Improving Energy Efficiency of Wi-Fi Sensing on Smartphones

Kyu-Han Kim,[1] Alexander W. Min,[2] Dhruv Gupta,[3] Prasant Mohapatra,[3] Jatinder Pal Singh[1]

[1]Deutsche Telekom R&D Laboratories USA, Los Altos, CA 94022

[2]Department of EECS, The University of Michigan, Ann Arbor, MI 48109

[3]Department of Computer Science, University of California at Davis, Davis, CA 95616

{kyu-han.kim, jatinder.singh}@telekom.com, alexmin@eecs.umich.edu, {dhgupta, pmohapatra}@ucdavis.edu

***Abstract*—Mobile data usage over cellular networks has been dramatically increasing over the past years. Wi-Fi based wireless networks offer a high-bandwidth alternative for offloading such data traffic. However, intermittent connectivity, and battery power drain in mobile devices, inhibits always-on connectivity even in areas with good Wi-Fi coverage. This paper presents *WiFisense*, a system that employs user mobility information retrieved from low-power sensors (e.g., accelerometer) in smartphones, and further includes adaptive Wi-Fi sensing algorithms, to conserve battery power while improving Wi-Fi usage. We implement the proposed system in Android-based smartphones and evaluate the implementation in both indoor and outdoor Wi-Fi networks. Our evaluation results show that WiFisense saves energy consumption for scans by up to 79 % and achieves considerable increase in Wi-Fi usage for various scenarios.**

I. Introduction

With the increasing popularity of Internet enabled mobile devices, and a plethora of mobile web applications [1], [2] in use, the data consumption on smartphones has sky-rocketed over the past years. Several nation-wide wireless carriers have reported a staggering increase in mobile data traffic since their release of smartphones [3], [4].

While Wi-Fi networks present a high bandwidth, and often cheaper, alternative to mobile Internet users for offloading this data traffic, and a good option for service providers [5], there still exist several challenges. For satisfactory user experience, mobile Internet applications such as delay sensitive Voice-over-IP warrant seamless connectivity in areas with wireless coverage. However, this is taxing on the mobile phone battery owing to re-connections needed because of user mobility, and short transmission range of Wi-Fi networks. Although turning on Wi-Fi only on-demand, as implemented in most smartphones, might save energy, it dramatically reduces the usage of available Wi-Fi access opportunities. To maximize Wi-Fi usage, it is imperative to (i) determine an optimal Wi-Fi sensing frequency for energy-efficient Wi-Fi discovery and (ii) design a practical system that achieves both increased Wi-Fi usage and reduced energy consumption.

In this paper, we derive an optimal Wi-Fi sensing interval via an in-depth analysis, and identify several important factors to achieve energy-efficient Wi-Fi sensing for mobile devices. For the analysis, we model the availability of Wi-Fi connectivity as a two-state Markov model, and given the model, we derive an optimal sensing interval that minimizes missed Wi-Fi access opportunities, while conserving energy. Our analysis reveals that the optimal sensing interval depends on several network parameters, including user's average velocity and Wi-Fi access point (AP) density.

Based on the observations from the analysis, we design a mobile-centric Wi-Fi sensing system, called *WiFisense*, which maximizes the usage of open Wi-Fi access opportunities via the following salient features. First, WiFisense exploits user's movement information, extracted from low-power sensors such as an accelerometer, in order to determine when to sense. Such sensor-based mobility detection avoids triggering expensive location tracking or war-driving. Second, WiFisense includes *disconnected* sensing algorithms that determine Wi-Fi scanning frequency adaptively based on user mobility and AP density information, when a smartphone is not connected to any Wi-Fi network. Third, WiFisense is also equipped with *connected* sensing algorithms that pro-actively identify better APs by adapting scan-triggering threshold, while minimizing false triggering.

We implement WiFisense on Android Development Phones (ADPs) [6] and evaluate the implementation on multiple Wi-Fi networks. Our implementation has been evaluated on both an indoor office Wi-Fi network, and an outdoor Wi-Fi testbed consisting of 34 nodes. In addition, we have also used real-life traces from several users' smartphones, to validate the effectiveness of WiFisense. Our evaluation results show that WiFisense significantly improves energy efficiency for always-on Wi-Fi connectivity. The sensor-assisted sensing algorithms reduce the number of scans by up to 79 % for various scenarios. Further, the adaptive scan-triggering threshold algorithm helps determine optimal sensing time, while maintaining false-triggering as low as 4.3 %.

This paper makes the following main contributions.

- We numerically analyze an optimal sensing interval for a smartphone in general open Wi-Fi networks and characterize key factors for energy-efficient Wi-Fi sensing.
- We present the WiFisense system that takes into account the factors identified in the analysis and provides mobility-aware Wi-Fi sensing capability for smartphones.
- We implement WiFisense on Android-based smartphones, and evaluate its performance extensively on both indoor and outdoor Wi-Fi testbeds.
- Via real-life experiments on our testbeds, we demonstrate that WiFisense reduces sensing frequency by up to 79 %, while keeping a false-triggering rate low.

The remainder of the paper is organized as follows. Section II discusses related work. Section III studies an optimal sensing interval via analysis. Section IV presents the design and algorithms of WiFisense. Section V describes a system prototype of WiFisense and presents the evaluation results. Finally, Section VI concludes this paper.

II. Related Work

In [7], authors use received signal strength changes from cellular towers as a trigger for Wi-Fi scanning. However, cellular signal strength varies considerably with changing locations and heterogeneous network environments (e.g., cell-tower density). In [8], authors use a Bluetooth fingerprint to detect available APs. However, this requires a training phase to create and maintain the fingerprint database, which could be expensive given the fact that Bluetooth devices are usually highly mobile. In [9], the authors propose to use a ZigBee radio to detect Wi-Fi signals in ISM bands. However, this approach requires the use of a second radio, and thus might not be feasible for most smartphones.

In [10], authors present a scheme to pro-actively scan for available APs to improve hand-off latency. In [11], authors propose a scheme that reduces handoff latency by associating with the AP that provides the longest connectivity. In [12], the authors propose to use habitual human mobility to forecast Wi-Fi connectivity. On the other hand, our work focuses more on energy-efficient discovery of Wi-Fi access opportunities.

In [13], the authors present an energy-aware fair-scheduling algorithm that minimizes Wi-Fi radio wake-up time and eliminates unnecessary retransmissions. In [14], authors present a scheme that saves energy by combining tiny gaps between packets into meaningful sleep intervals, thus allowing mobiles clients to sleep during data transfers. While these works focus on minimizing energy spent in data transfer, our goal is to improve the energy efficiency of the Wi-Fi scanning and migration process. The work most directly related to ours is presented in [15]. The authors propose using the Wi-Fi interface opportunistically, in place of cellular network, to conserve energy during data transmissions. However, their main focus is to improve energy efficiency for data transfer based on *prior* estimated network condition information.

The main differentiating factor for our scheme is that it only requires knowledge of the user state and local network information (e.g.,AP density, average RSSI) for maximizing Wi-Fi access opportunities, and results in the increased usage of Wi-Fi networks and considerable energy savings.

III. Optimal Sensing Interval Analysis

To maximize Wi-Fi discoveries for energy-constrained mobile devices, we need to find an optimal sensing interval that balances increased Wi-Fi access opportunities and sensing-induced energy cost. In this section, we analytically derive an optimal sensing interval for a mobile device to minimize energy spent on scanning per transmitted bit within a scanning interval.[1] Based on the derivation, we characterize important design parameters, such as user's mobility and network density, to be used for WiFisense.

A. Wi-Fi Connectivity Model

In order to analyze an optimal Wi-Fi sensing interval, it is essential to estimate how often a mobile user experiences AP arrivals. To this end, we make the following two assumptions. First, we assume that the AP distribution follows a point Poisson process with the average density ρ, i.e., number of APs per m^2. A recent measurement study presented in [16] also supports our assumption of Poisson-like AP distribution.

[1]We interchangeably use "sensing" and "scanning" throughout this paper.

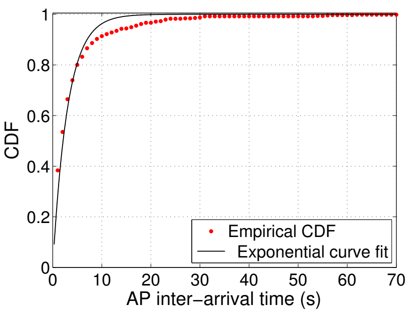


Fig. 1. AP inter-arrival time distribution: The measurement result shows the inter-arrival time can be accurately approximated as exponential distribution.

To further justify this assumption, we analyzed WLAN trace data available from Crawdad [17], which indicates that the AP inter-arrival time can be approximated as an exponential distribution (Fig. 1). Second, we assume that the client-AP connectivity lifetime, i.e., the time span during which the physical distance between the client and AP is smaller than the effective transmission range, can be approximated as an exponential distribution, i.e., $T_l \sim \exp\left(\frac{\bar{v}}{R}\right)$, where $\bar{v}$ is the average speed of the mobile client, and R is the common transmission range of the APs.[2] This can be justified based on the analytical and measurement study in [18].

Given these assumptions, we can approximate the average arrival rate of new APs as $\lambda \approx 2R\bar{v}\rho$ where ρ is the average AP density. Further, we can model the availability of Wi-Fi networks as a Markov model with two states, i.e., *disconnected* (D) and *connected* (C) (Fig. 2). The *disconnected* state implies that a mobile device has no available APs, and *connected* refers to the device being associated. Now, the AP arrival and departure rates are given as $\lambda = 2R\bar{v}\rho$ and $\mu = \frac{\bar{v}}{R}$, respectively. Thus, the state transition probabilities for the two state model can be derived as $\lambda_d = \lambda$ and $\lambda_c = \frac{(\mu-\lambda)\lambda}{\mu}$ where λ_d and λ_c are the state transition probabilities for D$\rightarrow$C and C$\rightarrow$D, respectively.

B. Optimal Scanning Interval

Based on the above connectivity model, we derive an optimal scanning interval (T_s) that saves energy while minimizing the missed Wi-Fi access opportunities. We define *missed opportunity* as the average fraction of time that a Wi-Fi access opportunity has not been detected by a mobile client.

Proposition 1: (Missed Wi-Fi Opportunity) Consider a smartphone that is disconnected from Wi-Fi networks and periodically scans the spectrum at a fixed interval T_s. Let $T_{m,d}(T_s)$ denote the missed connectivity time (until it finds an available AP) using the fixed scan interval T_s. Then, we have:

$$T_{m,d}(T_s) = \frac{\lambda_d}{\lambda_d + \lambda_c}\left(1 - e^{-(\lambda_d+\lambda_c)T_s}\right), \quad (1)$$

where $\lambda_d = 2R\bar{v}\rho$ and $\lambda_c = 2R\bar{v}\rho\,(1 - 2R^2\rho)$.

Proof: Let $T_{m,d}(t)$ denote the average missed AP connectivity (in terms of time) during time period $(\tau, \tau + t)$. When there was no available AP at time τ, the opportunity for connectivity (i.e., AP association) occurs only when an available AP arrives during the time period t. Thus, the

[2]We assume a unit-disk model for the range for analytical tractability.

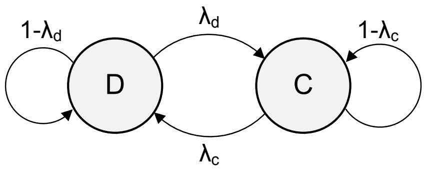


Fig. 2. Markov model for Wi-Fi availability.

expected amount of missed opportunity for the duration of missed connectivity in $(\tau, \tau+t)$ can be expressed as:

$$T_{m,d}(t) = \int_0^t \frac{\mathbb{F}_d(x)}{E[T_d]}\, T_{m,\overline{c}}(t-x)\,\mathrm{d}x, \tag{2}$$

where $\mathbb{F}_d(t)$ is the survivor function defined as $\mathbb{F}_d(t) = \int_x^\infty f_d(u)\mathrm{d}u$. $T_{m,\overline{c}}(t-x)$ is the expected missed connectivity in the time duration $(\tau, \tau+t-x)$ when D→C transition occurs *exactly* at time τ (the overline is used to differentiate this case), which can be expressed as:

$$T_{m,\overline{c}}(t) = t\int_t^\infty f_c(x)\,\mathrm{d}x + \int_0^t f_c(x)\big(x + T_{m,\overline{d}}(t-x)\big)\,\mathrm{d}x. \tag{3}$$

Similarly, we can derive:

$$T_{m,\overline{d}}(t) = \int_0^t f_d(x)\, T_{m,\overline{c}}(t-x)\,\mathrm{d}x. \tag{4}$$

While we can calculate the expected amount of missed opportunity, $T_{m,d}$, based on Eqs. (2), (3), and (4), such a derivation in time domain is complicated. Thus, we can solve for $T_{m,d}$ in the frequency domain by taking the Laplace transform, i.e.,

$$T_{m,d}(s) = \frac{\mathbb{F}^*_d(s)\, T^*_{m,c}(s)}{E[T_c]}, \tag{5}$$

$$T_{m,\overline{c}}(s) = \frac{f^*_c(0) - f^*_c(s)}{s^2} + f^*_c(s)\, T^*_{m,\overline{d}}(s), \tag{6}$$

and

$$T_{m,\overline{d}} = f^*_{m,d}(s)\, T^*_{m,\overline{d}}(s). \tag{7}$$

Based on Eqs. (5), (6), and (7), the expected missed opportunity $T_{m,d}(s)$ is given as:

$$T_{m,d}(s) = \frac{\mathbb{F}_d(s)}{E[T_d]\, s^2} \cdot \frac{f^*_c(0) - f^*_c(s)}{1 - f^*_c(s) f^*_d(s)}. \tag{8}$$

For exponentially-distributed connected and disconnected periods, i.e., $f_d(t) = \lambda_d e^{-\lambda_d t}$ and $f_c(t) = \lambda_c e^{-\lambda_c t}$, the Laplace transforms are given as $f^*_d(s) = \frac{\lambda_d}{\lambda_d + s}$ and $f^*_c(s) = \frac{\lambda_c}{\lambda_c + s}$. By substituting these in Eq. (8) and taking the inverse Laplace transform, we get:

$$T_{m,d}(t) = \frac{\lambda_d}{\lambda_d + \lambda_c}\Big(1 - e^{-(\lambda_d+\lambda_c)t}\Big). \tag{9}$$

Thus, Eq. (1) follows. ■

Proposition 1 indicates that the missed opportunity increases with increasing scanning interval T_s. From Eq. (1), the missed Wi-Fi opportunity $T_{m,d}(T_s) \to \frac{E[T_c]}{E[T_c]+E[T_d]}$ as $T_s \to \infty$. This confirms our intuition that as the scanning interval goes to infinity, the user will miss-detect all the Wi-Fi opportunities.[3]

Based on above analysis, we now derive an optimal scanning interval vis-a-vis energy overhead incurred. Specifically, we want to minimize energy spent on scanning per transmitted bit within a scanning interval T_s. Note that here we consider the disconnected scenario, and thus assume that the energy consumption for data transmission is negligible. When a mobile user scans for APs with a fixed interval T_s, the fraction of the time that he will detect Wi-Fi opportunity is $1 - T_d - T_{m,d}(T_s)$, where $T_d = \frac{\lambda_c}{\lambda_d+\lambda_c}$ is the average fraction of time in disconnected state. Then, we can find the optimal scanning interval as follows:

$$T^*_s = \arg\min_{0<T_s\le T_{max}} \frac{\mathcal{E}_{scan}}{T_s\,(1 - T_d - T_{m,d}(T_s))\,E[\mathcal{R}]}, \tag{10}$$

where T_{max} is the maximum scanning interval, which is a design parameter, and $\mathcal{E}_{scan}$ is the energy consumption for scanning. For example, T_{max} can be set to a small value for delay-sensitive applications (e.g., VoIP). In Eq. (10), the expected achievable data rate is denoted as $E[\mathcal{R}]$, and it depends on the AP with which a client associates.

Proposition 2: (Optimal Scanning Interval) The optimal scanning interval that maximizes energy efficiency (in Eq. (10)) is:

$$T^*_s(\bar{v}, \rho) = \left[\frac{1}{4R\bar{v}\rho\,(1 - R^2\rho)}\right]^+, \tag{11}$$

where $\bar{v}$ is the average speed, ρ is the average AP density, and R is the AP's transmission range (e.g., 200 m).

Proof: Let us denote $\frac{\mathcal{E}_{scan}}{T_s\,(1-T_d-T_{m,d}(T_s))\,E[\mathcal{R}]}$ by $\mathcal{G}(T_s)$, which represents our objective function in Eq. (10), i.e., energy spent on scanning per transmitted data bit. For given energy consumption for scanning, $\mathcal{E}_{scan}$, and average achievable data rate, $E[\mathcal{R}]$, the optimal scanning interval T^*_s in Eq. (10) can be defined by evaluating T_s that maximizes $\mathcal{G}(T_s)$ as:

$$\frac{\partial\,\mathcal{G}(T_s)}{\partial\,T_s} = \frac{\mathcal{E}_{scan}}{T_s\,(1 - T_d - T_{m,d}(T_s))\,E[\mathcal{R}]} = 0$$
$$\Longrightarrow \lambda_d\left(\frac{1}{\lambda_d+\lambda_c} - T_s\right)e^{-(\lambda_d+\lambda_c)T_s} = 0. \tag{12}$$

Obviously, $\mathcal{G}(T_s)$ is monotonically increasing function of T_s in $(0, \frac{1}{\lambda_d+\lambda_c}]$. Thus, the optimal value of T_s is given as:

$$T^*_s = \frac{1}{\lambda_d+\lambda_c}. \tag{13}$$

Thus, Eq. (11) follows. ■

Proposition 2 implies that the optimal scanning period in Eq. (11) depends on the average moving speed ($\bar{v}$) of a user and average AP density (ρ). For example, when a user moves faster, the AP arrival rate (λ_d) increases, and thus the user is encouraged to scan more frequently, expecting to detect an available AP(s) soon, and vice versa.

C. Observations

From the above analysis, we hereunder identify key factors that affect the optimal sensing interval:

- *Movement of mobile users*: The velocity of mobile users is an important factor in determining the sensing interval. However, in practice the velocity changes over time, and this variation has to be monitored and incorporated into sensing algorithms.
- *Network information*: Network information (e.g., AP density) is critical in determining the sensing frequency. Although we use a certain distribution for network density

[3]Note that in the analysis, we do not consider the relatively negligible time needed for scanning, while accounting for access opportunity time.

in our analysis, real network information needs to be obtained and used by a mobile device at a given location.
- *Heterogeneous Wi-Fi Networks*: Each network provides different network performance (e.g., $E[\mathcal{R}]$), and Wi-Fi sensing needs to be aware of such heterogeneity to better identify usable Wi-Fi networks at a given location.

In what follows, we will design a practical Wi-Fi sensing system based on the key observations above.

IV. The WiFisense Design

We first provide an overview of the WiFisense design and then explain the details of the disconnected and connected sensing components.

A. Overview

WiFisense is a Wi-Fi sensing system for smartphones that significantly improves sensing efficiency and the usage of open Wi-Fi networks via the following salient features.

- *Leveraging user-mobility information*: WiFisense exploits user's mobility information obtained from motion sensors in smartphones. Mobility information is used to approximate distance that a user traveled.
- *Adaptive use of network information*: WiFisense opportunistically profiles network information (e.g., AP density and average received signal strength (RSS)) and adaptively determines sensing frequency (i.e., when to sense).
- *Proactive sensing*: WiFisense includes both disconnected and connected sensing algorithms, which proactively scan Wi-Fi access points, to improve users' Wi-Fi usage in an energy-efficient manner.

Algorithm 1 shows the overall operation of WiFisense, which can be explained for the following three recurring periods: (1) *Movement monitoring period*, during which a smartphone periodically senses user's movement using an accelerometer at a given duty cycle[4]; (2) *Disconnected period*, during which a smartphone is disconnected from Wi-Fi networks, and it periodically scans ISM bands based on user movement and AP density information to discover available Wi-Fi networks; and (3) *Connected period*, during which a smartphone is connected to Wi-Fi networks, and it proactively scans other usable APs[5] based on movement and average RSS information to find a better AP.

In the following subsections, we will elaborate on sensing algorithms for both disconnected and connected scenarios.

B. Disconnected Sensing

Disconnected sensing (DS) is essentially a scanning operation performed when smartphones are not connected to Wi-Fi networks. Here, 'disconnected' states include (i) smartphone using alternative access networks (e.g., 3G) and (ii) smartphone not using any access network but being on stand-by. As an example of case (i), the Wi-Fi manager in iPhone 4 periodically scans ISM bands and pops up a message asking whether a user wants to connect to available Wi-Fi networks.

DS in WiFisense opportunistically triggers Wi-Fi scanning based on user's mobility information. Intuitively, physical

Algorithm 1 Wi-Fi Sensing Operation

```
(1) Movement monitoring period
 1: wake up the sensor for $d$ seconds;
 2: read the sensor data;
 3: classify user's movement activity;   /* e.g., stand, walk, run */
 4: calculate moving distance;
(2) Disconnected period
 5: $\widehat{d}\leftarrow$ estimated travel distance since the last sensing;
 6: if $\widehat{d} > d_{th}$ then   /* $d_{th}$ is an adaptive distance threshold */
 7:     trigger Wi-Fi sensing;
 8:     update AP density information ($n_{ap}$); update $d_{th}$;
 9:     if there exists a usable AP then
10:         go to the connected sensing (3);
11:     end if
12: end if
(3) Connected period
13: $\widehat{d}\leftarrow$ estimated travel distance since the last sensing;
14: /* update scan triggering probability $P_{scan}$ */
15: $P_{scan} \leftarrow \beta + \min\{\lfloor \frac{d}{d_{unit}} \rfloor \times \Delta, 1-\beta\}$;   /* $\beta \in [0,1]$ */
16: if RSS $< r_t$ and rand[0,1] $< P_{scan}$ then
17:     trigger Wi-Fi sensing; perform migration, if necessary;
18:     /* update adaptive RSS threshold $r_t$ */
19:     $r_t \leftarrow$ average RSS of top $k$ APs;
20: end if
21: if no usable AP available then
22:     go to the connected sensing (2);
23: end if
```

movement can increase opportunity of finding new APs, compared to stationary scenarios. Table I shows the average number of new APs discovered via scanning, while a mobile user moves (e.g., standing, walking, and running) along the road side in Silicon Valley, California for ten minutes. As shown in the table, the increase in user mobility can be translated to a higher number of discovered Wi-Fi access opportunities. However, to incorporate the mobility into WiFisense, there are several challenges to be solved:

- *Movement/distance estimation*: DS must be able to estimate movement or distance information in an energy-efficient manner. Numerous theoretical or trace-based mobility models have been proposed in literature [19], but it is difficult to apply them for personalized mobility patterns for each smartphone user in real time. The tracking of mobile users (e.g., Global Positioning System or GPS) is known to be power-intensive [20]. In contrast, WiFisense adopts activity-based mobility estimation used in [21]–[23] to infer user's *meaningful* activities, including standing, walking, and running, based on accelerometer readings with a high accuracy (see Fig. 3). Although such estimation may not be accurate enough to track fine-grained trajectory (like GPS) of users, it is sufficient for estimating approximate movement of a mobile user at a reasonable energy cost (65 mW of an accelerometer vs. 600 mW of GPS).

TABLE I
Average number of APs detected per minute

Velocity (m/s)	Mean	Standard deviation
0-1 (standing)	0.9	1.595
2-5 (walking)	10.4	7.471
6-8 (running)	28.7	16.563

[4]We use a 20 % duty cycle (e.g., 1 sec per 5 sec) in our experiments.

[5]A 'usable' AP refers to one that provides RSS above a certain threshold, e.g., -85 dBm.

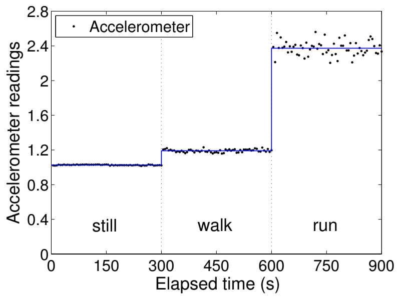


Fig. 3. Accuracy of activity detection: The data obtained from the accelerometer on G1 phone with 20 % duty cycle under different activities.

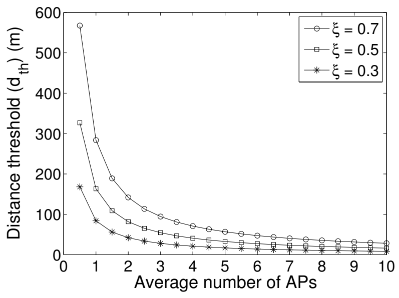


Fig. 4. Distance threshold d_{th} for AP scanning decreases as the average AP density as well as detection probability increase.

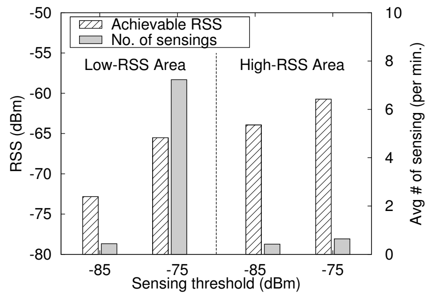


Fig. 5. Tradeoff between achievable RSS and migration overhead for different threshold values at different AP densities.

- *Monitoring network parameters*: DS also needs to be aware of network information, such as node density and RSS, around mobile users to avoid unnecessary sensing (e.g., in areas with sparse Wi-Fi networks). The exponential distribution of AP density, as used in our analysis, might be accurate over long-time periods, but in practice, it might not be suitable as mobile users typically move around in selective areas. In [24], authors assume that such network information can be provided by cellular operators. Several collaborative efforts from Wi-Fi communities (e.g., [25], [26]) help collect such information, but its coverage is often incomplete. In contrast, DS opportunistically measures local AP density for an individual mobile user, whenever DS triggers spectrum scanning. Also, DS records its statistics associated with cell ID information for future usage.
- *Heterogeneous network environments*: Due to heterogeneous network environments, DS in WiFisense needs to adapt its sensing parameter to improve energy efficiency. In urban (rural) areas with dense (sparse) Wi-Fi networks, DS might need to decrease (increase) its distance threshold (d_{th}) to reduce the delay (save the energy) in detecting APs. To incorporate such scenarios, WiFisense uses a probabilistic approach to adjust the threshold. A mobile device triggers AP scanning when it travels more than a distance threshold, d_{th}, which we define as the minimum distance that a user needs to move to find at least one available AP with a probability $\xi \in (0, 1)$, i.e., $1 - Poi(n_{ap} = 0) \geq \xi \Leftrightarrow e^{-2\rho R d_{th}} \geq 1 - \xi \Leftrightarrow d_{th} \geq -\frac{\pi R \ln(1-\xi)}{2E[n_{ap}]}$, where n_{ap} is the number of available APs detected by a previous scan. Here, $E[n_{ap}] \approx \pi R^2 \rho$, and $E[n_{ap}]$ can be estimated using the exponential weighted moving average (EWMA).

Fig. 6 illustrates the operation of DS. From previous scan results and/or statistics on average AP density, WiFisense (e.g., at t_0) calculates d_{th}. For instance, as shown in Fig. 4 from our analysis, if the average number of APs is 2 and ξ is set to 0.3, then d_{th} is 42 m. At the same time, DS periodically reads movement information to calculate actual distance ($\widehat{d}$) that a user has traveled since the last scanning. Once the estimated travel distance exceeds d_{th}, DS triggers Wi-Fi sensing (e.g., at t_1, t_2 in Fig. 4) and identifies usable APs.

C. Connected Sensing

Connected sensing (CS) is a proactive Wi-Fi sensing operation that helps WiFisense find better APs. Here, 'connected' is the state in which a mobile device is associated with an AP, and is in an area with at least one usable AP. Today, mobile users are simultaneously exposed to multiple open Wi-Fi networks (with or without authentication), including residential networks, enterprise/campus networks. Mobile users can leverage their presence for high bandwidth availability, energy efficiency, and cellular-network usage reduction.

To exploit such Wi-Fi access opportunities, CS proactively scans Wi-Fi networks to find better APs with higher RSS as the mobile user moves. We note that mobile devices such as smartphones need to remain connected to access networks, even if users are not actively using them, as required by several background applications such as messaging, email synchronization, and others. However, to implement CS, several challenges need to be addressed:

- *Monitoring of connected state*: CS must be able to not only monitor the conditions of the currently associated AP, but also sense other available APs'. To minimize any disruption in sensing other or newly available APs, a mobile device can trigger scanning only when the device is disconnected from the network [27]. However, as we observed in Fig. 5, such a fixed-threshold approach prevents the mobile clients from exploiting the available Wi-Fi access opportunities (high RSS area) [28], or causes excessive sensing overheads (low RSS area). In contrast, WiFisense adopts a dynamic RSS threshold adjustment scheme in which CS adjusts its RSS threshold r_t based on the average RSS for top k usable APs within a list of APs.[6] The higher the average RSS, the higher the probability that a mobile device finds better APs.
- *Coping with temporal and spatial RSS fluctuations*: CS in WiFisense has to minimize false triggering of Wi-Fi sensing. Temporal fluctuations in RSS can cause frequent false-triggering, consuming energy. Further, as mobile users move, CS needs to differentiate the temporal RSS fluctuations from the spatial ones to minimize any missed opportunity. Most existing work rely on RSS values to determine migration points (e.g., [29]). In contrast, WiFisense jointly uses RSS and movement information to trigger scanning. When a mobile device is stationary, it refrains from triggering Wi-Fi sensing, even if RSS is below r_t, in a probabilistic manner. On the other hand, when the RSS hits r_t and a user has moved a certain distance, a mobile host triggers Wi-Fi sensing to find new APs (see Algorithm 1.(3)).
- *Overhead for bandwidth estimation*: CS needs to be equipped with energy-efficient bandwidth estimation tools. Given a list of available APs from Wi-Fi scanning, CS needs to identify the 'best' AP that can provide high bandwidth

[6]Note that k is a system parameter. We set k as 3 in our experiments.

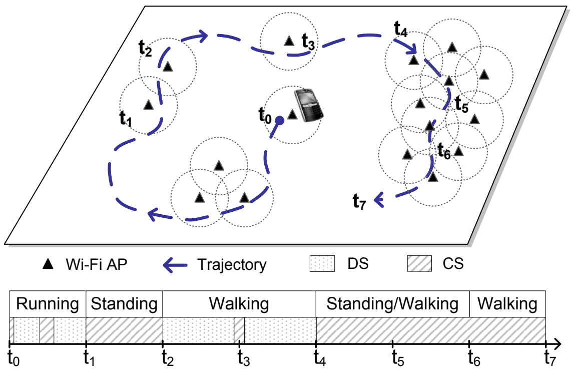


Fig. 6. Illustration of disconnected and connected states: As a mobile user moves around in different velocity, WiFisense identifies different AP densities and its movement over different areas, and thus triggers either disconnected or connected sensing algorithms, accordingly.

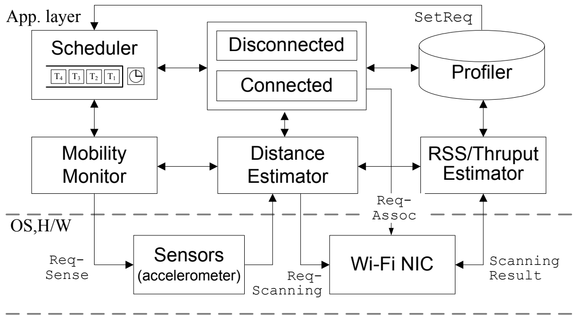


Fig. 7. Software Architecture: WiFisense is implemented in Android-based smartphones. It periodically monitors user's movement as well as Wi-Fi network condition and performs energy-efficient Wi-Fi sensing.

to determine the migration of current association. Although active probing techniques are known to be accurate for end-to-end bandwidth estimation [12], [30], such active probing methods incur additional energy and communication overheads [31]. To overcome the challenge, WiFisense takes a hierarchical approach, where it uses RSS values for the wireless last-hop (AP-to-Client) and uses existing backhaul information supported by either IEEE 802.11k [32] (for Ethernet backhaul) or a routing protocol [33] (e.g., Expected Transmission Time for mesh backhaul).

Let us consider the connected cases in Fig. 6. When a mobile device is mostly stationary (e.g., t_4, t_5, t_6), WiFisense suppresses any false triggering from temporal RSS fluctuation. Once a mobile user starts moving, it might experience RSS degradation, and if the measured RSS value hits the RSS threshold (r_t), CS further calculates movement-based probability P_{scan} to incorporate the probability that a user's recent movements have created. If both conditions (RSS and movement) are met, then CS triggers Wi-Fi scanning.

V. Implementation and Performance Evaluation

We first describe the implementation details of WiFisense and, then present the experimental setup and results.

A. Implementation Details

Fig. 7 depicts the software architecture of WiFisense implemented in Android Developer Phones (ADPs). Although WiFisense can be applied to any mobile platform, we present the architecture specifically for Android, for ease of presentation and openness of the platform.

WiFisense has been implemented as an application and consists of four major components. First, the *mobility module* (mobility monitor and distance estimator) periodically monitors user's movement information by using accelerometer readings and the Goertzel classification algorithm [34]. Second, the *sensing module* (disconnected and connected) performs periodic and on-demand Wi-Fi scanning and determines an AP to be associated with. Third, the *Wi-Fi condition monitor* (RSS/throughput estimator) is responsible for measuring and analyzing network conditions around a mobile node. Fourth, the *profiler module* maintains statistics such as AP density and average RSS of Wi-Fi networks.

B. Experimental Setup

To evaluate our implementation, we use both outdoor and indoor testbeds. Our outdoor testbed (Fig. 8) is situated in a forest reserve, and consists of 34 mesh nodes at 31 physical sites distributed over 2,000 acres of wilderness and hilly terrain in Davis, California. The testbed supports live traffic and is actively used by several researchers. The number of visible APs within the testbed ranges from 0 to 8. Detailed specification is available in [35]. Our indoor testbed is deployed inside a multi-story building in Los Altos, California, and consists of 15 APs across two floors and public APs (e.g., Google Wi-Fi). Each node in the testbed consists of Gateworks Cambria embedded devices, running OpenWRT [36].

In the above settings, we use the Davis testbed to evaluate WiFisense for *disconnected* scenarios because of its sparse coverage and hilly terrain, whereas we use the Los Altos testbed mainly for *connected* scenarios because of its high density and urban location. Also, we compare the performance of WiFisense with periodic sensing (`PS`), fixed threshold (`FT`) and `Oracle`. `PS` triggers Wi-Fi sensing periodically with a pre-defined fixed threshold. `FT` triggers Wi-Fi sensing when RSS from a current AP falls below a threshold. In `Oracle`, we assume a mobile client always detects the AP with maximum RSS, without incurring any overhead and we use `Oracle` as a performance benchmark.

C. Experimental Results

We evaluate WiFisense in its ability to improve several key aspects, including energy efficiency, detection delay, and false triggering rate, in various scenarios.

1) Energy and delay gains of disconnected sensing: We first quantify the gains in energy consumption, and AP detection delay, of the disconnected sensing algorithm on the Davis testbed, which is only sparsely covered by APs. In the experiment, we let a mobile user move around, while repeating different activities (i.e., standing, walking) for 10 minutes. We measure the energy consumption (in J) due to Wi-Fi sensing and the delay in detecting an available AP. We perform multiple runs and compare the performance of WiFisense with `PS` at fixed intervals of 10, 20, 30 and 60 seconds. Further, the distance threshold (d_{th}) for triggering AP scanning is adjusted based on the estimated AP density (as explained in Section IV-B).

Figs. 9 and 10 show that WiFisense reduces the energy consumption while keeping the detection delay reasonably low.

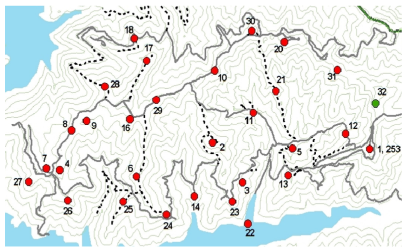


Fig. 8. Topology of the outdoor testbed (QuRiNet) at Davis. This testbed provides Wi-Fi coverage with both connected and disconnected states.

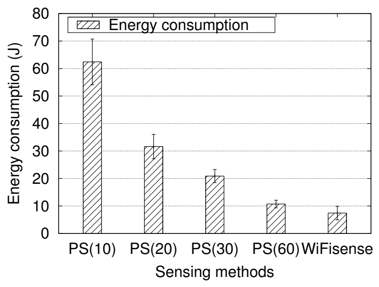


Fig. 9. Energy consumption of periodic scanning (PS) vs. WiFisense: WiFisense greatly reduces the number of AP scanning, compared to PS.

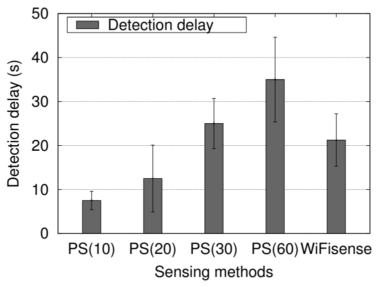


Fig. 10. Delay of periodic scanning (PS) vs. WiFisense: WiFisense keeps detection delay reasonably low, compared to PS.

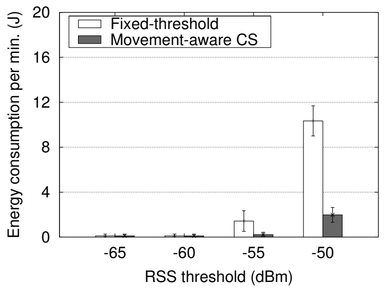


Fig. 11. Energy saving of movement-aware sensing (standing, indoor, connected).

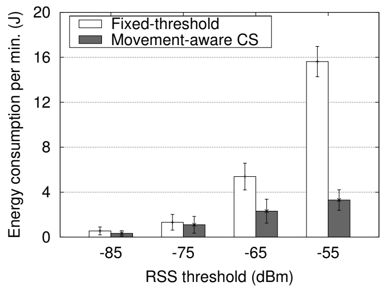


Fig. 12. Energy saving of movement-aware sensings (standing+walking, indoor, connected).

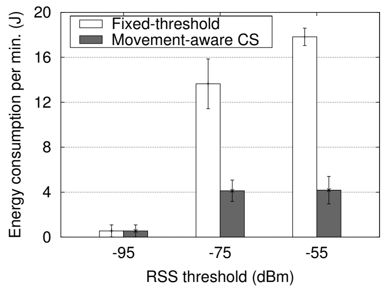


Fig. 13. Energy saving of movement-aware sensing (standing+walking, outdoor, connected).

For instance, compared to PS with 30 s interval, WiFisense saves energy by over 60 % and still reduces the average detection delay. This benefit essentially comes from WiFisense's ability to exploit the user's movement information obtained from the built-in accelerometer, and adapt the sensing parameter (d_{th}), accordingly.

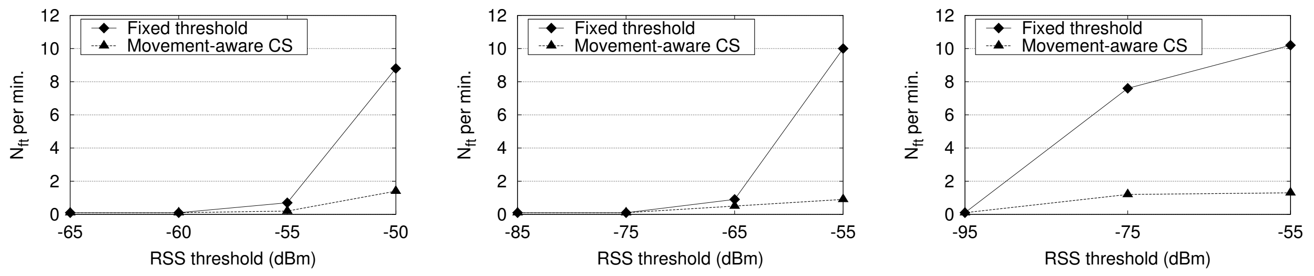
2) Energy benefit of movement-aware connected sensing: We now evaluate the energy saving benefits of the movement-aware connected sensing (CS) that adjusts the scan-triggering probability P_{scan} based on user's movement activity. In this experiment, we let a mobile user carrying a smartphone move around both inside and outside a building in the Los Altos testbed, and measure the average energy consumption (J per minute). During the experiments, we employ various combinations of network settings and user mobility scenarios: (i) indoor with standing for 10 minutes, (ii) indoor with standing and walking for 10 minutes, and (iii) outdoor with standing and walking for 10 minutes. As we set different triggering thresholds, we compare the energy consumption of movement-aware CS and FT. Note that we turn off r_t adaptation to study the benefit of P_{scan} and we further use different thresholds for each scenario to reflect different network characteristics.

Figs. 11, 12 and 13 show the average energy consumption that is derived from the average number of sensings and migrations (i.e., association to a new AP) per minute. As shown in the figures, movement-aware CS in WiFisense reduces energy consumption by 78 %, 79 %, and 77 % compared to FT in the above three scenarios (e.g., threshold is -55 dBm), respectively. This benefit mainly comes from WiFisense's ability to suppress unnecessary scanning based on user's movement information. In particular, as the RSS threshold (r_t) increases, FT suffers from *over-scanning*, while CS intelligently suppresses the scanning, thereby preserving energy. Note that we set the unit distance (d_{unit}) and β to 10 m and 0.1, respectively.

3) Pros & cons of movement-aware CS: While movement-aware CS can efficiently suppress unnecessary Wi-Fi sensing, it can also increase the chance of mis-detecting better WiFi connection opportunity, and vice versa. Here, we quantify the efficiency of movement-aware CS in terms of *false-triggering* (denoted as N_{ft}) and missed opportunity to access the best available Wi-Fi network (denoted as R_{mo}) by analyzing the trace from the previous experiments. N_{ft} represents the average number of Wi-Fi sensings that a mobile device has conducted in unit time (i.e., 1 minute) and has found no AP with higher RSS. This is an important metric to measure the efficiency of Wi-Fi sensing, since a too frequent false-triggering of sensing results in waste of energy. As a counterpart, R_{mo} represents the fraction of time during which the mobile is associated with suboptimal APs, compared to Oracle.

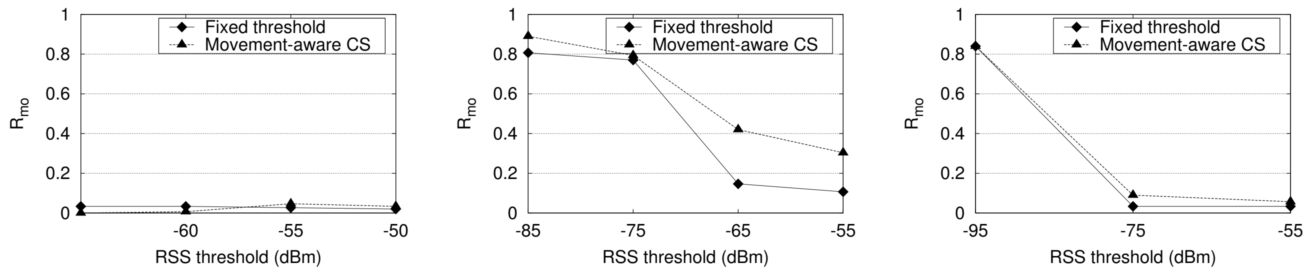
Fig. 14 shows the false-triggering rate with different fixed thresholds. As shown in the figure, the movement-aware triggering in WiFisense maintains a low false-triggering rate even with high RSS triggering threshold, indicating that movement-aware triggering performs well in finding Wi-Fi access opportunities. On the other hand, FT without using movement information suffers from frequent false-triggerings, resulting in wasted energy.

Fig. 15 shows another interesting aspect of movement-aware triggering. The figure shows R_{mo} in various scenarios. When a user is stationary, the missed opportunity R_{mo} is kept low regardless of the RSS threshold for triggering sensing (Fig. 15(a)). On the other hand, as the user moves, movement-aware CS misses Wi-Fi opportunities, especially when the threshold is set to a low value (i.e., -85 dBm), because the mobile device is being conservative in scanning while the RSS



(a) Scenario for standing, indoor (b) Scenario for standing, walking, indoor (c) Scenario for standing, walking, outdoor

Fig. 14. False-triggering rate of WiFisense vs. FT. Movement-aware triggering in WiFisense is effective in that it suppresses unnecessary triggering due to temporal RSS fluctuation (case for standing–(a)) and that it finds better APs as a user moves (cases for walking–(b),(c)).



(a) Scenario for standing, indoor (b) Scenario for standing, walking, indoor (c) Scenario for standing, walking, outdoor

Fig. 15. Missed opportunity rate of movement-aware WiFisense vs. FT. When a user is stationary (a), WiFisense and FT do not miss opportunity since he does not see new APs. However, when a user is moving (b), (c), both schemes causes association with suboptimal APs, missing better APs. This is because both schemes use a fixed threshold and can be solved by using dynamic threshold adaptation in WiFisense, whose impact will be shown in Section V-C4.

fluctuates significantly due to user's movements. This can be mitigated by dynamically adjusting its threshold depending on network environments, which we discuss next.

4) Adaptation of triggering thresholds: We now study the adaptation of triggering thresholds in CS. In particular, we evaluate how effectively the triggering parameters (i.e., RSS triggering-threshold, r_t, and triggering probability, P_{scan}) adapt to different contexts. In this experiment, a mobile user carrying a smartphone performs standing and walking activities inside and outside the Los Altos testbed for 15 minutes. We divide the experiment areas into four regions: i) region-1: standing, indoor, ii) region-2: walking, indoor, iii) region-3: standing/walking, outdoor, and iv) region-4: walking, indoor. While the smartphone performs CS over the areas, we record the progression of the two parameters over time.

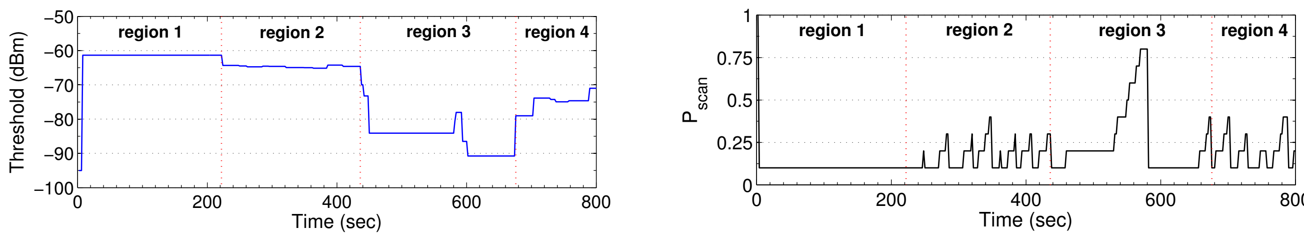
Fig. 16 shows the progression of the parameters during our experiment. As shown in Fig. 16(a), CS in WiFisense dynamically adapts its sensing-triggering threshold (r_t), depending on the network conditions. For instance, when a user is indoor (e.g., region 1) under high-RSS and multiple APs, CS increases the threshold r_t to make the best use of available APs, whereas when a user is outdoor with low-RSS APs (region 3), CS decreases r_t to not trigger sensing too often. Furthermore, WiFisense increments the triggering probability (P_{scan}) only when user moves a certain distance, i.e., d_{unit}, and reset it to 0.1 upon association to a new AP, as shown in Fig. 16(b). When a user is stationary (region 1), CS suppresses unnecessary scans by maintaining low P_{scan}, resulting in low false-triggering rate. On the other hand, when the user is moving (from outdoor to indoor), it increases P_{scan} to find better APs at new locations.

5) Effectiveness of joint adaptation of triggering thresholds: We further demonstrate the effectiveness of joint threshold adaptation in connected sensing. WiFisense dynamically adjusts the sensing-triggering parameters (r_t, P_{scan}) to find better APs in a given the network environment, while being energy-efficient. In the experiments, we let a mobile user running CS's adaptation walk in our Los Altos testbed indoor for 12 minutes. At the same time, we run FT with different triggering RSS-thresholds from -85 to -55. We measure the average RSS from APs with which each scheme suggests to associate, the total number of sensings, and the number of (re-)associations. Then, we compare how closely CS in WiFisense adjusts its sensing-triggering threshold to the threshold that provides an average RSS of Oracle. For an evaluation metric, we define and use false-triggering rate R_{ft}, which is defined as the number of failed association over the number of sensings, i.e., $R_{ft} = \frac{(N_{sense} - N_{assoc})}{N_{sense}} \times 100$.

Table II shows that the higher the fixed threshold (e.g., -55 dBm) the higher the achieved average RSS at the cost of sensing overheads. On the other hand, WiFisense achieves the average RSS close to the case of FT(-65), and further reduces the R_{ft} by up to 88 %. This benefit essentially comes from the joint adaptation of the triggering thresholds. For example, in the experiment region 1, 2 in in Fig. 16(a), WiFisense adaptively increases the scan triggering-threshold (r_t). As a result, CS maintains, on average, a low false triggering rate (4.3 %), compared to FT.

VI. Conclusion

This paper presents WiFisense, an energy-efficient Wi-Fi sensing system that maximizes the usage of Wi-Fi networks



(a) The progression of r_t

(b) The progression of P_{scan}

Fig. 16. Connected sensing (CS) adapts its scan-triggering thresholds, depending on user's movement and network environments. When a user is stationary and indoor, under high-RSS multiple APs (i.e., region 1), the RSS threshold is high and the movement probability is kept low. On the other hand, when a user is moving and outdoor with low-RSS APs (e.g., region 3), the threshold becomes low and the moving probability increases.

TABLE II
CORRECTNESS OF A TRIGGERING PARAMETER ADAPTATION

Methods	avgRSS	N_{sense}	N_{assoc}	R_{ft} (%)
`FT` (-85)	-67.17	4	3	25.0
`FT` (-75)	-61.43	14	13	7.1
`FT` (-65)	-59.04	38	24	36.9
`FT` (-55)	-58.77	174	26	85.1
WiFisense	-60.81	23	22	4.3

`FT` (x): Fixed threshold with RSS triggering threshold $r_t = x$.
avgRSS: Average RSS in dBm, R_{ft}: False triggering rate.
N_{sense}: A total number of sensings.
N_{assoc}: A total number of associations.

on smartphones. We first study optimal sensing interval via analysis, which reveals key factors, such as user's movement activity and local AP density, for energy-efficient Wi-Fi sensing. We then propose the WiFisense design that covers both disconnected and connected cases of Wi-Fi sensing. In its core, WiFisense uses a low-power accelerometer to infer user's movement, and further learns local network information to determine sensing frequency. Adaptive sensing-triggering algorithms are introduced to optimize Wi-Fi sensing frequency and to reduce false triggering. We implemented WiFisense on Android-based smartphones and evaluated its performance extensively. Our evaluation results show that WiFisense reduces energy consumption in scanning by up to 79 %, while reducing false-triggering to 4.3 %.

REFERENCES

[1] "Apple app store." http://www.apple.com/iphone/apps-for-iphone.
[2] "Android market." http://www.android.com/market.
[3] AT&T Faces 5,000 Percent Surge in Traffic, http://www.internetnews .com/mobility/article.php/3843001.
[4] T-Mobile's growth focusing on 3G, http://connectedplanetonline.com/ wireless/news/t-mobile-3g-growth-0130/.
[5] Smart phones are making Wi-Fi hotspots hot again, http://www.physorg.com/news180286896.html.
[6] Open Handset Alliance (Android), http://www.openhandsetalliance.com/.
[7] H. Wu, K. Tan, J. Liu, and Y. Zhang, "Footprint: Cellular Assisted Wi-Fi AP Discovery on Mobile Phones for Energy Saving," in *Proc. of ACM WiNTECH*, Sep 2009.
[8] G. Ananthanarayanan and I. Stoica, "Blue-Fi: Enhancing Wi-Fi Performance using Bluetooth Signals," in *Proc. of ACM MobiSys*, June 2009.
[9] R. Zhou, Y. Xiong, G. Xing, L. Sun, and J. Ma, "ZiFi: Wireless LAN Discovery via ZigBee Interference Signatures," in *Proc. of ACM MobiCom*, Sep 2010.
[10] I. Ramani and S. Savage, "SyncScan: Practical Fast Handoff for 802.11 Infrastructure Networks," in *Proc. of IEEE INFOCOM*, March 2005.
[11] M. Kim, Z. Liu, S. Parthasarathy, D. Pendarakis, and H. Yang, "Association Control in Mobile Wireless Networks," in *Proc. of IEEE INFOCOM*, April 2008.
[12] A. J. Nicholson and B. D. Noble, "BreadCrumbs: Forcasting Mobile Connectivity," in *Proc. of ACM MobiCom*, Sep 2008.
[13] E. Rozner, V. Navda, R. Ramjee, and S. Rayanchu, "NAPman: Network-Assisted Power Management for WiFi Devices," in *Proc. ACM MobiSys*, June 2010.
[14] F. R. Dogar, P. Steenkiste, and K. Papagiannaki, "Catnap: Exploiting High Bandwidth Wireless Interfaces to Save Energy for Mobile Devices," in *Proc. of ACM MobiSys*, June 2010.
[15] A. Rahmati and L. Zhong, "Context-for-Wireless: Context-Sensitive Energy-Efficient Wireless Data Transfer," in *Proc. of ACM MobiSys*, June 2007.
[16] D. Qiu, D. D. Lorenzo, S. Lo, D. Boneh, and P. Enge, "Physical Pseudo Random Function in Radio Frequency Sources for Security," Stanford University, Tech. Rep., Feb 2009.
[17] A. LaMarca and J. Hightower, "CRAWDAD trace intel/placelab/ placelab/downtown (v. 2004-12-17)," Dec. 2004.
[18] W. Wang and M. Zhao, "Joint Effects of Radio Channels and Node Mobility on Link Dynamics in Wireless Networks," in *Proc. of IEEE INFOCOM*, April 2008.
[19] T. S. Rappaport, "Wireless communications: Principles and practice, second edition," Jan. 2002.
[20] Z. Zhuang, K.-H. Kim, and J. Singh, "Improving Energy Efficiency of Location Sensing on Smartphones," in *Proc. of ACM MobiSys*, June 2010.
[21] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachar, and N. Sadeh, "A Framework of Energy Efficient Mobile Sensing for Automatic User State Recognition," in *Proc. of ACM MobiSys*, June 2009.
[22] E. Miluzzo, N. D. Lane, K. Fedor, R. Peterson, H. Lu, M. Musoles, S. B. Eisenman, X. Zheng, and A. T. Campbell, "Sensing Meets Mobile Social Networks: The Design, Implementation and Evaluation of the CenceMe Application," in *Proc. of ACM SenSys*, Nov 2008.
[23] Pedometer - Android app, http://code.google.com/p/pedometer/.
[24] Y. Choi and S. Choi, "Service Charge and Energy-Aware Vertical Handoff in Integrated IEEE 802.16e/802.11 Networks," in *Proc. of IEEE INFOCOM*, May 2007.
[25] Skyhook Wireless, http://www.skyhookwireless.com.
[26] J. Pand, B. Greenstein, M. K. D. McCoy, and S. Seshan, "Wifi-Reports: Improving Wireless Network Selection with Collaboration," in *Proc. of ACM MobiSys*, June 2009.
[27] V. Mhatre and K. Papagiannaki, "Using Smart Triggers for Improved User Performance in 802.11 Wireless Networks," in *Proc. of ACM MobiSys*, June 2006.
[28] A. J. Nicholson, Y. Chawathe, M. Y. Chen, B. D. Noble, and D. Wetherall, "Improved Access Point Selection," in *Proc. of ACM MobiSys*, June 2006.
[29] T. Goff, N. B. Abu-Ghazaleh, D. S. Phatak, and R. Kahvecioglu, "Preemptive Routing in Ad Hoc Networks," in *Proc. of ACM MobiCom*, July 2001.
[30] J. Strauss, D. Katabi, and F. Kaashoek, "A Measurement Study of Available Bandwidth Estimation Tools," in *Proc. of ACM IMC*, Oct 2003.
[31] L. M. Feeney and M. Nilsson, "Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment," in *Proc. IEEE INFOCOM*, April 2001.
[32] S. Mangold and L. Berlemann, "IEEE 802.11k: Improving Confidence in Radio Resource Measurements," in *Proc. of IEEE PIMRC*, Sep 2005.
[33] "Optimized link state routing." http://www.ietf.org/rfc/rfc3626.txt.
[34] G. Goertzel, "An algorithm for the evaluation of finite trigonometric series," *The American Mathematical Monthly*, vol. 65, 1958.
[35] "Qurinet: Quail ridge wireless mesh networks." [Online]. Available: http://spirit.cs.ucdavis.edu/quailridge
[36] OpenWRT, http://openwrt.org/.