3 Interference Detection

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

3.1 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 [8, 9]. 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 [10].

3.2 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 [11]. 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.

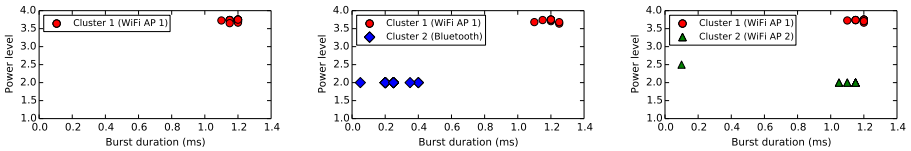
Cluster assignment Each RSSI burst is assigned to the cluster that has the closest center. More specifically, an RSSI burst $b_i \in B$ is assigned to the cluster j whose center has the minimal Euclidean distance to b_i. We denote the cluster center to which b_i is assigned by $m(b_i)$, defined as $m(b_i)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

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



(a) Only WiFi AP 1 active (b) WiFi AP 1 and Bluetooth active (c) WiFi AP1 and AP2 active

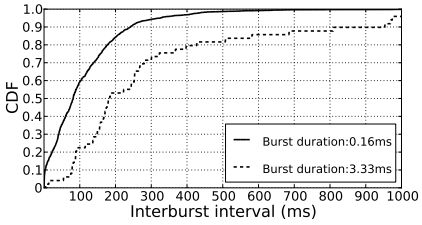
Fig. 1. Clusters detected by SpeckSense in the anechoic chamber for different interference scenarios. Each marker represents an RSSI burst, and the marker's shape indicates which cluster the burst was assigned to. The number of clusters found by SpeckSense corresponds to the number of interferers.

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 1(a), 1(b) and 1(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 1(b)), as well as power level (Figure 1(c)).

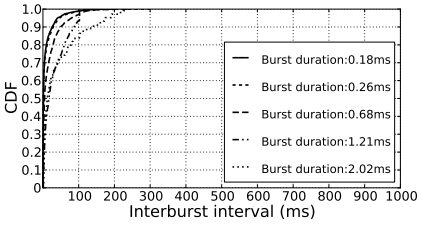
Note that emissions from different sources may overlap in time, for example, microwave emissions overlapping with Bluetooth bursts. In such cases, the clustering algorithm detects only the *dominant* interferer (i. e., the microwave). SpeckSense addresses this concern by observing RSSI values over a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

4 Interference Classification

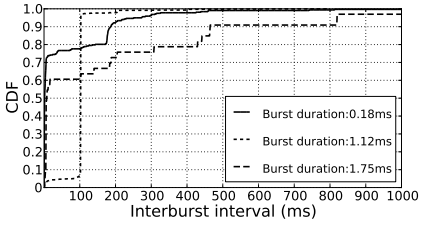
SpeckSense classifies interference by inspecting each detected cluster for temporal patterns in RSSI bursts. In doing so, SpeckSense informs link-layer protocols whether the observed channel activity is periodic, bursty or a combination of both. This facilitates a meaningful assessment of channel quality 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.



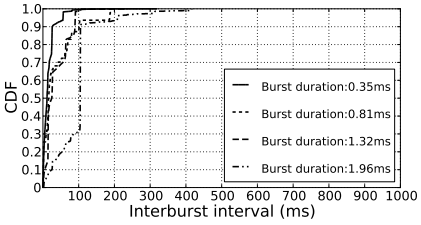
(a) Bluetooth file transfer, Avg. Interburst interval = 253 ms



(b) WiFi-enabled file download, Avg. Interburst interval = 23 ms



(c) Web browsing over WiFi, Avg. Interburst interval = 146 ms



(d) WiFi repeater traffic, Avg. Interburst interval = 50 ms

Fig. 2. 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.

4.1 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 from channel interference on a wireless network application is determined by several factors such as device usage patterns, application data requests as well as underlying communication protocols in use. 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 2 shows the cumulative distribution of the inter-burst separation for different clusters for some of the aforementioned cases (for additional details, refer to [12]). We observe that for cases where bursty traffic is involved, such as in Figures 2(b), and 2(d), 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.

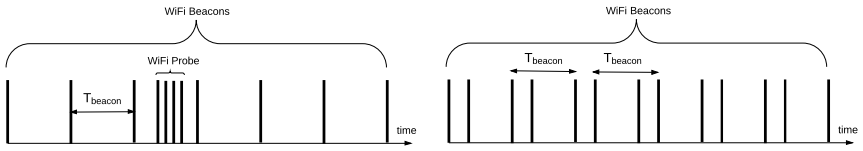
Based on these observations, SpeckSense uses an average inter-burst separation threshold of 100 ms, which has shown good results in distinguishing conditions of light channel traffic (cf. Figures 2(a), and 2(c)) from severe interference (cf. Figures 2(b) and 2(d)).

4.2 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 management frames 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 management frames 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 3(a)). Moreover, beacons from multiple WiFi access points within interference range may have similar on-air transmission times and RSSI values (see Figure 3(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 3.2. In every run, the input to the algorithm is a temporal sequence of RSSI bursts from a cluster. Let t_i denote the time at which the ith burst in the cluster was recorded by the node, where $1 \leq i \leq n$. 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



(a) Periodic WiFi beacons interspersed by probe messages.

(b) Periodic WiFi beacons from two access points

Fig. 3. 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

1: **Inputs**
2: ▷ n is the number of RSSI bursts over time T
3: ▷ $d_T = (d_t^1, d_t^2 \dots d_t^{n-1})$ is the sequence of inter-burst separations
4: **Outputs**
5: ▷ $P(d_\tau)$ is the confidence value for every $d_\tau \in L$
6: ▷ t_p is the detected periodicity of the sequence
7:
8: $L \leftarrow \emptyset$
9: **for** $d_t^i \in d_T$ AddToSet(L, d_t^i) **end for**
10: **for** $d_t^i \in (d_t^1, d_t^2 \dots d_t^{n-1})$ **do**
11: $s \leftarrow d_t^i$
12: **for** $d_t^j \in (d_t^{i+1}, d_t^{i+2} \dots d_t^{n-1})$ **do**
13: $s \leftarrow s + d_t^j$
14: UpdateSet(L, s)
15: **end for**
16: **end for**
17: **for each** $d_\tau \in L$ **do**
18: $n_\tau \leftarrow \lfloor \frac{T}{d_\tau} \rfloor$
19: $P(d_\tau) = 2C(d_\tau)/(n_\tau(n_\tau + 1))$
20: **end for**
21: $t_p = \text{argmax}_{d_\tau} P(d_\tau)$

included in the set L (Procedures 1, 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 $2\epsilon_\delta \approx 0.65$ 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 $d_\tau, 2d_\tau, 3d_\tau \dots kd_\tau$, 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 1, line 2 in *UpdateSet*). If not, the sum is added to the list, and its count is set to 1 (Procedures 1, lines 5–6 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

Procedures 1 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
```

```
1: procedure UpdateSet(L, d_t)
2:     if ∃d_τ ∈ L|d_t (mod d_τ) ∉ (ε_Δ, d_τ − ε_Δ) then
3:         C(d_τ) ← C(d_τ) + 1
4:     else
5:         L ← L ∪ d_t
6:         C(d_t) ← 1
7:     end if
8: end procedure
```

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 3(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 1.4 ms, gives good results. SpeckSense uses round$(P(d_\tau))$ as a measure for the number of distinct 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, 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 [9, 13] 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 [14] as well as microwave bursts [12].

5 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.

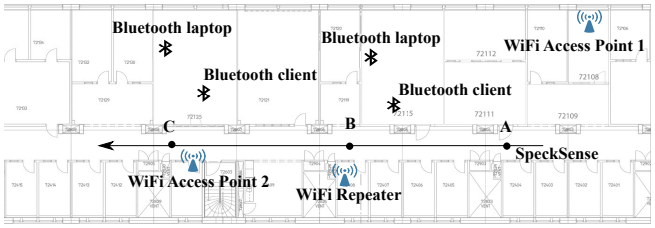


Fig. 4. Experimental setup in the office corridor. We evaluate SpeckSense at locations A, B and C in the presence of WiFi and Bluetooth interference.

5.1 Detecting Concurrent Interferers

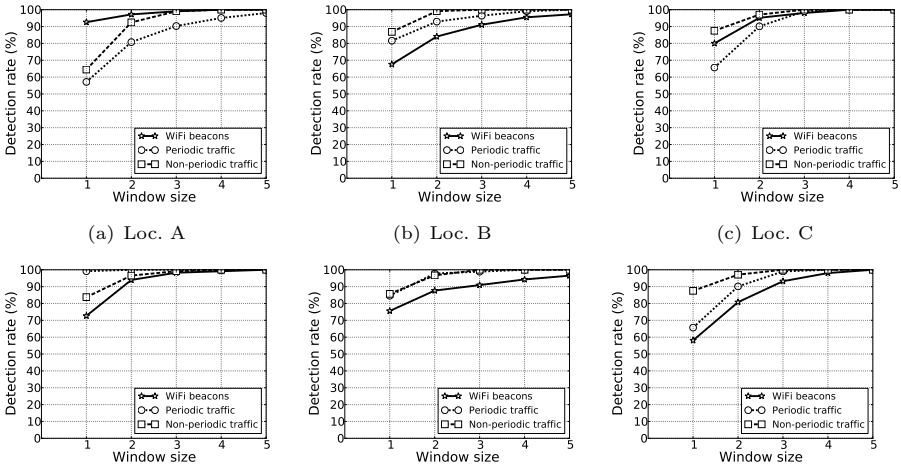
Indoor environments represent challenging conditions for SpeckSense due to non-line of sight between nodes that causes multipath fading effects. The extent of these effects may also vary over time, e.g., due to people moving, thereby increasing the variance in received signal strength on a sensor node. 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. 4. 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. 3.1. 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 5 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



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

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 5(a), 5(b), and 5(c) represents periodic TCP bursts in WiFi data transfers. In contrast, *periodic traffic* in Figures 5(d), 5(e), and 5(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 1 shows the 50th and 90th percentile 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.

5.2 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 exter-

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

Table 1. SpeckSense can detect multiple WiFi access points deployed over different locations on the office corridor. The values (50^{th} and 90^{th} percentile) indicate that SpeckSense can detect WiFi activity even in the presence of ambient 802.15.4 traffic.

nal 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 [15, 16], 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 [16], and 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 quality. The channel quality is 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. [17]. In our implementation, Chrysso *threshold* uses the best four channels in decreasing order of channel quality.

We experimentally evaluate the aforestated strategies on the Indriya WSN testbed [18], 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 [19]. We perform six experimental runs for each variant of Chrysso described above.

Table 2 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 approximately

	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 2. Data collection performance (averaged over six runs) on a 85-node testbed, highlighting the advantages derived from interference avoidance. SpeckSense with Chrysso performs best compared to other alternatives on avoiding interfered channels.

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 [20]. 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 an existing multichannel protocol such as Chrysso benefits from the interference classification output provided by SpeckSense.

6 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. [8] 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. [21] sample the spectrum for interferer classification but they only target interference occurring at regular intervals. Likewise, Zhou et al. [9, 13] propose an algorithm that is restricted to detecting WiFi beacons from RSSI traces. Another approach based on spectrum sampling is by Bloessl et al. [22]. In contrast to SpeckSense, their approach is limited to the detection of single interference sources. Ansari et al. [23] 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. [24] 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 [17, 25, 26], or packet reception counts [27] as selection criteria. In contrast to these approaches, we take the source of interference into account.

7 Conclusion

In this paper we have presented SpeckSense, a detection and classification scheme for concurrent multi-source interference affecting wireless sensor networks. Experiments in a real setting have shown that SpeckSense detects multiple interferers in over 90% of the cases. We have also evaluated SpeckSense as a low-layer service to recommend interference-free channels for WSN data collection. Experiments combining the results of SpeckSense with a multichannel protocol have shown a significant improvement in data yield at lower duty cycle.

Acknowledgement This work has been partially supported by the European Commission with contract INFSO-ICT-317826 (RELYonIT) and SSF.

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. Zuniga, "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, "A lightweight classification algorithm for external sources of interference in ieee 802.15.4-based wireless sensor networks operating at the 2.4 GHz," *IJDSN*, 2014.
8. 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.
9. 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.
10. C. A. Boano, T. Voigt, C. Noda, K. Romer, and M. Zuniga, "JamLab: Augmenting sensornet testbeds with realistic and controlled interference generation," in *ACM IPSN*, 2011.

11. 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.
12. V. G. Iyer, F. Hermans, and T. Voigt, "Detecting and avoiding multiple sources of interference in the 2.4 ghz spectrum," Department of Information Technology, Uppsala University, Tech. Rep. 2014-023, Dec. 2014.
13. 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.
14. F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with Glossy," in *IPSN*, 2011, pp. 73–84.
15. 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.
16. B. Al. Nahas, S. Duquennoy, V. Iyer, and T. Voigt, "Low-Power Listening Goes Multi-Channel," in *IEEE DCOSS*, Marina Del Rey, CA, USA, 2014.
17. 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.
18. 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.
19. 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.
20. R. Jain, *The art of computer systems performance analysis*. John Wiley & Sons, 2008.
21. 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.
22. 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.
23. 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.
24. 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.
25. J. Ansari and P. Mähönen, "Channel selection in spectrum agile and cognitive MAC protocols for wireless sensor networks," in *ACM MobiWac*, 2010.
26. 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.
27. 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.