Passive, Privacy-preserving Real-time Counting of Unmodified Smartphones via ZigBee Interference

Roman Lim, Marco Zimmerling, Lothar Thiele
Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland
{lim, zimmerling, thiele}@tik.ee.ethz.ch

***Abstract*—The continuing proliferation of smartphones makes them an effective means to monitor the number of people within an area, for example, to gain insights into customer engagement in retail and to enable an intelligent traffic system in a city. However, current approaches to obtain this information are either invasive as they require to continuously run a dedicated smartphone app, or they compromise users' privacy by sniffing the MAC addresses of their smartphones. As a consequence, lawyers, authorities, and the population are very skeptical toward adopting such innovative systems. We present DEVCNT, the first system that counts in real-time the number of Wi-Fi enabled smartphones in a non-invasive manner while preserving by design the privacy of the smartphone users. This paper details how DEVCNT detects active Wi-Fi scans performed by smartphones on a ZigBee device, and how DEVCNT uses the number of detected scans to estimate the number of Wi-Fi enabled smartphones. Results from controlled and real-world experiments show that DEVCNT: (*i*) detects more than 99 % of active Wi-Fi scans even under interference from multiple wireless technologies, (*ii*) achieves up to 91 % accuracy in the estimated smartphone counts, and (*iii*) provides meaningful estimates in a real test run involving hundreds of Wi-Fi transmitters.**

I. INTRODUCTION

Smartphones are an integral part of our daily lives, and will be even more so in the future. Ericsson forecasts 5.6 billion smartphones around the world in 2019, accounting for 60 % of all mobile subscriptions [8]. This proliferation is intriguing as it opens up the possibility to exploit smartphones for collecting statistically significant amounts of data about the way people behave and interact [3]. In addition, nowadays almost every smartphone has Wi-Fi built in. Due to ever-increasing mobile data traffic [8], users activate Wi-Fi on their smartphones to get fast and cheap connectivity to a Wi-Fi network whenever possible. To discover these offloading opportunities, smartphones actively scan for Wi-Fi access points (APs) by periodically sending out probe request frames, or *probes* for short.

Motivation. We seek to leverage the high proliferation of Wi-Fi enabled smartphones to determine their numbers in real-time based on the probes they emit. Given the high penetration of smartphones across the general population, such counts are of great value in numerous applications. For example, they can be used to estimate the density of a crowd, which is an important feature in crowd management [11] to prevent disasters like the 2010 Love Parade stampede that killed 18 people [25]; in retail, they provide insights into customer engagement and help improve in-store sales [5]; and in a city, information about the number of pedestrians, cyclists, and motorists using particular road segments at any given time enables an intelligent transport system where traffic lights adapt to reduce travel times [24].

Prior to the advent of the smartphone, video surveillance and image processing have been used to estimate the number of people in an area [13]. These solutions, however, need special-purpose powered equipment and impact privacy. Today, an alternative solution is to encourage people to run an application on their smartphones that periodically sends the GPS location to a server [27]. While here users can opt out at any time, this approach is invasive in that it alters the smartphone software, and works only outdoors where GPS reception is possible.

A non-invasive approach is to use existing Wi-Fi APs or to deploy dedicated Wi-Fi monitors to estimate people count by sniffing the unique MAC addresses of their smartphones, which are contained in clear text in every probe [5], [20]. From a privacy perspective, this is even worse than video surveillance because the owner, which can be unequivocally identified from the MAC address, has no means to notice that she is being tracked. Thus, lawyers, authorities, and the population take a skeptical position *despite the use of anonymization techniques such as MAC address hashing*. This could be witnessed, for example, by a public outcry and eventual ban of such a system in the City of London [23], and unclear current law preventing the deployment of a smart traffic system in Copenhagen [24].

Contribution. To address these issues, we introduce DEVCNT, a system providing real-time estimates on the number of Wi-Fi enabled smartphones within an area in a non-invasive manner. DEVCNT preserves *by design* the privacy of smartphone users, thus overcoming the concerns associated with prior approaches and fostering the rapid deployment of innovative applications.

To this end, DEVCNT takes advantage of cross-technology interference in the 2.4 GHz band. As described in Sec. II, the IEEE 802.11 standard prescribes that an *active scan* should involve sending a probe on each Wi-Fi channel. Since each Wi-Fi channel overlaps with at least one ZigBee channel, a ZigBee device can perceive a probe transmission as a short increase in the received signal energy. DEVCNT uses one or multiple battery-powered ZigBee devices to flexibly cover large areas. Every device periodically reports the number of active Wi-Fi scans it has seen to a sink. Based on the received scan counts, DEVCNT estimates at the sink the number of Wi-Fi enabled smartphones within the range of each ZigBee device.[1]

DEVCNT works both indoors and outdoors, and provides a new estimate every few seconds. DEVCNT is non-invasive and fully passive in that it neither modifies the software running on the smartphones, nor does it externally affect the smartphones' operation (as done in [20]). Because it is technically impossible for a ZigBee receiver to demodulate Wi-Fi frames, DEVCNT cannot track individual smartphones nor identify their owners.

[1] Since this paper deals primarily with the physical and media access layers of IEEE 802.15.4 and IEEE 802.11, we do not discern between these standards and the respective industrial alliances ZigBee and Wi-Fi.

Finally, by using low-power wireless battery-powered devices, DEVCNT reduces costs and increases flexibility during system installation, maintenance, and removal compared with previous solutions that use video surveillance or Wi-Fi monitors.

Achieving these favorable properties while providing accurate smartphone counts is challenging for at least three reasons:

- Amplitude and time resolution of received signal strength (RSSI) information on a ZigBee receiver is coarse-grained. DEVCNT must therefore cope with inaccuracies when characterizing Wi-Fi transmissions by their length and signal strength, which are the only features DEVCNT is left with to detect scans on a device that cannot demodulate Wi-Fi.
- DEVCNT cannot differentiate between individual smartphones because it cannot not see their (unique) MAC addresses, probes from different smartphones may have the same length, and RSSI is a poor classifier due to mobility and multipath fading. While this preserves privacy, it also makes counting Wi-Fi enabled smartphones a difficult task.
- Sending all RSSI samples to a central sink for processing is prohibitive due to the limited bandwidth. Thus, as detailed in Sec. III, DEVCNT performs most of the required processing on the ZigBee devices. This, in turn, implies that each ZigBee device needs to multiplex its single microcontroller unit (MCU) between three tasks: (*i*) reading RSSI samples, (*ii*) processing samples to detect and count scans, and (*iii*) sending scan counts to the sink. DEVCNT must temporally decouple (*i*) and (*ii*) from (*iii*) synchronously on all devices to avoid distorting the smartphone count estimations due to self-interference, while simultaneously reducing the time needed for (*iii*) to have more time available for (*i*).

As discussed in Secs. IV and V, we tackle these challenges by designing novel signal processing, feature extraction, and classification algorithms. Given that these algorithms need to execute in real-time on devices with severely limited memory and compute power, our algorithms strike a balance between feasibility and optimality. The key insight we use to count smartphones without being able to distinguish them is that the active Wi-Fi scanning rate of a sizable population of smartphones follows a specific and typically narrow distribution. To enable temporal decoupling and fast all-to-one data collection, we use Glossy [9] to accurately time-synchronize the ZigBee devices and Chaos [16] as efficient communication support.

To demonstrate the feasibility of our design, we implement a DEVCNT prototype on the TelosB [21] platform. In Sec. VII, we evaluate DEVCNT in controlled experiments with up to 31 smartphones from 4 different vendors running iOS or Android, and during a real-world test run in a large lecture hall with more than 100 students. Our results show the following:

- DEVCNT accurately detects more than 99 % of Wi-Fi scans despite realistic interference from Bluetooth, ZigBee, and Wi-Fi traffic, the latter including TCP and UDP streams.
- DEVCNT detects scans with an accuracy above 90 % even on ZigBee devices that are 50 m away from the phone.
- In a controlled experiment where the precise ground truth is available, DEVCNT estimates the number of Wi-Fi enabled smartphones with accuracies of up to 91 %.
- In a real-world test run where the ground truth is extremely difficult to obtain, DEVCNT's processing pipeline sustains signals from hundreds of Wi-Fi transmitters, thus providing meaningful smartphone counts that match the expectations.

Sec. VIII discusses trade-offs and limitations of DEVCNT, Sec. IX reviews related work, and Sec. X concludes the paper.

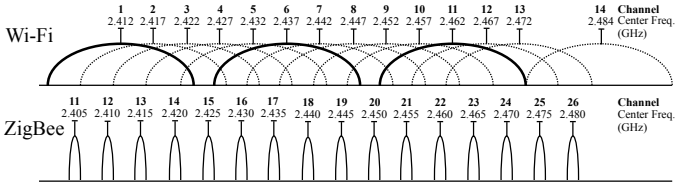


Figure 1. IEEE 802.11b/g (Wi-Fi) and IEEE 802.15.4 (ZigBee) channels in the 2.4 GHz industrial, scientific and medical (ISM) band.

II. BACKGROUND AND TERMINOLOGY

DEVCNT takes advantage of cross-technology interference between Wi-Fi and ZigBee in the 2.4 GHz band. IEEE 802.11 defines in total 14 channels, as shown in Fig. 1. Each channel is 22 MHz wide and the center frequencies range from 2.412 to 2.484 GHz. Channel 14 is forbidden in most parts of the world, and channels 12 and 13 are typically not used in North America due to FCC regulations. IEEE 802.15.4 defines in total 16 channels with a bandwidth of 2 MHz each and center frequencies between 2.405 and 2.480 GHz. Thus, ZigBee and Wi-Fi can interfere when operating in overlapping channels.

While often regarded as a major impediment to the performance of ZigBee networks [17], DEVCNT takes advantage of this kind of interference. It uses *received signal strength (RSSI)* information available on commodity ZigBee devices to detect an interfering Wi-Fi transmission from a short-lived increase in the RSSI. According to the IEEE 802.15.4 standard, the RSSI value is an average over the last 8 *symbol periods* (128 μs), and is typically updated on a per symbol basis, that is, every 16 μs. Furthermore, many ZigBee radios allow to adjust their operating frequency with a certain granularity (*e.g.*, 1 MHz on the CC2420 radio [22]). As explained in Sec. VI, we use this feature in DEVCNT to best align the operating frequency of the ZigBee devices with the center frequency of a Wi-Fi channel.

The majority of Wi-Fi networks operates in infrastructure mode, where *access points (APs)* manage all communications. A client, such as a laptop or a smartphone, needs to associate with an AP before it can use any network services. The IEEE 802.11 standard defines, among other things, a process called *active scanning*, whereby a client sends probe request frames (*probes*) to discover an AP. The standard prescribes that an active *scan* should involve broadcasting a probe on each channel, awaiting and processing possible responses from APs in between. Nevertheless, no further detailed specification of active scanning is provided in the IEEE 802.11 standard. As a result, different Wi-Fi drivers implement active scanning slightly differently. For example, we noticed that many drivers send multiple probes (mostly two) on each channel, as opposed to just one probe. Smartphones perform active scanning periodically every few tens of seconds [10], depending on the operating system and the current operating mode (*e.g.*, whether the smartphone is actively used or in standby mode).

Although there is also a *passive scanning* process foreseen in the IEEE 802.11 standard, where clients passively listen for beacon frames from APs, mobile devices mostly rely on active scanning because it is faster. Typically, APs announce their presence by sending a beacon frame every 102.4 ms. A client must listen for at least that period on every channel to finish a passive scan, whereas probe requests are usually handled within a much shorter time. In the remainder of this paper, we use the term *scan* to refer to active scans.

An IEEE 802.11b/g frame is preceded by a preamble that is at least 96 μs long. This is above the 16 μs symbol period of

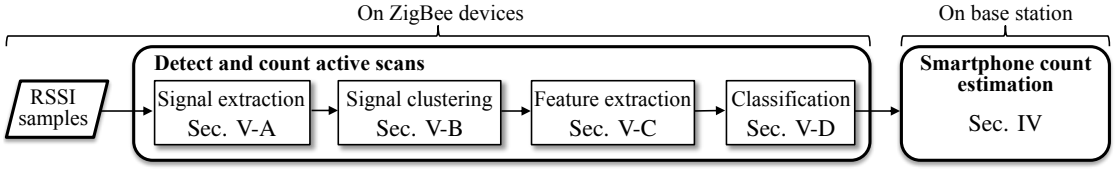


Figure 2. High-level view of DEVCNT. *A multi-hop low-power wireless network of ZigBee devices continuously samples the received signal strength (RSSI), and uses a novel signal processing pipeline to detect and count active Wi-Fi scans performed by smartphones. Based on the scan counts periodically reported by the ZigBee devices,* DEVCNT *estimates on a base station the number of Wi-Fi enabled smartphones within the reception range of each ZigBee device.*

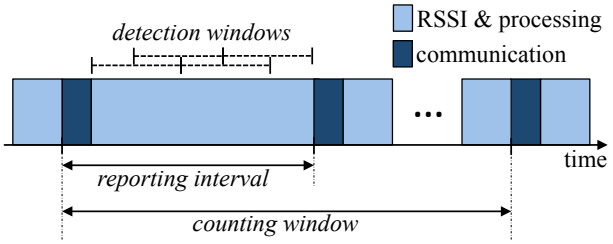


Figure 3. Illustration of how the activities of RSSI sampling, processing, and communication evolve over time in DEVCNT. *To avoid negative self-interference effects, RSSI sampling and processing are temporally decoupled from communication. The reporting interval determines the timeliness of smartphone counting in* DEVCNT*, which is on the order of a few seconds.*

IEEE 802.15.4, so Wi-Fi frames are, in principle, detectable from the RSSI samples of a ZigBee radio. After the preamble, a Wi-Fi probe contains in unencrypted form the client's MAC address, the broadcast address or the SSID of a known Wi-Fi network of up to 32 bytes, the data rates supported by the client, and other (*e.g.*, vendor-specific) information.

III. DEVCNT OVERVIEW

We present DEVCNT, the first system that provides real-time estimates on the number of Wi-Fi enabled smartphones in an area in a fully passive and non-invasive manner without revealing the identity or movement profile of smartphone users.

DEVCNT counts the number of Wi-Fi enabled smartphones based on the probes they send while actively scanning for nearby Wi-Fi APs. Instead of directly eavesdropping on probes using a Wi-Fi capable receiver, DEVCNT detects probes *indirectly* via their interference patterns at a ZigBee receiver. Since ZigBee radios cannot demodulate Wi-Fi probes, DEVCNT does not see the unique MAC addresses contained therein and hence is unable to identify individual phones. As a result, DEVCNT *preserves by design the privacy* of the smartphone users, which sharply differentiates DEVCNT from prior art [5], [13], [20]. Moreover, DEVCNT is *fully passive* and *non-invasive*: It does not influence a smartphone's normal operation, for example, by soliciting more probe transmissions [20], nor does it require modifications to the smartphones themselves, such as installing and running a dedicated application [27].

Fig. 2 provides a high-level view of DEVCNT, while Fig. 3 illustrates how the different activities in DEVCNT evolve over time. DEVCNT uses a multi-hop low-power wireless network of battery-powered ZigBee devices (*nodes*) deployed across the area of interest; a sink node is connected to a base station. Each node continuously samples the received signal energy by retrieving RSSI from the radio. Based on these RSSI samples, a node detects and counts the number of scans over a sequence of overlapping *detection windows*, as illustrated in Fig. 3, using the four-stage signal processing pipeline shown in Fig. 2. All nodes transmit the number of scans they detect within a regular periodic *reporting interval* to the sink. On the base station, DEVCNT uses all scan counts received over a certain *counting window* (see Fig. 3) to estimate the number of Wi-Fi enabled smartphones within the reception range of each node.

To avoid interference between ZigBee devices, which could adversely affect the smartphone count estimations, DEVCNT decouples RSSI sampling and processing from communication over time. Temporal decoupling requires the devices be time-synchronized, which we achieve by letting the sink perform a Glossy network flood [9] at the beginning of every communication phase illustrated in Fig. 3. In fact, the communication phases should be as short as possible to maximize the time the radio is available for RSSI sampling. We thus leverage Chaos as efficient communication support [16]. Chaos enables DEVCNT to collect small amounts of data, such as a 1-byte scan count, from 100 nodes within less than 100 milliseconds [16], thus increasing the time available for RSSI sampling. Sec. VII shows that DEVCNT computes new estimates every few seconds based on up-to-date scan counts collected from ZigBee devices, enabling real-time crowd monitoring and analysis.

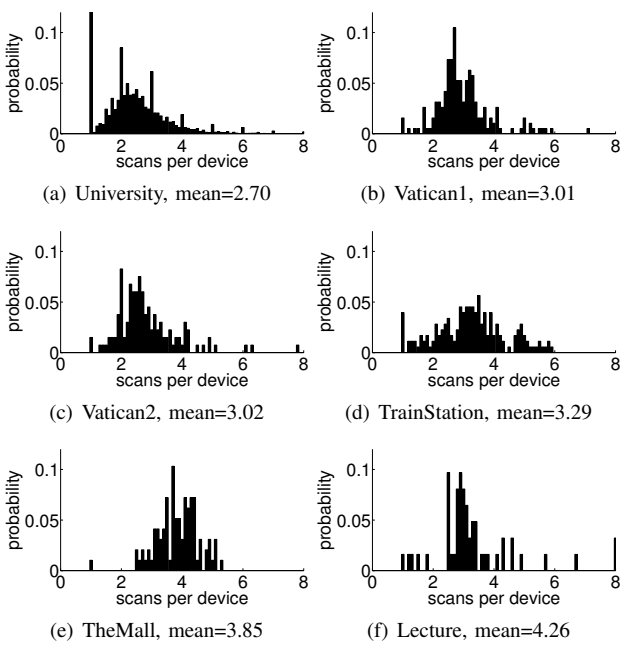
By designing and implementing a DEVCNT prototype, we demonstrate that it is indeed possible to perform almost the entire processing *online* on resource-constrained devices. This includes in particular non-trivial signal processing, clustering, feature extraction, classification, and filtering algorithms to detect and count scans (see Fig. 2), which has been considered too computationally demanding and hence impossible [28].

Next, Sec. IV details our approach to estimating the number of Wi-Fi enabled smartphones, and Sec. V describes how we detect and count active Wi-Fi scans on ZigBee devices.

IV. ESTIMATING SMARTPHONE COUNTS

Counting smartphones without being able to identify them is difficult. Indeed, DEVCNT cannot identify smartphones by their MAC addresses, since a ZigBee device cannot demodulate Wi-Fi frames. Identification through RSSI is extremely noisy due to mobility and environment dynamics [20]. Another option could be to use some form of fingerprinting to passively identify a smartphone based on device- and/or driver-specific variations in active scanning [10], [19]. Unfortunately, such variations (*e.g.*, in the scanning rate) can be extremely small, especially across smartphones from the same vendor running identical Wi-Fi drivers and operating systems, requiring observation periods of an hour or more [19]. Moreover, these techniques assume that consecutive scans of the same device can be grouped together by matching packet contents— however, DEVCNT cannot access the contents of Wi-Fi packets.

From the discussion above, it is clear that the only viable option we are left with is to estimate the number of Wi-Fi enabled smartphones based on statistical information about their active scanning behavior. As discussed in Sec. II, smartphones perform active Wi-Fi scans with a certain periodicity to quickly discover a nearby AP. Nevertheless, the interval between scans is not fixed and depends on several factors, including the smartphone vendor and model, the Wi-Fi driver, the operating



(a) University, mean=2.70 (b) Vatican1, mean=3.01

(c) Vatican2, mean=3.02 (d) TrainStation, mean=3.29

(e) TheMall, mean=3.85 (f) Lecture, mean=4.26

Figure 4. Empirical probability distribution of scan counts within a 3-minute time window for different real-world datasets containing massive Wi-Fi traces.

system, the applications that are currently running, and whether the smartphone is in active or in standby mode. So a natural questions that arises is whether statistical information about the frequency of active scans is indeed useful to accurately estimate the number of Wi-Fi enabled smartphones.

To answer this question, we analyze five different publicly available datasets collected by researchers from the Sapienza University of Rome, Italy [2]. The datasets contain significant traces of Wi-Fi probes recorded with a commodity Wi-Fi card in monitor mode in diverse scenarios: at international events (Vatican1 and Vatican2), in a big mall (TheMall), at the central train station (TrainStation), and at one of the main entrances of Sapienza (University) [3]. In addition, we analyze one dataset recorded by us in a (Lecture) hall. The traces range from a few hours to several weeks in length, containing between a few thousand and several million probes. They also differ as to whether the smartphones were likely mobile (University) or static (Lecture), whether the users were likely using their phone (TheMall, TrainStation) or just carrying it in standby mode in their pockets (University), and so on.

Fig. 4 plots for each dataset the empirical probability distribution of the number of active scans per smartphone over an interval of 3 minutes. We see that despite the diversity of scenarios, all distributions have a very similar shape and a mean of around 3 scans per device. Based on the scenario, we notice slight differences between the means. In the University trace, for example, smartphones are only for a short time in the vicinity of the Wi-Fi sniffer located at an entrance, and also likely in standby mode as they are carried in bags or pockets. As a result, the average number of scans seen from a device is a bit smaller (2.70). By contrast, in the TrainStation and TheMall traces, the smartphones are more static and actively used (*e.g.*, while waiting for a train, taking a break, etc.), so the average number of scans per device is a bit higher (3.29–3.85).

These observations are encouraging in that the average number of active scans per device over some *counting window* is fairly stable despite several influencing factors. This raises

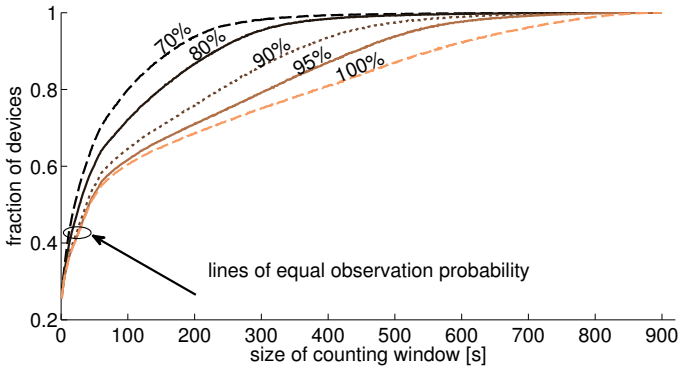


Figure 5. Fraction of devices that can be seen within a certain interval in the University Wi-Fi dataset. *The empirical cumulative distribution function is shown for five different observation probabilities.*

the hope that we can use this figure to accurately estimate the number of Wi-Fi enabled smartphones, especially if the expected mobility and usage of smartphones are known, which is a valid assumption in the applications we target. Experiments in Sec. VII-B show that DEVCNT can estimate smartphone counts with an accuracy of up to 91 %. Key to this performance is (*i*) the ability to accurately detect active scans on a ZigBee device, as described in the following section, and (*ii*) an appropriate size of the counting window, as discussed next.

Size of counting window. The counting window (see Fig. 3) should be rather large to likely contain one or more active scans from a significant fraction of smartphones to provide *accurate* estimates, whereas it should be rather short to provide *timely* estimates. To study this trade-off, we use the University trace and plot in Fig. 5 the fraction of devices seen with a certain probability depending on the size of the counting window. As expected, if we want to detect a given fraction of smartphones with a higher probability, we have to choose a larger counting window. For example, using a counting window of 3 minutes, we see with 80 % probability roughly 85 % of smartphones. The size of the counting window can be adjusted to match the accuracy and real-time requirements of the application.

V. DETECTING AND COUNTING ACTIVE WI-FI SCANS

As mentioned in the previous section, DEVCNT relies on accurate scan counts to facilitate meaningful estimates on the number of Wi-Fi enabled smartphones in a given area. To this end, DEVCNT processes RSSI samples in *real-time* on the nodes using a novel four-stage pipeline, as shown in Fig. 2.

In a first step, DEVCNT extracts interesting portions from a trace of RSSI samples, that is, short-lived periods of elevated signal strength, further referred to as *signals*.

Next, DEVCNT takes advantage of pertinent features of the active scanning function, including the periodicity with which probes are sent during an active scan. Specifically, DEVCNT (*i*) groups all signals observed over a fixed-length detection window together, and (*ii*) clusters the signals inside each group based on their length. The reasoning behind (*ii*) is that probes sent by a smartphone during an active scan have the same length; clustering ensures that signals that likely originate from the same smartphone are also considered together.

Afterward, DEVCNT checks whether a detection window contains signals from an active scan or not. To do so, it first computes a set of features for each cluster in that window. Then, it uses these features to classify each cluster as either ”contains a scan” (Y) or ”contains no scan” (N). If a detection

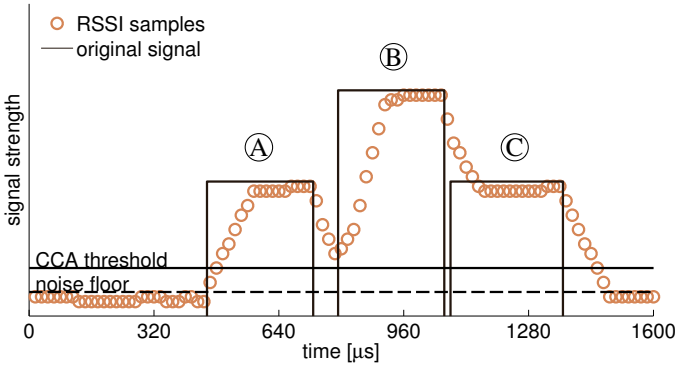


Figure 6. Example of three original signals and the corresponding trace of RSSI samples. *The RSSI values are an average over the last 8 symbol periods. Thus, a naïve threshold-based approach cannot reliably discern close signals.*

window contains at least one (Y) cluster, DEVCNT considers that detection window as containing an active scan.

At the end of every reporting interval (see Fig. 3), a node sums up the number of detection windows with an active scan seen in the current interval, and transmits this scan count (via Chaos) to the sink, which forwards it to the base station.

In the remainder of this section, we take a closer look at each of the four processing steps above. Experimental results in Sec. VII show that due to our techniques DEVCNT detects and counts active scans with an accuracy above 99 % despite realistic interference from other wireless technologies.

A. Signal Extraction

A trace of RSSI samples is not very useful by itself. Rather, DEVCNT must first identify and extract parts of an RSSI trace that may (or may not) belong to a probe transmission. For this reason, DEVCNT needs to find on the fly the beginning and end of periods with elevated RSSI readings (signals), which is however a non-trivial task.

Fig. 6 shows an example RSSI trace with three signals. As mentioned before, RSSI is an average over the last 8 symbol periods (128 μs), which leads to an inherent smoothing effect. Therefore, as visible for signal Ⓐ, it takes 8 samples until the RSSI readings match the amplitude of the original signal. This complicates determining the beginning of a signal. Moreover, if the gap between two signals is less than 128 μs, such as between signals Ⓑ and Ⓒ, the RSSI values remain above the noise floor because they account for parts of either signal. Thus, a naïve threshold-based approach, which could be realized using the clear channel assessment (CCA) capability of a ZigBee radio, cannot reliably discern close signals.

Our solution to these problems is based on the observation that once a signal is present, the RSSI will plateau after some time and eventually start to fall again. Specifically, we abstract this trend as a sequence of states: *rising* $\rightarrow$ *steady* $\rightarrow$ *falling*. A node dynamically determines the current state by looking at the differences between RSSI samples. Based on this idea, we devise the state machine shown in Fig. 7 to accurately identify the beginning and the end of signals.

The state machine operates on a sequence $s_1, s_2, \ldots, s_i$ of RSSI samples. The processing starts when the CCA pin set by the radio indicates an RSSI level above some threshold. Upon taking a transition, we read the next RSSI sample s_i. We remain in state *rising* as long as s_i is greater than the average of the two previous samples. Otherwise, we advance to an auxiliary state *first*, which serves to initialize two variables

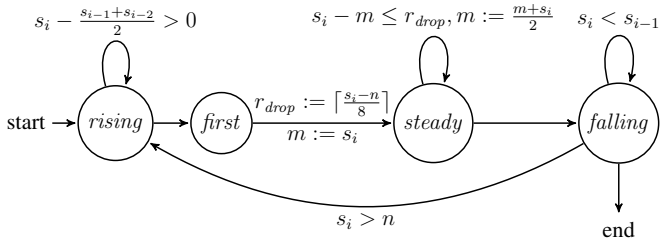


Figure 7. State machine implemented by a DEVCNT device to determine the beginning and the end of a signal based on a trace of RSSI samples.

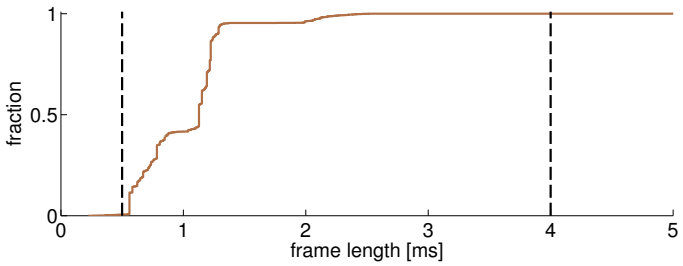


Figure 8. CDF of probe lengths of over 7 million probes in the University trace. *More than 99 % of probes have a length between 0.5 ms and 4 ms.*

needed to determine the end of the signal: m, a moving average of the RSSI samples and r_{drop}, the *drop rate*. We set the drop rate dynamically, because strong signals cause a larger drop in the (averaged) RSSI values than weak signals. For example, in Fig. 6 the drop rate of Ⓑ is higher than the drop rates of Ⓐ and Ⓒ. We set the drop rate r_{drop} to one eighth of the difference between the RSSI at the beginning of a plateau and the noise floor n. Once the original signal disappears, the RSSI will linearly drop by r_{drop} every symbol period due to the averaging. Based on both m and r_{drop}, we decide whether to stay in state *steady* or to move to state *falling*. The processing stops when the RSSI values fall below the noise floor.

In this way, we precisely identify on the fly the beginning and end of signals. To save memory and computational resources, we use this information to already filter out signals too short or too long to be a probe. Based on the University data set, which contains more than 7 million probes recorded over a period of 10 weeks, we plot in Fig. 8 the cumulative distribution function (CDF) of probe air times. We find that 99.5 % of probes are between 0.5 ms and 4 ms long. Thus, a DEVCNT node only stores the average RSSI as well as the start and end times of signals that fall into this range.

B. Signal Clustering

An individual signal alone does not provide enough information to decide whether it is from a probe or not. Instead, we should look for relations among multiple signals in order to make this decision. To see why this is a sensible approach, we chart in Fig. 9 a RSSI trace recorded with a TelosB during an active scan of a smartphone. Each signal corresponds to a probe, and signals with a similar amplitude correspond to probes that are sent on the same channel. We clearly notice, for example, a periodic pattern, which is however only apparent when looking at the entire group of signals.

Ideally, such group formation distinguishes between signals from an active scan and other signals. As mentioned in Sec. II and visible from Fig. 9, probes that belong to the same active scan have the same length and quickly follow each other. We exploit these properties by (*i*) grouping signals observed within

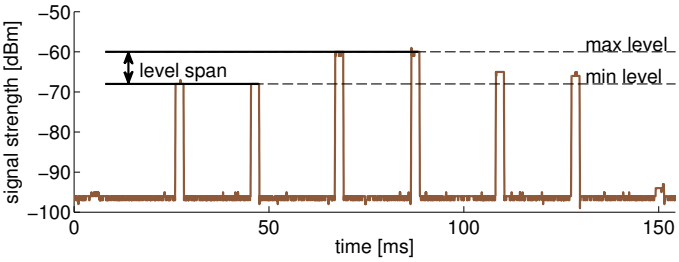


Figure 9. RSSI trace recorded with a TelosB during an active Wi-Fi scan of a nearby smartphone. *Probes sent on adjacent channels exhibit a characteristic periodicity and difference in signal strength (level span) across channels.*

a short *detection window*, and (*ii*) clustering the signals inside each group based on their length.

Size of detection window. The detection window should be large enough to contain sufficient signals from the *same* scan, but short enough so that it likely contains no signals from *different* scans. We find a good detection window size based on a 2-hour trace captured with a Wi-Fi receiver in a student lab. Due to the overlap of adjacent Wi-Fi channels, a ZigBee device often sees probes from 3 adjacent channels, as in Fig. 9. In our trace, we find that in more than 99 % of the cases a scan across 3 adjacent channels takes less than 290 ms. We found very similar numbers also in other traces. Thus, we use detection windows that are $2 \times 290 = 580$ ms wide, and let them overlap by half of this size, as shown in Fig. 3. This is because if we were to use contiguous detection windows, we would miss scans that cross detection window boundaries.

Clustering signals by length. To cluster the signals in a group, we first sort them by length. Then, we form clusters of signals so that signals in different clusters differ by more than a certain threshold. According to Nyquist's Theorem a sampling rate of $1/16\,\mu$s (*i.e.*, the inverse of the IEEE 802.15.4 symbol period) allows to sample changes in the received signal strength at half of this rate. Thus, signal lengths that differ by less than 2 symbol periods ($32\,\mu$s) are indistinguishable. To compensate for possible errors due to the averaging performed by a ZigBee radio, we add a slack of 2 symbol periods and use an inter-cluster separation of $64\,\mu$s in signal length.

C. Feature Extraction

Next, DEVCNT must decide whether a given cluster contains signals from a scan (Y) or not (N). To enable such classification, we require a set of features that (*i*) are expressive to reliably distinguish between (Y) and (N) clusters, and (*ii*) can be quickly computed on resource-constrained devices.

We explored various features and combinations thereof, but eventually settled on two features that best satisfy our needs. The first feature is based on the *autocorrelation*, and allows us to check for the presence of repeating patterns across the signals in a cluster. For example, when applied to the signals in Fig. 9, this feature, f_p, can clearly identify a periodicity. The periodic pattern results because scans are a repeated sequence of probes sent in every channel. To distinguish between beacon frames sent by APs and active scans, we exclude the typical beacon frame period of 102.4 ms from our feature by only considering lags in the range [15, 95] ms.

The second feature, called *level span*, exploits that probes sent on adjacent Wi-Fi channels during a scan result in different signal levels at a ZigBee radio. As shown in Fig. 9, we define the level span, f_l, as the difference between the highest and the lowest amplitude across all signals in a cluster

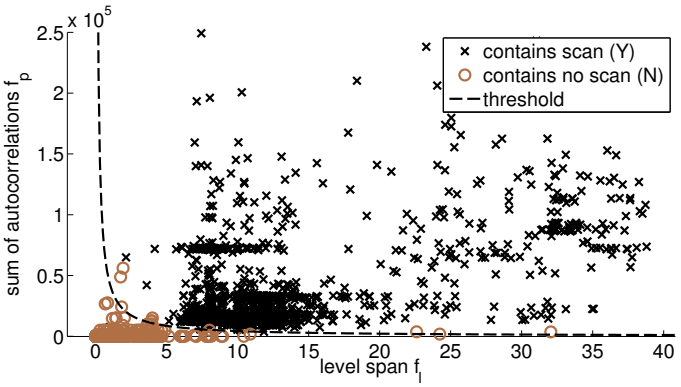


Figure 10. Illustration of classifying signal clusters into those containing an active scan (Y) and those containing no active scan (N), based on data from a real-world experiment with multiple interference sources. *By applying a threshold on the product of the sum of autocorrelations feature f_p and the level span feature f_l,* DEVCNT *accurately classifies almost all clusters.*

to check for this kind of pattern. By applying a threshold on the product of both features, we can accurately distinguish between (Y) and (N) clusters, as shown in Fig. 10 for data from an experiment with several interferers (see Sec. VII-A1).

While the level span feature f_l can be quickly computed even on a resource-constrained platform, this does not hold for the periodicity feature f_p based on the autocorrelation, as also acknowledged by prior work [28]. We explain in the following how we tackle this challenging problem in DEVCNT.

Intuition. The periodicity feature f_p only cares about *when* signals in a cluster occur, and not about their amplitude. We thus represent a cluster as a discrete-time binary time series $\{x_i\}_1^w$, where x_i is 1 if and only if at time instant i a signal is present, and w is the size of a detection window. The autocorrelation ρ at lag τ is defined as

$$\rho(\tau) = \sum_{i=\tau+1}^{w} x_i x_{i-\tau}. \tag{1}$$

Checking whether a cluster exhibits a periodic pattern with a period in the interval $[a, b]$ entails computing the autocorrelation $\rho(\tau)$ using (1) for *each* lag $\tau \in \{a, a+1, \ldots, b\}$. This, however, leads to prohibitive processing times on a resource-constrained platform.

The key insight we use to overcome this problem is that there is instead a way to efficiently compute the *sum* f_p of autocorrelations $\rho(\tau)$ over all lags τ in the interval $[a, b]$

$$f_p = \sum_{\tau=a}^{b} \rho(\tau). \tag{2}$$

Intuitively, using the sum makes sense, because higher individual autocorrelations indicating periodicity result in a higher sum. While the inverse of this argument is not always true, empirical evidence from our real-world experiments shows that this approach is highly effective.

Algorithm. To efficiently compute the sum of autocorrelations feature f_p, we note that (2) can be transformed into

$$f_p = \sum_{j=1}^{w-a} \sum_{k=j-a}^{\min(w,j+b)} x_j x_k. \tag{3}$$

Crucially, (3) no longer iterates over individual lags τ: it essentially sums across the area that is bounded by a and b.

Nevertheless, rather than summing up numerous "useless" 0's across the entire area, it is sufficient to only consider subareas containing 1's. The beginning and end of these subareas are precisely the x_i that mark the beginning and end of a period in which signals are present in a cluster. These observations materialize in an efficient algorithm for computing the sum of autocorrelations feature f_p. We illustrate this algorithm through an illustrative example and its pseudocode in the Appendix.

D. Classification

At the end of a detection window, each DEVCNT node computes the two features above for each individual cluster and feeds them into a classification algorithm. If one or more clusters in a detection window are classified as containing signals from an active scan (Y), the node considers the whole detection window as containing a scan. As a result, it increments its local scan count, which it sends every reporting interval to the sink and then resets to zero.

Because fast processing is key in DEVCNT, we opt for a computationally cheap decision tree classifier [6]. For the same reason, instead of considering the two features separately, we use a threshold on their product for classification. We found this approach to be slightly more efficient in most of our tests without sacrificing classification accuracy. Therefore, the classification works on a decision tree with one branch (*i.e.*, one single *if*-statement) and incurs little runtime overhead. We use the `fitctree` function available in MATLAB to determine a threshold on the product of the two features, using a training set collected in a controlled experiment with several smartphones from different vendors and different interference sources, described in Sec. VII-A1.

VI. IMPLEMENTATION

With the help of the FlockLab testbed at ETH Zurich [18], we have implemented a DEVCNT prototype on top of the Contiki operating system [7]. Our prototype targets the TelosB platform, which features an 8 MHz MSP430 MCU, an IEEE 802.15.4-compliant 250 kbit/s low-power CC2420 radio, 10 kB of RAM, and 48 kB of program memory [21]. The operating frequency of the radio can be programmed in steps of 1 MHz. We exploit this feature to tune the radio's operating frequency to the center frequency of a specific Wi-Fi channel.

Our DEVCNT prototype uses a 580 ms detection window. Nodes report their scan counts with a reporting interval of 5 s, and the smartphone count estimations are based on a 3-minute counting window. At the end of each reporting interval, we allocate 122.5 ms for letting the sink first initiate a Glossy flood [9] to keep the nodes time-synchronized, and then collect a 1-byte scan count from each node using Chaos [16].

VII. EVALUATION

This section evaluates DEVCNT in controlled experiments and a real-world trial. We start by investigating in Sec. VII-A the accuracy with which DEVCNT detects active Wi-Fi scans, with and without interference and depending on the distance between the smartphones and a ZigBee device. In Sec. VII-B, we assess the accuracy of DEVCNT's smartphone count estimations for different numbers of Wi-Fi enabled smartphones. Finally, in Sec. VII-C, we report on DEVCNT's performance during a short-term deployment in a large lecture hall.

Table I. SMARTPHONES AND OPERATING SYSTEMS USED IN THE EXPERIMENTS OF SEC. VII-A.

Model	Operating system
Samsung Galaxy Nexus	Android 4.2.1
Samsung Galaxy S II	Android 4.1.2
Samsung Galaxy S3 Mini	Android 4.1.2
HTC Desire	Android 2.3.7
iPhone 4	iOS 7
iPhone 5	iOS 7

A. Active Scan Detection Rate

We first evaluate the accuracy of scan detections, which is a key prerequisite to obtain accurate smartphone counts.

Setup. To avoid any bias in the measurements due to uncontrolled interference sources, we conduct these experiments in an environment where we verified with a spectrum analyzer that there is no interference in the 2.4 GHz ISM band. In particular, we conduct the indoor experiments in Sec. VII-A1 and Sec. VII-A2 in an underground garage, and the outdoor experiment in Sec. VII-A2 in an open field.

We use six different smartphones that run three different versions of Android and iOS 7, as listed in Table I. On Android phones, we install a dedicated application that triggers active Wi-Fi scans with a period of 20 s. Because a similar application is not available for iOS, we manually trigger active Wi-Fi scans on these two phones by retrieving the list of available APs.

We use one TelosB node connected to a laptop. The node reads out the RSSI register of the CC2420 radio at the IEEE 802.15.4 symbol rate of 62.5 kHz and logs them over the serial port. To obtain ground truth, we put the Wi-Fi card on the laptop in monitor mode and use Wireshark to log every observed Wi-Fi frame. Both the ZigBee radio and the Wi-Fi radio are tuned to an operating frequency of 2.422 GHz, corresponding to IEEE 802.11 channel 7.

Methodology and metric. We evaluate the performance of DEVCNT's signal processing pipeline (see Fig. 2) in terms of *scan detection rate*, that is, the number of active Wi-Fi scans correctly detected by DEVCNT from ZigBee RSSI traces over the number of active Wi-Fi scans in the Wireshark logs.

1) Impact of Interference: We first look at the robustness of the active scan detection rate against several typical interference sources in the 2.4 GHz band.

Setting. We consider five different interference settings in distinct 10-minute runs: (*i*) no interference, (*ii*) Wi-Fi TCP traffic, (*iii*) Wi-Fi UDP streaming, (*iv*) Bluetooth, and (*v*) ZigBee. We place the TelosB at a distance of 10 m from the smartphones. The interferers are 14 m and 10 m away from the TelosB and the smartphones, respectively.

For Wi-Fi settings (*ii*) and (*iii*), we associate a second laptop to an AP that also operates on channel 7. We generate TCP traffic by repeatedly accessing a web page with an HTTP client on this laptop. After each access, the HTTP client waits for a random interval between 1 and 5 s before it requests the next web page. We use Iperf to generate a UDP stream with a bit rate of 400 kbit/s, which is the rate of a typical Internet video stream.[2] We play music over a Bluetooth headset in setting (*iv*). In setting (*v*), we let another TelosB node transmit 30-byte packets with a random interval in [0.5, 1.5] s.

For every interference setting, we extract from the RSSI trace all sequences of the size of a detection window (580 ms) that contain a scan and label those sequences as class "contains a scan" (Y). To get a representative set of sequences

[2] https://support.google.com/youtube/answer/2853702

Table II. SCAN DETECTION PERFORMANCE WITH AND WITHOUT INTERFERENCE FROM VARIOUS INTERFERENCE SOURCES.

Interference setting	**Scan detection rate** (avg, std)
No interference	(100.0 %, 0.0 %)
Wi-Fi TCP traffic	(99.3 %, 1.3 %)
Wi-Fi UDP stream	(99.5 %, 0.9 %)
Bluetooth headset	(99.3 %, 1.7 %)
ZigBee device	(99.8 %, 0.6 %)

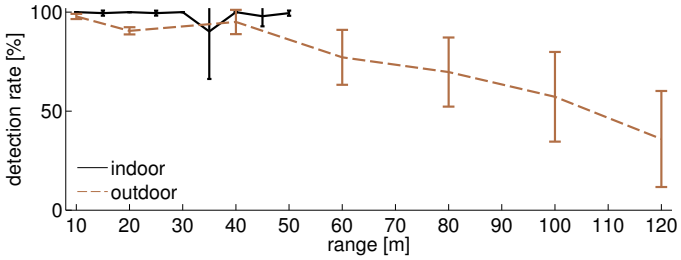


Figure 11. Average and standard deviation of scan detection rate in DEVCNT across six smartphones placed at different distances from a ZigBee device, measured both indoors and outdoors. DEVCNT *can reliably detect active Wi-Fi scans from a smartphone that is approximately 50 m away.*

that "contain no scan" (N), we extract the same number of sequences from random positions in the trace without a scan. For each sequence, we calculate the two features f_p and f_l used by our classifier (see Sec. V-C). To assess the classification performance, we use 10-fold cross validation, each fold containing features from all five interference settings with similar frequency. In total, we evaluate 2784 sequences out of which 50 % contain an active Wi-Fi scan.

Results. Table II lists the scan detection rates achieved by DEVCNT in the five interference settings. We see that DEVCNT achieves an average accuracy above 99 % across the board. This shows that DEVCNT reliably detects active scans despite interference from various common interference sources.

Despite this impressive accuracy, we note that Wi-Fi interference hardly presents a significant problem for DEVCNT in a real deployment, because channel assignment in Wi-Fi production networks mostly focuses on a few non-overlapping channels [1]. Since probes are sent on each Wi-Fi channel during an active scan, tuning DEVCNT to the center frequency of an unused Wi-Fi channel is therefore a viable option to reduce, or completely remove, the influence of Wi-Fi interference.

For the remaining experiments, we use the traces from this interference experiment to train our classifier, that is, to obtain the threshold on the product of the two features f_p and f_l.

2) Impact of Distance: Next, we study how the scan detection rate is affected by the distance between the smartphones and the DEVCNT node.

Setting. We place the smartphones at different distances from the TelosB node. Outdoors, we check distances between 10 and 120 m; indoors, we are only able to go from 10 m up to 50 m due to the limited size of the underground garage. We place a second laptop running Wireshark next to the smartphones to capture all probes they emit, that is, the ground truth. We perform a 10-minute run at each distance.

Results. Fig. 11 shows the average scan detection rate across all six smartphones as a function of their distance to the TelosB; error bars indicate the standard deviation. We see that the average scan detection rate is above 90 % up to a distance of 50 m, and shows very little variations between the different smartphones. The performance drop at 35 m in the indoor experiment is presumably due to multipath fading caused by

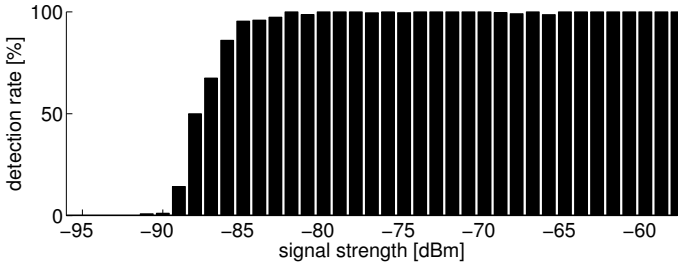


Figure 12. Average scan detection rate against received signal strength. *Active Wi-Fi scans with a received signal strength above -85 dBm are detected with a very high probability that is close to 100 %.*

the geometry of the underground garage. Beyond 50 m, the scan detection rate decreases steadily to 36 % at a distance of 120 m. We also note that the scan detection rate varies more between phones at larger distances: some smartphones have a larger Wi-Fi transmission range than others.

To further explain these results, we plot in Fig. 12 the scan detection range against the received signal strength. We see a pronounced drop for signals below -85 dBm. This suggests that the scan detection rate largely depends on the signal levels.

In summary, we learn from these experiments that a single DEVCNT node is sufficient to reliably detect active scans from smartphones that roam about, for example, a large store. To cover areas that extent beyond 50 m, however, more DEVCNT nodes should be deployed to obtain accurate estimates.

B. Accuracy of Smartphone Count Estimations

In this experiment, we evaluate the accuracy of DEVCNT's real-time smartphone count estimations.

Setting. We use again the underground garage to avoid any bias in our measurements due to uncontrolled interference. We place a TelosB node running DEVCNT in the middle of a 20 × 20 m area, and let it listen on Wi-Fi channel 6 to detect active scans. The node reports its scan counts every 5 seconds to a base station, where DEVCNT estimates the number of Wi-Fi enabled smartphones in the area. A laptop running Wireshark captures ground truth. In addition, we install one AP operating on channel 8 to mimic a realistic setup.

We use in total 31 smartphones, which run either iOS or Android. Including the smartphones from the previous experiments, there are 9 different models from 4 different vendors, representing a good mix of currently available smartphones. During the experiment, we change the number of smartphones inside the garage. Starting from 0, we add 10 phones after 15 min, another 10 after 30 min, and the remaining 11 after 45 min. Then, after 60 min, we start to actively use 10 of the 31 phones, unlocking the screen, scrolling through menus, or playing music. After 75 min, we stop using the phones. Finally, after 90 min, we start to remove phones: first a batch of 15 phones, and 15 min later the remaining 16 phones.

Meanwhile, DEVCNT estimates the number of Wi-Fi enabled smartphones every 5 seconds, based on the scan counts reported by the TelosB node. To this end, we use three different average scan rates for a 3-minute counting window: 2.70, 3.48, and 4.26. These correspond to the lowest and highest average scan rates observed in the datasets shown in Fig. 4 (3.48 is the average of 2.70 and 4.26). We compute the accuracy of the smartphone count estimations by comparing DEVCNT's estimates against ground truth obtained from the Wireshark logs when applying the same 3-minute counting window.

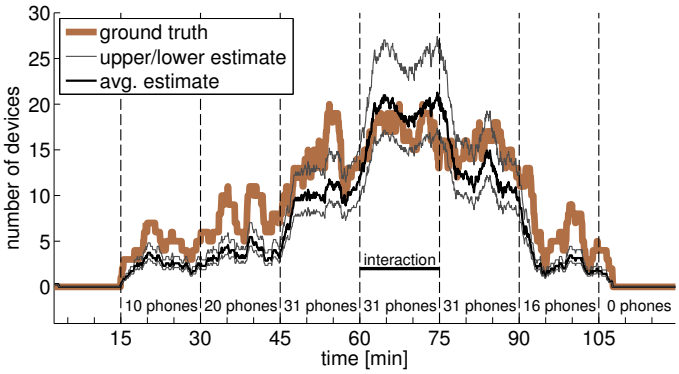


Figure 13. Estimated and real number of Wi-Fi enabled devices as smartphones are being added and removed over time. DEVCNT *provides accurate smartphone count estimates, achieving an average accuracy of up to 91 %.*

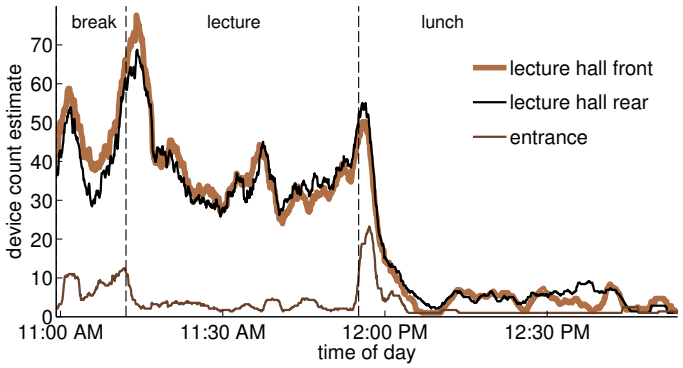


Figure 14. Estimated number of smartphones in a real world trial.

Results. Fig. 13 plots DEVCNT's estimates and ground truth over time. We first note that the number of smartphones that are physically present inside the garage is roughly double the number of smartphones in the *ground truth*. We attribute this to the fact that several phones performed very few active scans in the experiment, with intervals much larger than the 3-minute counting window we use. In fact, 9 smartphones did issue less than 3 active scans during the whole experiment, predominately such with an Android version of 2.3.7 or lower. We could not expect this behavior based on our analysis of large real-world datasets in Sec. IV, as there is no information available on silent smartphones.

Nevertheless, we observe from Fig. 13 that DEVCNT's estimates closely match ground truth as smartphones are being added and removed. When considering the average estimate, DEVCNT achieves an accuracy of 68.9 % throughout the entire 2-h experiment, which corresponds to an average absolute error of 3.0 smartphones. As mentioned earlier, we expect DEVCNT's estimates to be more accurate when the number of smartphones is higher. Our results confirm this expectation: Considering the interval between 45 min and 75 min in which all 31 smartphones are present, and by taking into account the different activity patterns, DEVCNT achieves an average accuracy of 87.3 % (1.8 average absolute error) while all phones are in stand-by mode, and an average accuracy of 90.5 % (1.65 average absolute error) while 10 of the smartphones are active.

As one would expect, DEVCNT is less accurate than a Wi-Fi-based solution (such as [5]), simply because it has less information at its disposal. In return, DEVCNT preserves by design the privacy of smartphone users, which is a strong asset when it comes to acceptance by law and the population [23], [24]. Nevertheless, an accuracy of 70–90 % is sufficient for many applications we target, and comparable to what has been reported in the literature, for example, when counting smartphones using audio tones [15] or when fingerprinting a Wi-Fi driver [10]. We thus conclude that DEVCNT provides accurate estimates on the number of Wi-Fi enabled smartphones if the mobility and usage profile of smartphones is known, which is a reasonable assumption in many applications [5], [11], [24].

Finally, we also logged performance counters throughout this experiment to study the processing overhead on the TelosB node. We find that the TelosB node was processing for only 1.7 % of the time. This shows that our novel signal processing pipeline is amenable to an efficient implementation even on severely resource-constrained embedded devices.

C. Real-world Test Run

In a final real-world trial, we deploy DEVCNT in a lecture hall to show the applicability of the system in an uncontrolled environment. Such an environment presents a significant challenge for DEVCNT because of two reasons. First, in a larger room with many people carrying smartphones there are many more signals to process on the nodes and hence processing time might be high and reach the limits of the system. Second, the system is exposed to interference originating from different surrounding devices that might emit patterns that have not been considered in the training step of the classifier.

Setting. We install a multi-hop wireless network consisting of three DEVCNT nodes, a relay node, and a sink. We place two nodes inside the lecture hall, in the front and in the rear area; we put the third node outside at one of the two main entrances. The lecture hall has a size of 20 by 25 m. We observe APs on channels 1, 6, and 11. To minimize the interference from the APs, we let all DEVCNT nodes listen on Wi-Fi channel 8.

We let the system run from 11 AM to 1 PM, so we observe a morning break, half a lecture, and a lunch break. At 11:30 AM we count by hand 111 students inside the lecture hall. While we also set up two laptops running Wireshark, we note that in this real-world setting *it is impossible to reliably determine the ground truth*. This is due to vastly different reception ranges of Wi-Fi and ZigBee radios: While a Wi-Fi receiver may be able to hear a probe from a weak sender (*e.g.*, located in another room), probes from this sender on adjacent channels are often too weak to be heard by a ZigBee node at the same distance.

Results. Fig. 14 shows the estimated smartphone counts over time for the 3 DEVCNT nodes. Although we lack ground truth, we see that DEVCNT's estimates throughout the deployment closely match our expectations and visual on-site observations. For instance, during the initial break at about 11:05 AM there is a drop in the estimated smartphone counts, because a few students leave the lecture hall. The peaks at the beginning and at the end of the lecture are due to students using their phones more intensively, which leads to smartphones performing more active scans. Furthermore, as expected, DEVCNT's estimates remain fairly stable during the lecture, and afterward drop to numbers close to zero as almost all students leave for lunch. Looking at the node at the entrance, we see that it generally sees fewer smartphones, yet the periods where students enter or leave the hall before and after the lecture are clearly visible.

We further note that both nodes inside the lecture hall see about the same number of smartphones. This is in accordance with the findings in Sec. VII-A2: Since one DEVCNT node can reliably detect devices within a radius of 50 m, a single node

would have been sufficient to cover the entire lecture hall.

These results show that DEVCNT can sustain signals from hundreds of Wi-Fi transmitters, as evident from our Wireshark logs, and delivers meaningful estimates in a real-world trial that resembles, for example, a retail or indoor concert setting.

VIII. DISCUSSION

DEVCNT estimates in real-time the number of Wi-Fi enabled smartphones within a given area. By detecting active Wi-Fi scans from RSSI traces, DEVCNT obtains these counts in a fully-passive, non-invasive, and privacy-preserving manner.

Any system relying on externally observable properties for counting misses those smartphones that do not disclose these properties, and DEVCNT is no exception. As such, like other solutions from academia [20] and industry [5], DEVCNT can only see phones that have Wi-Fi enabled and may count Wi-Fi transmitters other than phones. However, in the environments we target including open streets, shops, and train stations, the fraction of laptops and tablets is typically rather smaller.

DEVCNT supports deployments across large areas through multi-hop communications. In those scenarios, multiple ZigBee devices may detect and then count the same smartphone. A practical approach to ameliorate this over-counting problem would be to carefully select the locations of the ZigBee devices so as to reduce areas of overlapping reception ranges. Another possibility would be to exploit the fact that DEVCNT devices are time-synchronized, so an active scan that is detected by different devices at the same time likely originates from the same smartphone and could be accounted for only once. We intend to explore this idea in our future work.

DEVCNT preserves the privacy of smartphone users as it cannot identify individual phones. While this is arguably a desirable property, it leaves DEVCNT with no other option than to estimate the smartphone counts based on statistical information about the average active scan rate. DEVCNT provides accurate estimates whenever the observed population of smartphones behaves according to the expectations, for example, in terms of their degree of mobility and how frequently the smartphones are being used. If phones behave sharply differently, however, DEVCNT's estimates may become less accurate. Nevertheless, we found in our tests that phases of unusual behavior typically last for only a limited amount of time as visible, for example, in Fig. 14 right after the break. In that sense, DEVCNT is similar to participatory sensing approaches [27], where the available GPS data fluctuate because smartphone users have full control over the application and are free to opt out at any time.

The flexibility of battery-powered nodes is not for free: To capture as many active scans as possible, all DEVCNT nodes need to have their radio continuously turned on. In this case, a node powered by two AA batteries would last for a week, which is fine for short deployments (*e.g.*, during a concert). One way to save energy would be to turn off the radio for extended periods of time when there is low Wi-Fi activity, such as during the night or outside of a shop's opening hours. We leave such energy considerations for future work.

Overall, DEVCNT represents a new point in a multi-dimensional design space, trading some fidelity of the smartphone counts for full privacy of the smartphone users. Corresponding to this promising design point is a large number of application scenarios, ranging from crowd management [11] through public transport and event planning [4] to customer and visitor surveys [5], where DEVCNT could be highly beneficial.

IX. RELATED WORK

Our work on DEVCNT is related to prior efforts on leveraging the proliferation of smartphones for crowd counting and exploiting the interference between Wi-Fi and ZigBee.

Leveraging smartphones for crowd counting. Existing solutions employ different observable properties of a smartphone to count the number of people in an area or estimate the density of crowds. Such properties include audio tones [15], GPS coordinates [27], Bluetooth scans [26], and Wi-Fi probes [5], [20]. Conceptually, we can classify these solutions along three dimensions: privacy, invasiveness, and passiveness.

Both research [20] and commercial [5] systems exist that directly eavesdrop on Wi-Fi probes, using existing APs and/or dedicated Wi-Fi monitors. Being able to demodulate and decode Wi-Fi frames, these systems can easily identify and track individual smartphones based on the unique MAC addresses embedded in each probe. Although anonymization techniques such as MAC address hashing are apparently used [5], these systems may still be exploited (*e.g.*, by an attacker) to compromise the privacy of the smartphone users, who possess no means to "see" that they are being observed. Turning off Wi-Fi is therefore the only practical solution to guard against such impairment, but this may impact user experience [3].

Another class of approaches requires to modify the smartphone itself, for example, by installing and running a dedicated application. The system presented in [15] uses the built-in microphones to count smartphones by letting them exchange bit patterns encoded in audio tones. Others estimate crowd densities based on GPS data [27] or the number of discovered devices by Bluetooth scans [26]. These approaches are invasive and rely on the voluntary and enduring cooperation of users to produce meaningful estimates. Finally, [20] shows that it is possible to solicit more probe transmissions from unmodified smartphones to improve tracking performance.

Unlike these prior works, DEVCNT takes a fully-passive, non-invasive, and privacy-preserving counting approach. This approach relies on DEVCNT taking advantage of interference between ZigBee and Wi-Fi, similar to other systems that are however designed for different purposes, as discussed next.

Exploiting interference between ZigBee and Wi-Fi. SoNIC classifies interference in the 2.4 GHz band into "Wi-Fi," "microwave," or "Bluetooth" based on RSSI information available on a ZigBee device [12]. SpeckSense and ZiFi exploit interference from Wi-Fi beacon frames, which are easier to detect than active scans because they exhibit a more rigid periodicity. SpeckSense processes RSSI information on a TelosB device in order to avoid Wi-Fi interference [14]. ZiFi uses a built-in or external ZigBee radio to help a smartphone or laptop discover Wi-Fi APs in a more energy-efficient manner [29]. Different from DEVCNT, ZiFi benefits from ample resources of the host device compared to the limited memory and compute power of a low-power ZigBee mote. WizNet uses interference from probes, beacons, and other Wi-Fi traffic to monitor the spatio-temporal performance variations of Wi-Fi installations [28]. Similar to DEVCNT, WizNet uses, among other techniques, the discrete autocorrelation to identify probes from RSSI samples. However, unlike DEVCNT, WizNet sends compressed RSSI traces to a more capable sink for computing the autocorrelation offline. DEVCNT shows that active scan detection can indeed be performed online on mote-class devices, thereby reducing communication energy costs and bandwidth requirements by sending only the minimum amount of data to the sink.

X. Conclusions

We have presented DevCnt, the first system that supports real-time counting of unmodified Wi-Fi enabled smartphones while preserving the privacy of the smartphone owners. Using novel signal processing algorithms that execute on a multi-hop network of ZigBee devices, DevCnt detects and counts active Wi-Fi scans performed by smartphones based on characteristic patterns in RSSI traces. Combining these counts with statistical information about the average active scanning rate, DevCnt faithfully estimates the number of Wi-Fi enabled smartphones. DevCnt trades some fidelity in the smartphone count estimations for improved privacy. Results from controlled and real-world experiments show that DevCnt provides estimates with an accuracy of up to 91 %. We thus maintain that DevCnt is a viable and promising solution for low-cost, real-time crowd counting in a broad spectrum of innovative applications.

Acknowledgements

We thank Marco V. Barbera for providing us with helpful information about the datasets used in Sec. IV. The work presented in this paper was scientifically evaluated by the SNSF, and financed by the Swiss Confederation and by nano-tera.ch.

References

[1] A. Akella, G. Judd, S. Seshan, and P. Steenkiste. Self-management in chaotic wireless deployments. In *Proc. of the 11th Intl. Conf. on Mobile Computing and Networking (MobiCom)*, 2005.

[2] M. V. Barbera, A. Epasto, A. Mei, S. Kosta, V. C. Perta, and J. Stefa. CRAWDAD data set sapienza/probe-requests (v. 2013-09-10). Downloaded from http://crawdad.org/sapienza/probe-requests/, Sept. 2013.

[3] M. V. Barbera, A. Epasto, A. Mei, V. C. Perta, and J. Stefa. Signals from the crowd: Uncovering social relationships through smartphone probes. In *Proc. of the Internet Measurement Conf. (IMC)*, 2013.

[4] C. C. Cheong and R. To. Household interview surveys from 1997 to 2008—a decade of changing travel behaviours, May 2010. Online at http://goo.gl/aw3WFV.

[5] Cisco Systems. White paper: CMX Analytics, Apr. 2014.

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

[7] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proc. of the 1st IEEE Workshop on Embedded Networked Sensors (Emnets)*, 2004.

[8] Ericsson AB. Ericsson mobility report, June 2014.

[9] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient network flooding and time synchronization with Glossy. In *Proc. of the 10th Intl. Conf. on Information Processing in Sensor Networks (IPSN)*, 2011.

[10] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. Van Randwyk, and D. Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. In *Proc. of the 15th USENIX Security Symposium (SS)*, 2006.

[11] D. Helbing, L. Buzna, A. Johansson, and T. Werner. Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. *Transportation Science*, 39(1):1–24, 2005.

[12] F. Hermans, O. Rensfelt, T. Voigt, E. Ngai, L.-A. Norden, and P. Gunningberg. SoNIC: Classifying interference in 802.15.4 sensor networks. In *Proc. of the 12th Intl. Conf. on Information Processing in Sensor Networks (IPSN)*, 2013.

[13] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Trans. Syst., Man, Cybern. C*, 34(3):334–352, 2004.

[14] V. Iyer, F. Hermans, and T. Voigt. Detecting and avoiding multiple sources of interference in the 2.4 GHz spectrum. In *Proc. of the 12th Intl. Conf. on Embedded Wireless Systems and Networks (EWSN)*, 2015.

[15] P. G. Kannan, S. P. Venkatagiri, and M. C. Chan. Low cost crowd counting using audio tones. In *Proc. of the 10th ACM Conf. on Embedded Networked Sensor Systems (SenSys)*, 2012.

[16] O. Landsiedel, F. Ferrari, and M. Zimmerling. Chaos: Versatile and efficient all-to-all data sharing and in-network processing at scale. In *Proc. of the 11th ACM Conf. on Embedded Networked Sensor Systems (SenSys)*, 2013.

[17] C.-J. M. Liang, N. B. Priyantha, J. Liu, and A. Terzis. Surviving Wi-Fi interference in low power ZigBee networks. In *Proc. of the 8th ACM Conf. on Embedded Networked Sensor Systems (SenSys)*, 2010.

[18] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel. FlockLab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In *Proc. of the 12th Intl. Conf. on Information Processing in Sensor Networks (IPSN)*, 2013.

[19] C. C. D. Loh, C. Y. Cho, C. P. Tan, and R. S. Lee. Identifying unique devices through wireless fingerprinting. In *Proc. of the 1st ACM Conf. on Wireless Network Security (WiSec)*, 2000.

[20] A. B. M. Musa and J. Eriksson. Tracking unmodified smartphones using *Wi-Fi* monitors. In *Proc. of the 10th ACM Conf. on Embedded Networked Sensor Systems (SenSys)*, 2012.

[21] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proc. of the 4th Intl. Conf. on Information Processing in Sensor Networks (IPSN)*, 2005.

[22] Texas Instruments. CC2420 datasheet, 2014.

[23] The Guardian. City of London Corporation wants 'spy bins' ditched, Aug. 2013. Online at http://www.theguardian.com/world/2013/aug/12/city-london-corporation-spy-bins.

[24] The Local Denmark. Copenhagen to roll out new smart traffic system, Feb. 2015. Online at http://www.thelocal.dk/20150202/copenhagen-to-roll-out-new-smart-traffic-systems.

[25] The New York Times. Stampede at german music festival kills 18, July 2010. Online at http://www.nytimes.com/2010/07/25/world/europe/25germany.html.

[26] J. Weppner, P. Lukowicz, U. Blanke, and G. Tröster. Participatory Bluetooth scans serving as urban crowd probes. *IEEE Sensors Journal*, 14(12):4196–4206, 2014.

[27] M. Wirz, T. Franke, E. Mitleton-Kelly, D. Roggen, P. Lukowicz, and G. Tröster. CoenoSense: A framework for real-time detection and visualization of collective behaviors in human crowds by tracking mobile devices. In *Proc. of the European Conf. on Complex Systems (ECCS)*, 2012.

[28] R. Zhou, G. Xing, X. Xu, J. Wang, and L. Gu. WizNet: A ZigBee-based sensor system for distributed wireless LAN performance monitoring. In *Intl. Conf. on Pervasive Computing and Communications (PerCom)*, 2013.

[29] R. Zhou, Y. Xiong, G. Xing, L. Sun, and J. Ma. ZiFi: Wireless LAN discovery via ZigBee interference signatures. In *Proc. of the 16th Intl. Conf. on Mobile Computing and Networking (MobiCom)*, 2010.

Appendix

We further detail the algorithm used in DevCnt to efficiently compute the *sum of autocorrelations* feature f_p.

We know from Sec. V-C that this feature is defined as

$$f_p = \sum_{j=1}^{w-a} \sum_{k=j+a}^{\min(w,j+b)} x_j x_k. \qquad \text{(3 revisited)}$$

Here, the x_i are elements of a binary vector $X \in \{0,1\}^w$, indicating for each sample in the detection window of size w whether there is a signal present or not. The symbols a and b denote the limits for the lags of the autocorrelation function. In addition, we define l as the increasingly ordered set of indexes of value changes in the binary vector X, that is, $l := \{i | x_i \neq x_{i+1}\}$. In the following, we use the notation l_s to refer to the element in l at position s. Note that l changes from 0 to 1 at odd positions, while it changes from 1 to 0 at even positions.

A. Illustrative Example

We motivate our algorithm with the help of the example illustrated in Fig. 15. Here, X contains four signals, which results in a set l of size 8, as shown on the vertical and the horizontal axes. The elements to be summed up, $x_j x_k$, are laid out in a 2-dimensional bitmap. Dark boxes indicate elements that contribute to the sum, that is, where both x_k and x_j are 1. The limits of the covered area are determined by $[a, b]$, the interval of the considered lags of the autocorrelation.

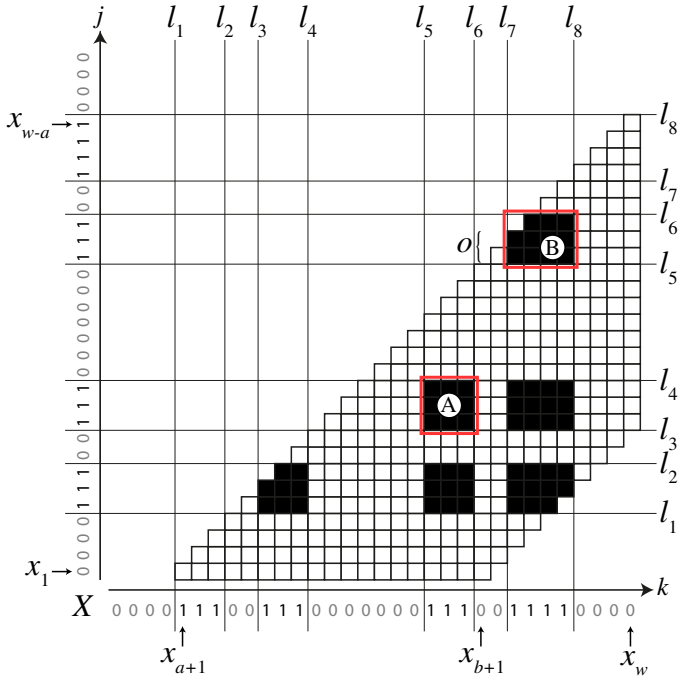


Figure 15. Sum of autocorrelations for an example binary vector X. *The boxes represent all the $x_j x_k$ that are added up in* (3)*. Black boxes are the elements that contribute to the sum. The area of adjacent black boxes can be calculated without iterating over every element therein.*

To compute the feature f_p, we add up the areas of (partial) black rectangles formed by one or more adjacent black boxes. The area of the completely black rectangle Ⓐ is the product of its length and width, that is, $\mathrm{Area}(\text{Ⓐ}) = (l_6 - l_5)(l_4 - l_3)$. The area of partially black rectangles, such as Ⓑ, can also be calculated without iterating over every element by considering the offset o of the diagonal through the rectangle.

For our purpose, converting X into the set of value change indexes l reduces both space requirements and computational complexity: A detection window of 580 ms sampled with a resolution of 16 μs results in a binary vector X of 36250 bits (8.53 kB). According to our experiments, the number of signals in a detection window is rather small, and therefore the set l is significantly smaller than $|X|$. The reduced input size also reduces the complexity of the algorithm as we do not need to iterate over all samples in X, but only over the elements in l.

B. Algorithm and Pseudocode

Algorithm 1 calculates f_p. It iterates over both dimensions j and k using l, and sums up the rectangles delimited by the corners $(l_k, l_j)(l_{k+1}, l_{j+1})$. An odd position in l indicates the start of a signal, and an even position indicates the end of a signal. Thus, for every iteration, the index variables j and k are incremented by 2 to select the next rectangle. We distinguish three cases: (*i*) a fully contained rectangle (line 7), (*ii*) a rectangle partially outside on the lower end of k (line 11), and (*iii*) a rectangle partially outside the upper end of k (line 15). For (*i*) the number of elements is the product of the width and the height. For (*ii*) and (*iii*), the function PartialRectangle() shown in Algorithm 2 additionally uses the offset o to calculate the number of elements that fall into this partial rectangle. In each iteration, the contribution of the current rectangle is added to the final result f_p.

Algorithm 1 Compute the *Sum of Autocorrelations* Feature

Input: l: indexes of level changes, a b: limits for lag
Output: f_p: sum of autocorrelation between a and b

1: $f_p \leftarrow 0$
2: $j \leftarrow 1$
3: **while** $j < |l|$ **do**
4: $\quad k \leftarrow 1$
5: $\quad$ **while** $k < |l|$ **do**
6: $\quad\quad$ **if** $l_k \geq l_{j+1} + a - 1$ **and** $l_{k+1} \leq l_j + b + 1$ **then**
7: $\quad\quad\quad f_p \leftarrow f_p + (l_{k+1} - l_k)(l_{j+1} - l_j)$
8: $\quad\quad$ **else**
9: $\quad\quad\quad$ **if** $l_k < l_{j+1} + a - 1$ **and** $l_{k+1} > l_j + a$ **then**
10: $\quad\quad\quad\quad o \leftarrow l_k - l_j - a + 1$
11: $\quad\quad\quad\quad f_p \leftarrow f_p + \mathrm{PartialRectangle}(l_{k+1} - l_k, l_{j+1} - l_j, o)$
12: $\quad\quad\quad$ **else**
13: $\quad\quad\quad\quad$ **if** $l_k > l_{j+1} + b$ **and** $l_{k+1} > l_j + b + 1$ **then**
14: $\quad\quad\quad\quad\quad o \leftarrow l_{j+1} - l_{k+1} + b + 1$
15: $\quad\quad\quad\quad\quad f_p \leftarrow f_p + \mathrm{PartialRectangle}(l_{k+1} - l_k, l_{j+1} - l_j, o)$
16: $\quad\quad\quad\quad$ **end if**
17: $\quad\quad\quad$ **end if**
18: $\quad\quad$ **end if**
19: $\quad\quad k \leftarrow k + 2$
20: $\quad$ **end while**
21: $\quad j \leftarrow j + 2$
22: **end while**

Algorithm 2 PartialRectangle(o, w_1, w_2)

Input: w_1, w_2: width and height of area, o: offset of diagonal
Output: A: number of elements contained in area

1: **if** $o > 0$ **then**
2: $\quad A_0 \leftarrow w_1 o$
3: $\quad d_1 \leftarrow w_1 - 1$
4: **else**
5: $\quad A_0 \leftarrow 0$
6: $\quad d_1 \leftarrow w_1 + o - 1$
7: **end if**
8: $d_2 \leftarrow \max(1, w_1 - (w_2 - o))$
9: $A \leftarrow A_0 + (d_1 + d_2)(d_1 - d_2 + 1)/2$