# Incremental Wi-Fi Scanning for Energy-Efficient Localization

Niels Brouwers Marco Zuniga Koen Langendoen
Embedded Software Group
Delft University of Technology
{n.brouwers, m.zuniga, k.g.langendoen}@tudelft.nl

***Abstract*—Wi-Fi based localization has proven to be a compelling alternative to GPS for mobile devices. But Wi-Fi scanning consumes a large amount of energy on smartphones because they perform full scans, i.e. *all* the channels in their band(s) are visited. This inefficient behavior greatly reduces battery life, raising the threshold for user acceptance. We propose a novel, incremental approach that reduces the energy consumption of Wi-Fi localization by scanning just a few selected channels. We evaluate our incremental scanning approach on eight Android devices using traces from five test subjects. Our results show that, compared to full scans, incremental scanning can reduce the energy consumption between 20.64% and 57.79%. The modern smartphones included in our study all show an energy reduction of at least 40%.**

## I. Introduction

Our work aims at reducing the energy consumption of Wi-Fi scanning, a popular localization alternative to GPS in urban areas. In Wi-Fi scanning, a list of access points and their corresponding signal strengths serves as a fingerprint that uniquely identifies the user's location. Commercial services exist that translate the scan results into geographical coordinates, such as Google's geolocation service [1], and Skyhook [2]. The accuracy of the final positions approaches that of GPS in urban environments, and Wi-Fi scanning has the advantage that it also works indoors. The problem with current Wi-Fi scanning algorithms is that they are designed to discover *all* nearby access points (APs), and as a consequence employ an exhaustive search of all available channels. Until recently, smartphones have had 11-13 channels in the 2.4 GHz spectrum available to them, but with the introduction of 802.11 a/b/g/n chipsets that add support for the 5 GHz band the total has increased to 32, significantly raising the energy cost of AP discovery. In the context of localization, the requirement that a scan must find all nearby APs can be relaxed to finding enough APs for a good location estimate. Discovering this sufficient subset usually does not require all channels to be included in the scan. The task of an energy-efficient Wi-Fi localization scanning algorithm therefore is to discover 'enough' access points while minimizing the number of channels scanned.

In this work we propose *incremental scanning*, a novel scanning technique that reduces the energy consumption of Wi-Fi localization by scanning the available channels one-by-one and terminating early once a sufficient subset of APs has been discovered. We identify three mechanisms that help achieve this goal. First, we show that there is a certain number of access points beyond which adding more does not significantly improve localization accuracy, which means that a scan may be terminated early as soon as this critical mass has been reached. Second, from our experiments we found that approximately three-fourths of all access points reside on just three channels, and that scanning the channels in order of popularity greatly speeds up the discovery process. Third, mobility studies have shown that users stay in the same location 89 % of the time [3]. We found that scanning a small number of channels (i.e. one or two) is enough to determine that the user has not moved since the previous scan. The results of our user-study in Section IV show that the three mechanisms underlying the incremental scanning policy only trade off a bit of accuracy for a serious reduction in energy consumption.

## II. Scanning Cost

When Wi-Fi scanning is used as a localization primitive an exhaustive scan is often wasteful as a subset of the available channel spectrum yields sufficient information. In this section we show the inefficiency of full Wi-Fi scanning and investigate the potential for energy savings due to incremental scanning.

Our analysis includes eight Android smartphones introduced between 2008 and 2012. Table I lists the devices included in our study. The devices span five years, four vendors, five different Wi-Fi chipsets, and seven SoCs (System on Chip). Note that this list includes the HTC Dream, which has been used extensively in earlier work, and is known under multiple monikers, including the 'Android Dev Phone' (ADP) and G1.

### A. Implementing Incremental Scanning

Since Android's Java API does not provide a means to selectively scan a subset of the available Wi-Fi channels, our first challenge was to implement selective scanning on the eight Android devices we work with. This is a non-trivial task given the variety of OS versions, Wi-Fi chipsets, and CPU architectures in our device pool. Changing the API directly would essentially result in a fork of the platform, requiring custom OS builds for each of the eight target devices, and breaking compatibility. Moreover, this would be much harder to reproduce. Instead, we have opted for a workaround that modifies only a single binary that can be built for each of the target devices with relative ease.

**Current architecture.** Figure 1 shows the components involved with Wi-Fi scanning. The *wpa_supplicant* is a standard Linux process responsible for discovery of, and authentication with access points. Android applications interact with the supplicant through the `WifiManager` class in the Android API, which in turn communicates with the supplicant over a socket. The supplicant contains a common core and one or more *drivers* that implement various protocols for talking to kernel drivers such as `wext` (wireless extensions), its modern replacement `nl80211`, or in the case of the HTC Dream

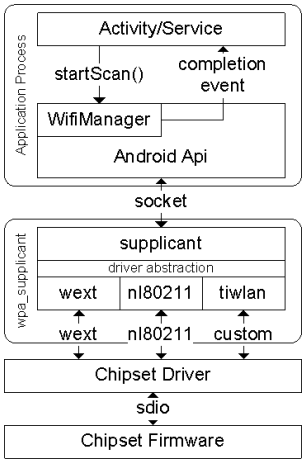


Fig. 1. The Android Wi-Fi stack. The Android application (top) interacts with the supplicant over a socket. The supplicant in turn talks to the Wi-Fi chipset driver.

and Hero, a custom driver specifically designed for the Texas Instruments WL1251 chipset (`tiwlan`).

**The problem.** An application starts a scan by calling the `WifiManager.startScan` method. An event is fired upon completion of the scan, at which time the application can retrieve the scan result from the `WifiManager` class. Chipset firmware, kernel drivers, supplicant drivers, and the supplicant itself all have support for scanning a given list of channels rather than the full spectrum. However, as the `startScan` method itself takes no arguments, there is currently no way to submit such as list from a Java application.

**Our solution.** Luckily, the `WifiManager` class does expose internal state of the supplicant that can be exploited for this purpose, which is the list of known access points. In our workaround, the list of channels that need to be scanned is encoded in a specially crafted SSID of a fictional access point. For example, if the application needs to scan channels 1, 6, and 11, it will create an AP with the SSID '@16B'. We modified the supplicant so that when it receives a scan request, it will first iterate over the list of known access points, decode a channel list if one is found, and pass it down into the supplicant driver. The HTC Dream and Hero require extra attention, since they use an older version of the supplicant, where the driver abstraction does not allow for a channel list to be passed from the common code into the supplicant driver. These devices use a custom supplicant driver, `tiwlan`, which we modified in a similar fashion.

Our solution has the advantage of being contained in a single Linux binary, `wpa_supplicant`, which can be built for a given target architecture and Android version, and pushed to a device once root access has been obtained. In the interest of reproducibility, we have made our code and compiled binaries available to the research community [4].

### B. Energy Consumption

With our modified supplicant we were able to measure the power consumption of partial scans on eight Android devices, using a Monsoon Power Monitor [5]. We developed an app that runs in the background and wakes up the phone once every ten seconds, performs a given task, and then goes back to sleep. In this way we measure the total cost of a given task, including the energy spent waking up from suspend, keeping the processor awake, and going back to suspend. We measured the consumption of four different operations: 1) performing a normal (full) access point scan, 2) scanning only a single channel, 3) sampling the accelerometer for five seconds, and 4) waking up the device and going back to suspend immediately without performing any task. During these experiments the phones were in airplane mode with Wi-Fi enabled, and set to the European Wi-Fi region.

TABLE I. OVERVIEW OF THE SMARTPHONES USED IN OUR ANALYSIS.

| Device | CPU | Year | 5 GHz |
|---|---|---|---|
| HTC Dream (G1, ADP) | Qualcomm 528MHz MSM7201A | 2008 | N |
| HTC Hero (G2) | 528MHz Qualcomm MSM7200A | 2009 | N |
| Sony Ericsson Xperia X8 | 600MHz Qualcomm MSM7227 | 2010 | N |
| Samsung Galaxy S | 1 GHz Samsung Exynos 3 | 2010 | N |
| Samsung Galaxy Nexus | 1.2 GHz TI OMAP 4460 | 2011 | N |
| Samsung Galaxy S2 | 2x1.2 GHz Samsung Exynos 4 | 2011 | Y |
| Samsung Galaxy S3 Mini | 2x1.2 GHz ST-Ericsson U8420 | 2012 | Y |
| HTC OneX+ | 4+1x1.5 GHz NVidia Tegra 3 | 2012 | Y |

Accelerometer sampling was measured as well because much previous work has focused on detecting mobility to avoid scanning when the user is passive [6]–[10]. In these approaches, the accelerometer is sampled for some seconds to detect user inactivity, in which case Wi-Fi scanning is suppressed. SensLoc [8] proposes sampling for five seconds, while EEMSS [7] samples for six. We conservatively chose the most energy efficient configuration of the two, sampling for 5 seconds at the lowest sampling rate provided by each phone. Finally, the cost of waking up the processor is studied because suspend mode is the default state of a smartphone, and coming out of- and going back to suspend introduces an overhead that presents a lower bound on the cost of any kind of periodic sensing task running in the background. Finally, we measured the cost of scanning a single channel to obtain a lower bound on the cost of incremental Wi-Fi scanning. The results are shown in Figure 2. There are large variations among different devices, both in terms of total cost of each of the operations, but also in their composition. These substantial differences in Wi-Fi scanning cost and CPU overhead illustrate that evaluating energy saving strategies requires thorough analysis across a range of devices. In the next subsections we analyze in detail the impact of these energy consumption patterns. For now it is important to note that on *all* platforms there is a significant energy gap between a full- and a single-channel scan, i.e. a large potential to benefit from incremental scans.

### C. Active State

The localization problem is usually divided into two states: *passive*, in which a user is not changing location, and *active*, when the user is mobile. This is because in the passive state the information gathered from the surrounding access points does not change significantly, and much work has focused on detecting user inactivity, e.g. using motion sensors, for the purpose of saving energy by suppressing Wi-Fi scanning. First, we focus on analyzing the impact of incremental scanning in the active state.

The de-facto approach during the active state is to perform periodic, full access point scans to obtain as much information about the user location as possible. However, as we will show in Section III-D, it is often possible to scan fewer channels while still obtaining a good location fix. The difference between a full- and single channel scan is the maximum reduction in energy that can be obtained in this way. Figure 2 shows

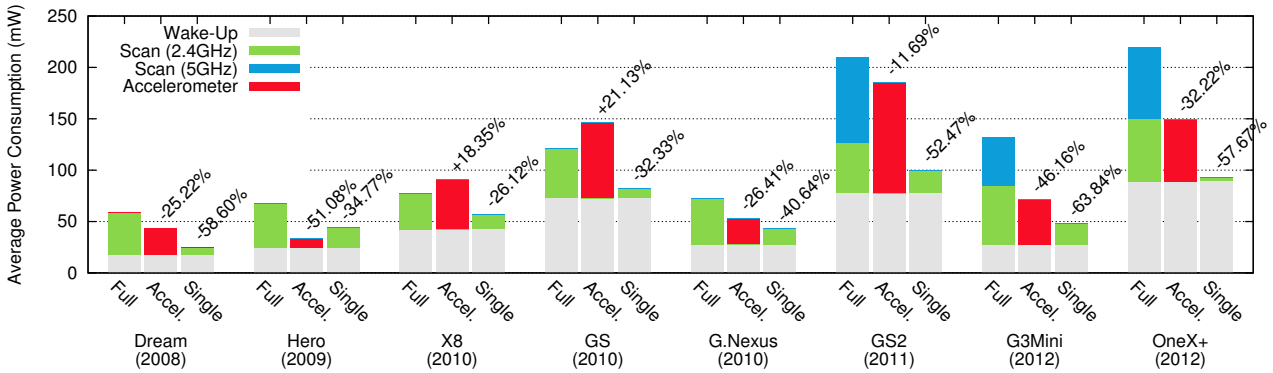


Fig. 2. Median power consumption of scanning and accelerometer sampling on various smartphones. For the Galaxy S2 and S3Mini, and the HTC OneX+ we also measured the cost of scanning the 2.4 GHz band only, which is shown as an annotation on the normal scanning cost.

that the potential savings vary from device to device, between 26.12 % on the X8, and 63.84 % on the GS3 Mini. From these numbers we can identify two trends. First, CPU overhead has increased significantly since the HTC Dream and HTC Hero devices on which much of the related work was evaluated. This can be seen by the increase in wake-up overhead, as well as the increased cost of accelerometer sampling. Scanning fewer channels reduces not only the cost of the Wi-Fi chip itself, but also the time the CPU has to be awake while waiting for the result. Second, newer devices such as the GS2, GS3 Mini, and HTC One X+ support the 802.11 a/b/g/n standard, which includes the 5 GHz band[1]. These devices include many more channels in a full scan, increasing its cost relative to a single-channel scan. The three devices that support the 5 GHz band, all present a potential energy savings of more than 50 %. Figure 3 illustrates in more detail the difference between a full- and single-channel scan on the dual-band GS2 phone. Note the long duration of the full scan as it visits the 32 available channels, as well as the large wake-up and suspend overhead, which dominates the cost of a single scan.

### D. Passive State

In the passive state, we found that first-generation devices (HTC Dream and Hero) –on which the accelerometer suppression work has been evaluated in the past– provides good savings for accelerometer sampling over Wi-Fi scanning. This is not necessarily true for newer devices. In particular, on the X8 and Galaxy S the cost of these operations is actually higher by 18.35 % and 21.13 % respectively, confirming the findings in [11]. This is because modern processors often have a larger energy overhead. This point is illustrated in Figure 4, which compares five seconds of accelerometer sampling on the GS2 and HTC Hero, devices that represent the extreme ends of the spectrum in our test. Note that the Exynos 4 chip in the GS2 requires a considerable amount of energy to come out of- and go back to the suspend state. Moreover, while the HTC Hero consumes hardly any energy idling in between samples, the GS2 shows a constant CPU overhead.

On more recent phones that are capable of using the 5 GHz spectrum, full scans are so costly that accelerometer sampling is again beneficial. But, as we will show in Section IV, scanning only one or two channels is usually enough to determine that a user has not moved, which brings energy consumption

[1]The Galaxy Nexus also supports a/b/g/n, but we were unable to make this device operate on the 5 GHz band. We attribute this to a bug in the firmware.

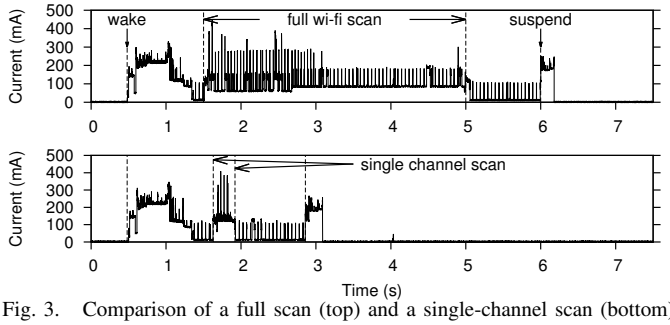


Fig. 3. Comparison of a full scan (top) and a single-channel scan (bottom) on the Samsung Galaxy S2 smartphone.

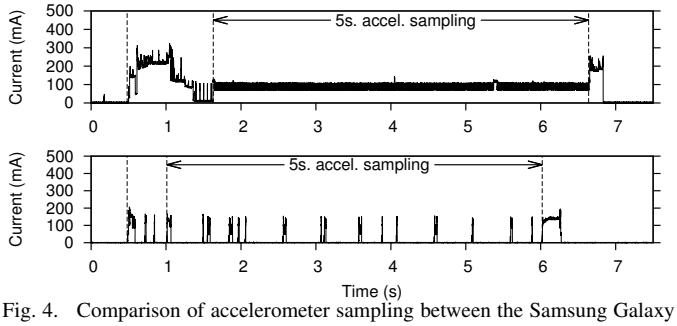


Fig. 4. Comparison of accelerometer sampling between the Samsung Galaxy S2 (top) and HTC Hero (bottom) smartphones.

during the passive state very close to that of single-channel scan. On all devices except the HTC Hero this operation consumes much less energy than accelerometer sampling, which motivates us to abandon this technique altogether and focus on methods that rely solely on Wi-Fi scanning.

### E. Energy Model

In this paper we evaluate a scanning algorithm that breaks off early when a certain amount of information has been discovered. To this end, we construct a power model $E^d_{scan}(n)$ for each device $d$ that tell us how much energy is consumed by scanning $n$ channels. Our measurements showed that the scanning cost $E^d_{scan}(n)$ is linear with the number of scanned channels $n$. This trend allows us to construct energy models for each of the devices by interpolating between the cost of a single-channel scan and that of a full scan. A special case is needed to account for the devices that support 5 GHz. For a fair comparison with 2.4 GHz devices and because our data

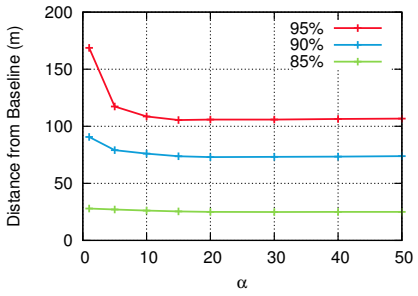


Fig. 5. Geolocation distance from GPS for varying $\alpha$, 85th, 90th, and 95th percentiles.

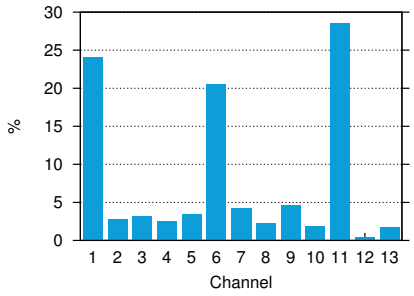


Fig. 6. Wi-Fi Channel Popularity (%).

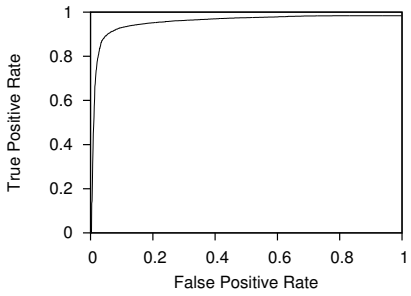


Fig. 7. Pareto front for our hysteresis classifier, as computed on our data set.

set does not contain access points in the newly operational 5 GHz band, we conservatively assume that when the 2.4 GHz needs to be fully scanned ($n = 13$), then a full sweep of the 5 GHz band is needed too. This is modeled by "jumping" immediately to the full cost of scanning both spectra, as laid out in the following model:

$$E^d_{scan}(n) = \begin{cases} E^d_{single} + (n-1) * \frac{E^d_{full,2.4} - E^d_{single}}{12} & \text{if } n \leq 12 \\ E^d_{full} & \text{if } n = 13 \end{cases}$$

$E^d_{scan}(n)$ includes the wake-up and suspend overhead on device $d$, $E^d_{full}$ is the cost of a full scan (2.4 and 5 GHz bands), $E^d_{full,2.4}$ is the cost of scanning the entire 2.4 GHz band and $E^d_{single}$ is the cost of scanning a single channel (the wake up and suspend overhead are included in this variable). The potential benefit of incremental scanning on Wi-Fi localization depends on two factors: the gap between a full and a single scan ($E^d_{full} - E^d_{single}$), and the number of channels used ($n-1$). Thus far, we have shown that the gap between full and single scans is long and worth exploring. Next, we explain the methods by which we minimize the number of channel scans.

## III. Incremental Scanning

We exploit three properties to reduce the number of scanned channels, while maintaining high localization accuracy: 1) a diminishing return in information from access points, 2) channel popularity, and 3) scan similarity during user inactivity. Since much of our analysis is data-driven, we will first describe the data set we collected; we will then discuss each of the above mentioned properties; and finally, we incorporate all our insights into our incremental scanning algorithm.

### A. Data Collection

We collected Wi-Fi access point (AP) scans and GPS location data from various urban locations in the Netherlands, Germany, Denmark, and Switzerland. The collection process was carried out by members of our research group, who were supplied with an Android smartphone and were instructed to manually annotate their activity (passive, walking, driving, etc.) through an on-screen interface. The goal was to get data both on localization accuracy as a function of AP-density, as well as to obtain ground truth about user activity.

Our data collection software issued a Wi-Fi scan every two seconds and recorded the MAC address, channel, SSID, and signal strength of each access point found. Each scan result is annotated with two latitude, longitude pairs; the most recent GPS location at the time the scan was started (if available), and a location estimate provided by Google's geolocation service [1] (obtained post-facto).

### B. Diminishing Returns

In Wi-Fi localization, access points are used as anchors to estimate the position of a user. In many anchor-based localization systems, it is known that every extra access point leads to diminishing returns in localization accuracy. This general trend holds for anchor-based systems ranging from simple centroid techniques [12] to more complex geometric methods [13]. Hence, the first question we have to ask is: how many access points $\alpha$ are needed to achieve most of the accuracy of Wi-Fi localization?

We investigate $\alpha$ by converting access point scans from our data set into geographical coordinates using Google's geolocation service [1]. We first removed duplicate locations recorded during inactive sessions to avoid skewing results, and selected only the first $\alpha$ APs for each scan. We then computed the distance between the localization result and the GPS coordinates that were collected alongside the Wi-Fi scanning data. Note that GPS does not constitute absolute ground truth, and itself is subject to localization error, but we believe it is a good approach for comparing relative results of different values of $\alpha$.

Figure 5 shows the 85th, 90th, and 95th percentiles of distance from the baseline (GPS). Note that this is not a generalizable result since the localization service depends on a database that is trained by user-submitted data, and thus subject to constant change. Moreover, performance may differ from location to location. However, it is clear that there is little improvement beyond $\alpha = 15$. In other words, scanning can be safely broken off early once 15 access points have been found.

### C. Channel Popularity

Given that only 15 access points are needed to reach a high localization accuracy, we now need to identify the channels with the highest density of access points. This will allow us to discover $\alpha$ access points more quickly and thus break off scanning earlier. Although access points may be configured to reside on any given channel, due to co-channel interference, channels 1, 6, and 11 are generally favored by system administrators and set as factory defaults. Figure 6 shows the popularity of the various Wi-Fi channels as found in our dataset[2]. For example, in a location with 20 access

[2]Data was collected with phones capable of using the 2.4 GHz spectrum only, so that 5 GHz access points are not included in out dataset.

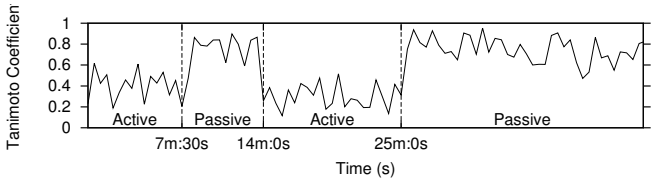


Fig. 8. Tanimoto coefficient computed over consecutive scan results, while alternating between walking and being stationary. The similarity score has a distinct range for both activities, although noise causes some overlap.

points within range (a very likely event nowadays), scanning three channels would be sufficient to reach the 15-AP mark. In general, channel popularity can differ from place to place, and is not known beforehand. But this problem can be easily solved by keeping track of the recent scanning history and use it to continuously update the ranking of channel popularity.

### D. Scan Similarity

When a person is not changing locations (passive state), consecutive scans yield similar results. This effect brings significant gains to incremental scanning because upon detecting that the first scanned channels have similar access points, the scan can terminate early. Our last goal is therefore to identify passive states as early as possible. Previous studies have used scan similarity to identify a person's mobility [8,14,15]. We enhance the mobility detection process of these studies by using a hysteresis margin.

**The single threshold approach.** In the above mentioned studies, the well-known Tanimoto coefficient is used to compare the similarity of consecutive scans:

$$T(\vec{f_1}, \vec{f_2}) = \frac{\vec{f_1} \cdot \vec{f_2}}{\|\vec{f_1}\|^2 + \|\vec{f_2}\|^2 - \vec{f_1} \cdot \vec{f_2}}$$

where $\vec{f_1}, \vec{f_2}$ are vectors capturing the signal strength of access points in two consecutive scans. If an access point is found in only one scan, the corresponding entry in the other vector is set to zero. A value of one indicates perfect similarity, zero represents total dissimilarity. Figure 8 shows the Tanimoto coefficient for a user that is alternately walking (low similarity) and stationary (high-similarity). In [8], an empirically derived threshold of $\gamma$=0.7 is proposed to distinguish the two states.

When analyzing our data set we found that the single threshold approach leads to significant overlap between the active and passive states, resulting in misclassification and missed opportunities for energy savings. For example, in Figure 8, a threshold of 0.7 would lead to several inactive periods being marked as active.

**Our hysteresis margin approach.** To improve the stability and accuracy of the existing method, we use a hysteresis margin with upper and lower thresholds $0 \leq \beta \leq 1$ and $0 \leq \gamma \leq 1$, respectively. We determine these thresholds empirically using a two step process. First, we derive a Pareto front, and then identify the best thresholds for our point of interest in the Pareto front.

Figure 7 shows the ROC-diagram of our classifier, obtained from an exhaustive search of the parameter space. We specify two performance metrics: the *true positive rate* (TPR), the ratio of samples correctly identified as passive, out of all samples labeled passive; and the *false positive rate* (FPR), the ratio of samples incorrectly labeled as passive, out of all samples labeled active. These metrics represent a tradeoff between information and energy. Minimizing the FPR minimizes information loss because a scan will not be terminated early if a user is active (moving). Maximizing the TPR minimizes energy consumption because we can exit the scan earlier.

TABLE II. CLASSIFIER PERFORMANCE NUMBERS. THE 'TRAINED FPR' COLUMN INDICATES THE MAXIMUM FPR FOR WHICH THE CLASSIFIER WAS TRAINED. THE OTHER COLUMNS SHOW THE ACTUAL RATES ACHIEVED ON THE TEST DATA.

| Trained FPR | $\beta$ | $\gamma$ | Achieved TPR | Achieved FPR |
|---|---|---|---|---|
| 0.03 | 0.910 | 0.859 | 0.5943 | 0.0373 |
| 0.04 | 0.847 | 0.805 | 0.7521 | 0.0458 |
| **0.05** | **0.825** | **0.619** | **0.8130** | **0.0552** |
| 0.06 | 0.788 | 0.563 | 0.8625 | 0.0640 |
| 0.07 | 0.758 | 0.533 | 0.8867 | 0.0770 |
| 0.08 | 0.727 | 0.518 | 0.8995 | 0.0839 |
| 0.09 | 0.718 | 0.475 | 0.9062 | 0.0908 |
| 0.10 | 0.705 | 0.444 | 0.9165 | 0.1082 |

Selecting an appropriate operating point depends on how much information loss the application allows. For example, if an information loss of 5 % is acceptable, the optimal parameter set is the one that maximizes TPR while achieving an FPR of $\leq 0.05$. In this paper we assume that the application tolerates a small amount of information loss in exchange for energy savings. Table II shows the results for various points in the Pareto front. We trained $\beta$ and $\gamma$ by exhaustively searching the parameter space (Trained TPR), and then we used ten-fold cross validation to obtain the "Achieved" TPR and FPR. Training for an FPR of 5 % yields a good balance: with only 5.52 % information loss the classifier is able to save energy by breaking early 81.3 % of the time. The parameter set that achieves these results is $\beta = 0.825$ and $\gamma = 0.619$.

It is important to note that the average of $\beta$ and $\gamma$ (0.72) is close to the empirical threshold of 0.7 found in an independent study performed on different cities [8]. We conjecture that this occurs because the probabilistic distribution of access points follows a similar pattern in areas with comparable penetration of wireless internet access points.

### E. Algorithm

Algorithm 1 merges the ideas from the previous sections in a simple scanning algorithm. The algorithm iterates over all channels in the $channelSequence$ list, a global variable that is initialized to include all available channels. At the end of the scan, the channel sequence is updated such that the most populated channel appears at the front of the list (line 12, *channel popularity property*). This ensures that the most populated channels are scanned first in the next round, thus helping the algorithm to terminate earlier. The $result$ variable maintains the set of all APs discovered during the current scan, including details such as MAC address, SSID, and signal level. The $scan$ method scans a single channel and returns a set of discovered APs. The algorithm breaks off scanning when $\alpha$ access points have been discovered (line 5: *diminishing returns property*) or the scan result is sufficiently similar to the previous one (line 8: *scan similarity property*).

The $similar$ method implements our hysteresis-based classifier. Note that when computing the Tanimoto coefficient between two scan results only the channels included in both scans are taken into account. For example, when comparing a full scan result to a single channel scan result, only the APs found on the channel that was scanned in the latter scan are used. Otherwise the large number of missing APs would

**Algorithm 1** Incremental Scanning Algorithm.

```
1: function INCREMENTALSCAN
2:     result ← {}
3:     for each channel ∈ channelSequence do
4:         result ← result ∪ scan(channel)
                                  ▷ Break if α APs have been found
5:         if |result| ≥ α then
6:             break
7:         end if
                              ▷ Break if the user is in the same location
8:         if similar(result, previous) then
9:             break
10:        end if
11:    end for
                                              ▷ Prefer popular channels
12:    channelSequence ← sortChannels(result)
13:    previous ← result
14:    return result
15: end function
```

drive the Tanimoto coefficient down and cause the algorithm to misclassify the user state as active.

## IV. Evaluation

Our evaluation is divided into two parts: the effectiveness of our algorithm in breaking off early from full scans, and the amount of energy saved on different platforms.

**Insight 1:** *most of the savings of incremental scanning are obtained in the passive state, where less than two channels need to be scanned.* Incremental scanning saves energy in areas of high AP density and during periods of user inactivity. This means that the performance of our system depends on the environment and daily rhythm of the user. To evaluate the real-world performance of incremental scanning we collected a second dataset from five users who live in Delft, a middle-sized city in The Netherlands. They volunteered to run a scanning application that took a full access point scan every 30 seconds for a full month as they went about their normal lives.

Figure 9 shows the complementary CDF of access-point density for each user. There are strong differences between users. For example, User 2 lives in the city center surrounded by many small apartments, whereas User 1 lives in the spacious suburbs. Therefore User 2's smartphone will have many more opportunities to break off a scan early, compared to User 1.

We applied Algorithm 1 to the user data off-line with $\alpha = 15$, and with $\beta = 0.825$ and $\gamma = 0.619$ (c.f. Table II). Table III shows that between 80 % and 90 % of the time the users are passive, and in this state fewer than two channel scans are required on average. When a user is active, an early break off is obtained between 1.89 % and 9 % of the time with 4-6 channel scans.

**Insight 2:** *depending on the platform, incremental scanning could save between 20.64 % and 57.79 % of energy used for Wi-Fi scanning.* To quantify the energy savings on all the <user, platform> tuples (40 combinations), we fed the scan results of the five users (Table III) to the energy models derived in Section II-E. Figure 10 shows the results. On devices that support the 2.4 GHz spectrum only, energy savings are roughly between 20-35 % depending on the device and user. On more modern devices, such as the Samsung Galaxy S2 and HTC One X+, savings are between 40-55 % largely because a full scan encompasses many more channels and thus consumes much more energy.

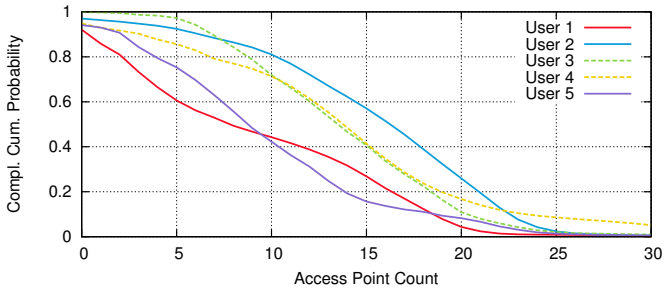


Fig. 9. Complementary cumulative probability (tail distribution) of access point density found in our user study data.

TABLE III. Early-breaking opportunities during our one-month user study.

| | Full Scan | $\|scan\| \geq \alpha$ | | Passive | |
|---|---|---|---|---|---|
| | Percentage | Percentage | Channels | Percentage | Channels |
| User 1 | 18.45 % | 2.18 % | 5.67 | 79.37 % | 1.18 |
| User 2 | 6.78 % | 9.00 % | 4.65 | 84.22 % | 1.44 |
| User 3 | 6.13 % | 4.68 % | 4.40 | 89.19 % | 1.70 |
| User 4 | 12.19 % | 7.97 % | 4.71 | 79.84 % | 1.33 |
| User 5 | 14.33 % | 1.89 % | 5.52 | 83.78 % | 1.46 |

## V. Related Work

A number of techniques have been proposed to reduce the cost of location sensing in smartphones, many of which are applicable to Wi-Fi scanning. For example, in many multi-sensor localization systems, motion sensors are used to detect when a user is not moving, so as to avoid energy-costly operations like Wi-Fi scanning or GPS [6]–[10,16]. This technique is known as *accelerometer suppression* or *sensor replacement*. Some of these systems, notably EEMSS [7], SensLoc [8], and WiFiSense [16] achieve further energy savings by duty cycling the accelerometer. However, these systems have been evaluated on only two different devices, the Nokia N95 introduced in 2007 ([6,7,10]) and the HTC G1 released in 2008 ([8,9,16]). We have shown in Section II, that the cost of accelerometer sampling on more modern devices from 2010 and onward is much higher, mainly due to an increased CPU overhead. As a result, accelerometer suppression is not competitive compared to scanning only a few channels during user inactivity. This was verified by [11], which evaluated the accelerometer suppression technique from SensLoc on an HTC Desire (2010) and found an energy consumption increase of 6 % over the 'always scan' baseline.

Another approach to reduce the energy consumption of localization systems is to exploit the relatively low entropy of human mobility [17], and try to *predict* when a user is about to move. For example, EnLoc [18] learns user behavior patterns and schedules location sensing in such a way as to maximize the localization accuracy for a given energy budget. SmartDC [14] models the problem of scheduling sensing moments as a Markov decision process. We consider this class of work to be complementary to ours. When the mobility predictor indicates an active state, our algorithm can provide Wi-Fi location information at low cost. In case of falsely predicting mobility when the user is actually idle, our algorithm could detect the stationarity condition at a very low energy cost and provide this information to the predictor engine to enhance its accuracy.

Partial scanning has been used to reduce the scanning delay during hand-off in mobile Wi-Fi [19]–[21]. Kim et al. [19]

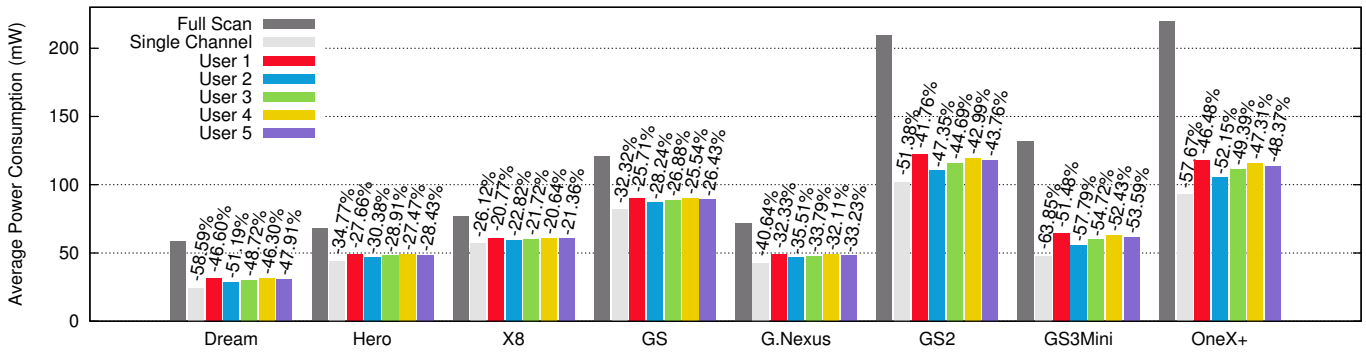


Fig. 10. Projected power consumption of incremental scanning computed from real-world user study data.

use a neighbor graph to construct a set of possible next APs and optimizes the scanning procedure and selectively scans only the channels where these are located. A channel mask is used in [20] to scan only those channels that contained APs on the previous scan, plus the popular channels 1,6, and 11. Cabernet [21] proposes a custom scanning algorithm designed to quickly locate open access points for use in vehicular wireless networks. The system scans continuously, visiting popular channels more often than less popular ones to improve the chance of discovering an open AP. We also leverage the lower energy cost of employing selective scanning but for Wi-Fi localization instead of hand-offs, which considers some fundamentally different properties such as the diminishing returns effect.

## VI. Conclusions

In this paper we presented *incremental scanning*, a novel approach for reducing the energy cost of Wi-Fi based localization. We use three techniques to reduce the number of scanned channels. First, we have shown that Wi-Fi localization services are subject to diminishing returns, and that discovering 15 access points is sufficient for an accurate fix. Second, the majority of APs reside on a small number of 'popular' channels, and scanning these channels first increases the chances of finding the fifteen APs early. Third, human beings are typically stationary 89 % of the time, in which case scanning only a small number of channels (typically 1-2) is enough to determine that the user has not moved. We constructed an energy model for eight devices, and computed projected power consumption based on traces from five users collected over a one-month period. This analysis shows that energy savings depend on both device and user behavior, and ranges between 20.64 % and 52.19 % on b/g/n devices, and between 41.76 % and 57.79 % on modern a/g/b/n devices.

## References


[1] "Google geolocation API," http://code.google.com/apis/gears/api_geolocation.html.

[2] "Skyhook wireless," http://www.skyhookwireless.com/.

[3] N. Klepeis, W. Nelson, W. Ott, J. Robinson, A. Tsang, P. Switzer, J. Behar, S. Hern, W. Engelmann *et al.*, "The national human activity pattern survey (nhaps): a resource for assessing exposure to environmental pollutants," *Journal of exposure analysis and environmental epidemiology*, vol. 11, no. 3, pp. 231–252, 2001.

[4] "Modifications to wpa_supplicant to support scanning only a subset of the available wi-fi channels on android smart phones." http://code.google.com/p/android-partial-scanning/.

[5] "Monsoon power monitor," http://www.msoon.com/LabEquipment/PowerMonitor/.

[6] M. B. Kjærgaard, J. Langdal, T. Godsk, and T. Toftkjær, "EnTracked: energy-efficient robust position tracking for mobile devices," ser. MobiSys '09, 2009, pp. 221–234.

[7] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh, "A framework of energy efficient mobile sensing for automatic user state recognition," ser. MobiSys '09, pp. 179–192.

[8] D. H. Kim, Y. Kim, D. Estrin, and M. B. Srivastava, "SensLoc: sensing everyday places and paths using less energy," in *8th ACM Conf. on Embedded Networked Sensor Systems*, Nov. 2010, pp. 43–56.

[9] Z. Zhuang, K.-H. Kim, and J. P. Singh, "Improving energy efficiency of location sensing on smartphones," ser. MobiSys '10, pp. 315–330.

[10] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell, "The jigsaw continuous sensing engine for mobile phone applications," ser. SenSys '10, 2010, pp. 71–84.

[11] H. Blunck, M. B. Kjærgaard, and C. Melchior, "Poster: evaluating energy consumption of sensing algorithms - solely via models?" ser. MobiSys '12.

[12] N. Bulusu, J. Heidemann, and D. Estrin, "Adaptive beacon placement," in *21st Int. Conf. on Distributed Computing Systems*, 2001, pp. 489–498.

[13] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks," ser. MobiCom '03, 2003, pp. 81–95.

[14] Y. Chon, E. Talipov, H. Shin, and H. Cha, "Mobility prediction-based smartphone energy optimization for everyday location monitoring," ser. SenSys '11, 2011, pp. 82–95.

[15] Y. Chon, N. D. Lane, F. Li, H. Cha, and F. Zhao, "Automatically characterizing places with opportunistic crowdsensing using smartphones," in *ACM Conf. on Ubiquitous Computing*, 2012, pp. 481–490.

[16] K. Kim, A. Min, D. Gupta, P. Mohapatra, and J. Singh, "Improving energy efficiency of Wi-Fi sensing on smartphones," in *INFOCOM*, 2011, pp. 2930–2938.

[17] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[18] I. Constandache, S. Gaonkar, M. Sayler, R. Choudhury, and L. Cox, "Enloc: Energy-efficient localization for mobile phones," in *IEEE INFOCOM*, 2009, pp. 2716–2720.

[19] S.-H. Park, H.-S. Kim, C.-S. Park, J.-W. Kim, and S.-J. Ko, "Selective channel scanning for fast handoff in wireless LAN using neighbor graph," in *Personal Wireless Communications*, 2004, pp. 194–203.

[20] S. Shin, A. G. Forte, A. S. Rawat, and H. Schulzrinne, "Reducing MAC layer handoff latency in IEEE 802.11 wireless LANs," ser. MobiWac '04, 2004, pp. 19–26.

[21] J. Eriksson, H. Balakrishnan, and S. Madden, "Cabernet: vehicular content delivery using WiFi," in *14th ACM int. conf. on Mobile computing and networking*, 2008, pp. 199–210.