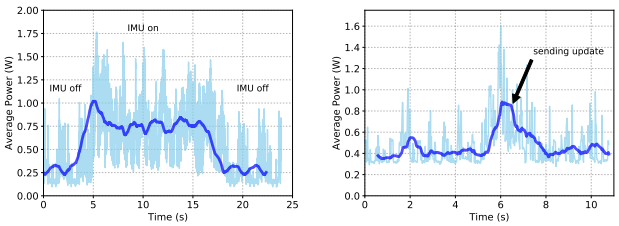
WiFi access points (APs). (3) We present a WiFi packets detection and collection method implemented on a set of off-the-shelves APs to detect the existing mobile devices within their coverage area and to collect the localization data. (4) We introduce a two-step position estimation algorithm which runs on the server side to aggregate the localization data and then calculates the current location of the mobile devices. In this realm, we adopt fingerprinting as our position fixing method. In the offline phase of fingerprinting, we adopt affinity propagation to cluster the collected RSSI measurements. Such clustering highly simplifies the positioning task via solely selecting the most relevant RSSI clusters. For the online phase, we provide a comparative study between adopting K-nearest neighboring and compressive sensing. (5) We present a proof-of-concept implementation and evaluation via a real-world scenario. To assess the performance of iSense, we consider two QoS measures, including the energy consumption of the mobile devices and the positioning accuracy. To this end, we constructed a testbed and several experiments were carried out in which iSense has been compared to three different position update protocols, namely dead-reckoning, time-based protocol, and distance-based protocol. The findings show a significant reduction in the energy overhead (up to 95%) on the mobile devices when participating in the crowd-sensing systems. Moreover, adopting the two-step localization method highly improves the position accuracy.

The remainder of this paper is structured as follows: Section II defines the tackled problem and formalizes our objectives. Section III presents the system model and the architectural framework of our system. Moreover, we introduce the WiFi traffic study through which we confirm the applicability of iSense in different scenarios. Section IV describes the device detection and data collection algorithm. The two-step localization approach is presented in Section V, before discussing the evaluation results in Section VI. Section VII reviews the related work in the realm of energy efficiency in crowd-sensing applications. Finally, Section VIII concludes the paper with an outlook on future work.

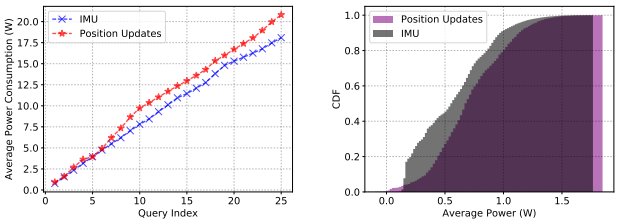
II. Problem Statement

In this section, we discuss the energy overhead on the mobile devices due to sensing and reporting their position information to the crowd-sensing servers. It is worth to mention here that this paper deals with the latter setting, where localization and position update messages dominate all other contributions to energy dissipation, such that techniques like adaptive sampling, compression, or piggybacking have limited impact. To distribute the sensing queries in crowd-sensing applications, the servers have to be aware of the mobile devices in the sensing area. This position knowledge is typically acquired by the mobile devices through continuously sensing their location and then sending position update messages to the corresponding servers. However, the task of frequently turning on the navigation system is inefficient in terms of the consumed energy. Additionally, reporting this information to the server has many drawbacks, including: (1) significantly increasing the energy consumption of the mobile devices, (2) these reports may be superfluous when the devices are outside the sensing area, and (3) frequently reporting the position information subjects the mobile devices to third-party attacks which may reveal the users identity.

Figure 1 depicts the energy overhead of activating the inertial measurement unit (IMU), i.e. accelerometer, gyroscope, and gravity sensors, and of sending the position updates via a WiFi network. A `Monsoon` power monitor tool has been connected to a Galaxy Nexus 4 mobile device to record the power consumption every 100 ms. Figures 1a and 1b depict the power peaks due to activating the IMU sensors and the WiFi module, respectively. Figure 1c shows the cumulative power for both tasks, i.e. sensing and reporting the position. It is obvious that continuously performing these two tasks add significant energy overhead on the mobile device. Whereas, Figure 1d show the cumulative distribution function (CDF) of the average power consumed while performing these two tasks. The figure demonstrates that activating the IMU module nearly consumes a similar amount of energy to that required for communication. Hence, reducing the number of updates while continuously sensing the position burdens the mobile devices. In this paper, we seek to reduce this overhead via adopting the iSense framework.



(a) Activating the IMU sensors (b) Sending position updates



(c) Cumulative power consumption (d) CDF of the power consumption

Fig. 1: Power consumption of position estimation and update reports

Formally, the tackled problem can be formulated as denoted by Equation 1. The main objective is to minimize the overall energy overhead of each participating mobile device $m \in \mathbb{M}$ where the symbols E_s, E_c, and E_u are the energy consumption required to perform data acquisition, positioning computations, and reporting the position information to the servers, respectively. This objective is governed by a quality condition which dictates that the geographic position of a mobile device P_s stored at the server as x–y coordinate has to be approximately

similar to the actual position $P_{current}$ of this mobile device.

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{m} (E_s(i) + E_c(i) + E_u(i)) \\ &\text{subject to} \quad P_s \cong P_{current} \quad \forall\ m \in \mathbb{M} \end{aligned} \tag{1}$$

III. System Overview

This section introduces the system architecture along with our assumptions. Assume an indoor environment equipped with N WiFi access points (AP), each covering an area of radius R meters. These areas are deliberately overlapped to guarantee full coverage. For simplicity, we assume a single crowd-sensing server which is physically connected to the scattered APs. We assume that these APs support a lightweight embedded Linux distribution such as `OpenWRT`[1] and `DD-WRT`[2]. Accordingly, we can easily run an application on each AP to monitor the surrounding area. The monitoring involves: (1) detecting all mobile devices that can be seen by the APs, (2) recording the RSSI measurements, and (3) forwarding an update to the back-end server. The building has M mobile users who are freely traversing the various floors. Figure 2 depicts the architecture of the proposed crowd-sensing framework.

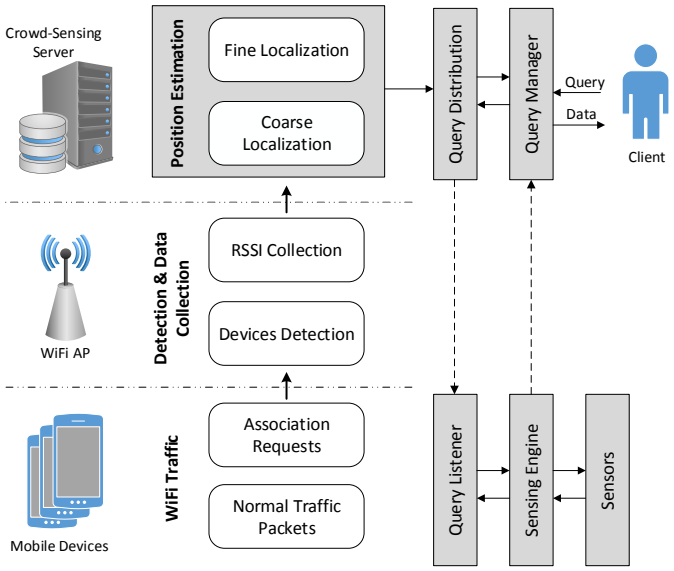


Fig. 2: System Architecture

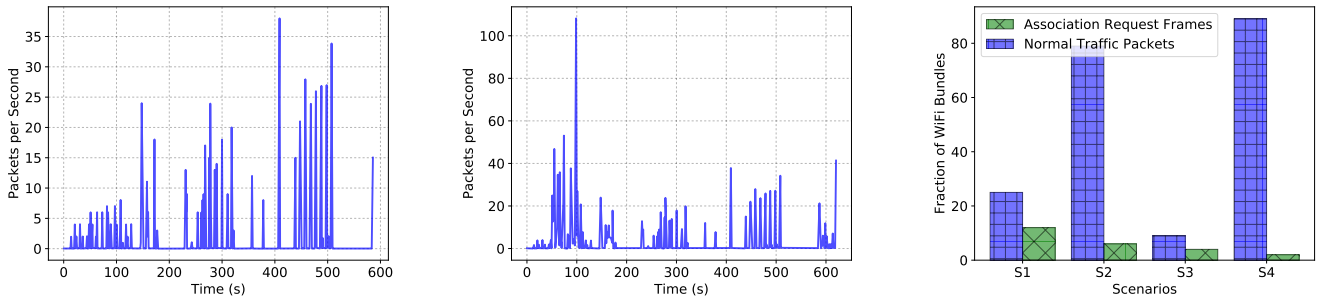
On receiving a sensing query q from the client, the *query manager* forwards the query q to the mobile devices that are located in the sensing area. As soon as the query manager receives the required sensor data from some mobile devices, it sends a termination message to all mobile devices that initially received q to stop the execution of the query. At the mobile devices, the *sensing engine* component is mainly responsible for reading the sensors data and reporting these readings to the back-end server. To distribute the query q, the *query distribution* component identifies a subset of mobile devices whose locations overlap with the sensing area. To this end, the server employs a two-step localization method, namely *coarse* and *fine* position sensing.

Before delving into the technical details of iSense, we have to examine the existence of sufficient WiFi traffic for position sensing. In fact, we conducted a series of experiments to collect the wireless traffic between a mobile device and a set of APs in different scenarios. For this study, we used the Wireshark[3] tool installed on an Apple MacBook. The MacBook laptop was configured to work in the monitoring mode. We employed a Google Nexus 4 smart phone as the targeted mobile device. The MacBook acts as a third party device which listens to the traffic between the mobile device and the various APs. We classified the collected signals into: (1) probe request messages, and (2) normal traffic from the typical mobile Apps such as Youtube and Facebook. To capture and analyze the network traffic in the low-level layer, the IEEE `radiotap` header—which provides information about the various IEEE802.11 frames such as the MAC timestamp and RSSI values—has been parsed. In order to capture the `radiotap` header, the IEEE 802.11 wireless card has to be adjusted into the monitoring mode which allows sniffing all the packets in the surround area. The monitoring mode is inherently supported in some wireless network interface cards where a machine that operates in the monitoring mode runs as a listener.

During the experiments, several scenarios were examined involving different mobility patterns and different workloads, i.e. activities running on the mobile device. Specifically, four scenarios were considered under different conditions: (1) **Scenario 1**, mobility without activities (i.e. only background services); (2) **Scenario 2**, mobility with activities (i.e. web browsing, video streaming); (3) **Scenario 3**, stationary without activities; and (4) **Scenario 4**, stationary with activities. The first two scenarios consider collecting WiFi packets during the movement of the mobile device. Whereas, the latter scenarios were achieved via keeping the mobile device on a table. For each data sampling, we ran the experiment for ten minutes. Subsequently, the collected traffic has been filtered out based on the MAC address of the smart phone. This filtering step is crucial to solely consider the traffic from/to our mobile device under test while discarding signals from other sources.

Figure 3a depicts the probe requests (aka association packets) while the mobile device was moving and no activities were running. As the figure shows, there exist plenty of wireless packets thanks to the device movement which intrinsically imposes a continuous handover between the neighboring APs to maintain the wireless connectivity. Figure 3b delineates the normal uplink traffic in the first scenario. Despite disabling the mobile applications, we found tens of packets are still transmitted to the APs owing to the background services. Nevertheless, there exist silence periods with no traffic, as depicted in Figures 3a-3b. To summarize the results of the four

[1]https://openwrt.org/
[2]http://www.dd-wrt.com/site/index
[3]https://www.wireshark.org/



(a) Probe requests in the first scenario

(b) Normal uplink traffic in the first scenario

(c) Summary of the recorded WiFi packets in all scenarios

Fig. 3: Results of WiFi traffic analysis in different scenarios

scenarios, Figure 3c delineates the fraction of recorded WiFi bundles, where a bundle comprises 20 WiFi packet per second. Due to enabling some activities, the traffic size is highly increased in the second scenario. In the third scenario, we noticed a periodic behavior of the probe requests. Although the number of probe requests are dropped due to the connection to a single AP, there still exists a sufficient traffic volume thanks to the background services of the mobile device (cf. Figure 3c). In iSense, the APs trigger the mobile devices to send *forced* probe requests when silence periods are detected (cf. Section IV). In the fourth scenarios, the traffic volume is tremendously increased while running the various mobile applications. It is important to mention here that the amount of the probe request frames varies based on the setting and the state of the mobile devices, i.e. their mobility pattern. To conclude, the already-existent WiFi traffic can sufficiently be utilized for position sensing at the back-end server.

IV. Detection & Data Collection

In this section, we explain the processing steps—performed on each AP—to detect the mobile devices and to record the required positioning data. In fingerprinting, sensing the position of a mobile device using a single AP may negatively affect the positioning accuracy. Hence, the algorithm has been designed to handle two classes of the mobile devices in each AP, namely (1) the *associated* devices which are already in direct connection with the AP, and (2) the *unassociated* devices which exist in the coverage area of the AP but are connected to neighboring APs. In the former class, a handler has been implemented to frequently read the associated mobile devices list on each AP. To handle the latter class, iSense analyzes the flight packets in the surround area to extract the MAC address and the RSSI value of each packet. To aggregate the readings from different handlers, a measurement pool is constructed. Figure 4 describes the detection and data collection algorithm. This algorithm takes the MAC addresses $\mathbb{A}$ of the participating mobile devices as an input. At the outset, two parallel threads are thrown to capture the RSSI values which are then stored in two lists, namely the associated mobile devices list (AML) and the unassociated

Require: MAC address list of the participating devices $\mathbb{A} = \{A_1, \cdots, A_M\}$, threshold ψ_p, time threshold ψ_t

1: UML $\leftarrow \emptyset$ ▷ Initialization
2: AML $\leftarrow \emptyset$
3: **instantiate** a wireless interface ▷ UML preparations
4: **instantiate** a PCAP interface & a network filter
5: **while** true **do**
6: AML $\leftarrow$ readAssociatedList($\mathbb{A}$)
7: UML $\leftarrow$ filterPCAP($\mathbb{A}$)
8: pool $\leftarrow$ klamanFilter(AML $\cup$ UML) ▷ updating the measurement pool
9: **for all** MAC address $A_i \in \mathbb{A} \cap$ pool **do**
10: **if** $|O^{A_i}| < \psi_p$ **or** $\tau_{j,k}^{A_i} > \psi_t$ **then**
11: **trigger** a probe request
12: **else** ▷ reducing the redundancy
13: $O^{A_i} \leftarrow$ centroid(O^{A_i}, ψ_t)
14: **end if**
15: **end for**
16: **forward** pool to the crowd-sensing server
17: **end while**

Fig. 4: Detection and data collection algorithm

mobile devices list (UML). As an initialization, the AML and the UML lists are set to $\emptyset$, i.e. null value (l. 1).

The AML handler uses the `iw` configuration utility[4] to extract the RSSI values and the MAC addresses (l. 6). Subsequently, it forwards this data to the measurement pool for further processing. For the unassociated mobile devices, the existing hardware does not support the direct recording of their RSSI values. As a workaround, the entire network traffic is dumped to extract the required information. The RSSI values are typically placed in the *RadioTap* header[5] which exists in the lower sub-layers of the MAC layer. To capture this header, a new wireless network interface has been created and is configured to work in the monitoring mode (l. 3, 4). Then, the UML handler utilizes the `PCAP` analyzer which provides APIs for capturing and filtering the network traffic (l. 7). The `PCAP`

[4]https://wireless.wiki.kernel.org/en/users/Documentation/iw
[5]http://www.radiotap.org/

APIs belong to the `libpcap` library[6] in the Unix system. Such a library provides two types of capturing, including: (1) capturing the network traffic and then saving it into a file in the pcap format for later filtering and analysis, and (2) online capturing and analyzing which provides a packet handler for each captured packet.

To improve the quality of the collected readings, iSense pre-processes them in a measurement pool before being sent to the crowd-sensing server. In this regard, a *Kalman filter* [4] is employed to eliminate the noise along with tracking the position changes of the mobile devices in the light of the preceding readings (l. 8). To overcome the silence periods in which the received observations O^{A_i} for each MAC address $A_i \in \mathbb{A}$ are less than a packet threshold ψ_p, the APs deliberately trigger the mobile devices to send probe requests to ensure connectivity (l. 11). Furthermore, the algorithm checks whether the time τ^{Ai} between two observations O_j, O_k does not exceed a time threshold ψ_t. Otherwise, the mobile devices are also triggered to obtain fresh observations (l. 11). To reduce the redundancy of the observations for each MAC address, the algorithm determines the centroid point within a time window specified by the threshold ψ_t. Finally, each AP forwards the collected data as a JSON object to the crowd-sensing server. To improve the position accuracy obtained by fingerprinting, iSense considers observations from at least two associated/unassociated APs (cf. Section V). Figure 5 visualizes the data flow between the various processing steps.

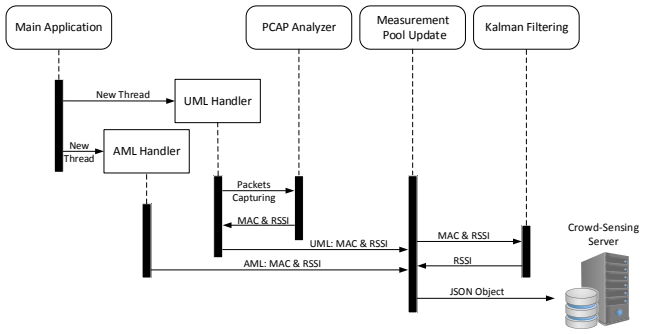


Fig. 5: Data flow diagram of the detection and data collection algorithm

V. Position Estimation

In this section, we discuss the implementation details of the two-step position sensing method. It is important to mention that this method is inspired from the work done in [5]. Feng et al. [5] propose a compressive sensing-based localization method through solving an ℓ_1-minimization problem. However, solving these convex optimization problems on the mobile devices is hardly viable owing to the limited computing power and energy budget. In contrast to [5], iSense provides a realistic implementation of the approach on the crowd-sensing servers. Figure 6 shows the processing steps performed on the crowd-sensing server to localize the participating mobile devices. To estimate the position information, we utilize the fingerprinting approach which comprises two phases, including: (1) an *offline* phase in which a radio map is constructed and is clustered to reduce the search space while matching the online observations to the radio map, and (2) an *online* phase in which the recorded observations are matched to a certain cluster (i.e. coarse localization), before the geographic position is precisely determined through formulating the localization task as a compressive sensing problem (i.e. fine localization). Below, we explain each component of the localization method.

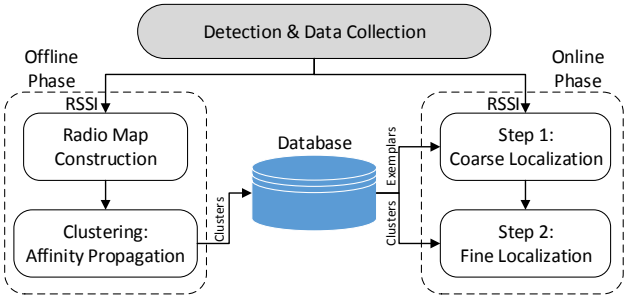


Fig. 6: Architecture of the position sensing method

A. Offline Phase

In this section, we discuss the radio map construction in more detail, before we describe the radio map clustering using the affinity propagation method.

1) Radio Map Construction: In fingerprinting, localization occurs through matching the captured RSSI readings with a radio map of the indoor environment. In iSense, the APs measure the RSSI values and forward them to the crowd-sensing server. For the crowd-sensing server to construct a reliable radio map, the geographic area of the indoor environment has to be divided into a set of small cells. Each cell has a *reference point* (RP) which is identified by its x-y coordinates. For each RP i, the AP j collects data samples $\psi_{i,j}^{(o)}(\tau), \tau = \{1, \cdots, t\}, t > 1$ by pointing the mobile device to different orientations $o \in \mathcal{O} = \{0°, 90°, 180°, 270°\}$ where t denotes the number of collected samples. The collected RSSI values in each RP are then averaged $\psi_{i,j}^{(o)} = \frac{1}{t}\sum_{\tau=1}^{t}\psi_{i,j}^{(o)}(\tau)$ forming a training record. The data collection procedure is repeated till all RPs in the radio map are visited. Accordingly, the radio map—for each orientation o—is constructed as a set of training records denoted by

$$\Psi^{(o)} = \begin{bmatrix} \psi_{1,1}^{(o)} & \psi_{1,j}^{(o)} & \cdots & \psi_{1,N}^{(o)} \\ \psi_{2,1}^{(o)} & \psi_{2,j}^{(o)} & \cdots & \psi_{2,N}^{(o)} \\ \vdots & \psi_{i,j}^{(o)} & \ddots & \vdots \\ \psi_{L,1}^{(o)} & \psi_{L,j}^{(o)} & \cdots & \psi_{L,N}^{(o)} \end{bmatrix} \tag{2}$$

where $\psi_{i,j}^{(o)}$ is the average RSSI value collected from AP j at RP i, $j \in \mathbb{R}^L$, L is the number of RPs in the radio map, and

[6]http://www.tcpdump.org/

N is the total number of APs connected to the crowd-sensing server.

2) Data Clustering: Clustering the radio data is primarily achieved to improve the positioning accuracy as well as to reduce the number of RPs and RSSI values required for sensing the position in the online phase. In this regard, we utilize the affinity propagation method [6] which exchanges update messages between the data points to identify a set of "exemplars", i.e. center points of the RPs that are representative of clusters. Initially, the algorithm calculates a similarity matrix $S \in \mathbb{R}^{(M,M)}$ between all RPs in the radio map Ψ.

$$S(i,k) = -||\text{RP}_i - \text{RP}_k||^2 \quad \forall\ i,k \in \{1,2,\cdots,M\} \tag{3}$$

Equation 3 denotes the Euclidean distance used for measuring the similarity between a candidate exemplar and the other RPs in the radio map. Subsequently, the algorithm considers all the RPs as potential exemplars through assigning them a preference value estimated as the median of the similarity matrix S. To select a probable exemplar, the data points recursively exchange two types of update messages, namely (1) the *responsibility* messages $R \in \mathbb{R}^{(L,L)}$, and (2) the *availability* messages $A \in \mathbb{R}^{(L,L)}$. First, the responsibility message $R(i,k)$ quantifies how well-suited RP k is to be selected as an exemplar for RP i relative to other candidate exemplars. Second, the availability matrix $A(i,k)$ that represents how "appropriate" it would be for RP i to pick RP k as its exemplar is updated. These update messages are continuously exchanged till the cluster boundaries remain unchanged over a number of iterations. The exemplars are extracted from the final matrices as those whose self-responsibility plus self-availability is positive (i.e. $r(i,i)+a(i,i) > 0$). After identifying the exemplars, each RP in the radio map is assigned to a cluster via measuring its Euclidean distance to all available exemplars. Accordingly, the output of this algorithm is a set of exemplars $H \in \mathbb{R}^v$ plus their corresponding clusters $\mathcal{C} = c_1, \cdots, c_v$ that are to be used in the online phase.

B. Online Phase

In this phase, the server collects the RSSI values from each AP and then it calculates the exact location. Below, we discuss the two-step localization algorithm implemented on the crowd-sensing server.

1) Coarse Localization: The core idea behind splitting the positioning task into two steps is to reduce the search space while matching the online RSSI values with the various radio map clusters. Since fewer RPs are considered, the coarse localization limits the maximum error to the selected subset. In the coarse localization step, we compute the similarity $s(\psi_r, \psi_j)$ between the online RSSI ψ_r and each exemplar $\psi_j \in H$ to decide which cluster it belongs to. In lieu of selecting one cluster, we keep the best-matched clusters that have the highest similarities. In this manner, the algorithm improves the accuracy by considering the cases in which the mobile device is close to the cluster boundaries. Accordingly, a new clusters set $\widetilde{\mathcal{C}} \subset \mathcal{C}$ is produced along with their exemplars $\widetilde{H}$. Hence, the final output of this step represents a partial radio map matrix given by

$$\widetilde{\Psi} = \psi_j^{(o)} \qquad \forall\ (j,o) \in \widetilde{\mathcal{C}}. \tag{4}$$

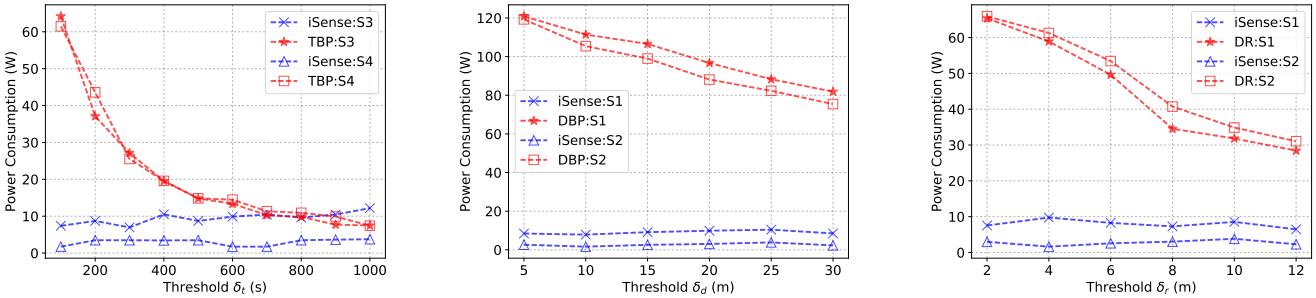
2) Fine Localization: As an example of the advanced methods which can be implemented on the crowd-sensing server, we utilize *Compressive Sensing* (CS) for providing a fine-grained position sensing. The core idea behind CS is to enable sampling below the *Nyquist* rate while offering precise recovery [7]. Generally, CS relies on two main principles: *sparsity* of the signals of interest, and *incoherence* which pertains to the sensing modality. A signal $X \in \mathbb{R}^s$ is said to be K-sparse, if it consists of K non-zero elements where $k \ll s$. Fortunately, the localization problem has sparse nature where the RSSI values at a given RP are unique in the discrete spatial domain. Hence, if we assume that the mobile device is ideally located at one of the RPs, the position sensing task can be formulated as a CS problem by

$$y = \Phi \times \widetilde{\Psi} \times x + \epsilon \tag{5}$$

where y is the online RSSI vector, Φ is a binary sparse measurement matrix, $\widetilde{\Psi} \in \mathbb{R}^{\widetilde{L},\widetilde{N}}$ is the partial RSSI matrix that is generated in the coarse localization phase given that $\widetilde{L} < L, \widetilde{N} < N$. The term ϵ is the measurement noise due to RSSI deviations and $x = [0,\cdots,1,0,\cdots,0]^T$ is a 1-sparse vector representing the device location. The measurement matrix Φ enables the crowd-sensing server to select k APs which are closer to the mobile devices. Consequently, the k-sparse matrix Φ can be estimated once the indices of the k-nonzero elements are found. To this end, we descendingly sort the online RSSI vector ψ_r, and then consider only the APs corresponding to the least indices. Generally, applying the CS theory requires the matrices Φ and $\widetilde{\Psi}$ to be incoherent, i.e. columns of these matrices have to be nearly orthogonal. Once the incoherence condition is satisfied, the devices position can be precisely estimated by solving the CS problem using ℓ_1-minimization solvers. If the recovered vector $\widehat{x}$ is a 1-sparse vector, it means the mobile device is exactly located at this RP. Otherwise, the position can be determined as the centroid point of the selected RPs in $\widehat{x}$.

VI. Performance Evaluation

To demonstrate the effectiveness of iSense, we tested our system in a real-world scenario. The goal behind our experiments is to obtain a real radio map from a geographic area that reflects a typical setting of indoor environment. Then, we collect a set of test points in order to study the performance of iSense in various scenarios. We first describe the setup of our evaluation, before we discuss the evaluation results in various scenarios. The performance is assessed in terms of the energy consumption of the mobile devices participating in the crowd-sensing framework. Furthermore, we evaluate the localization accuracy of the CS approach relative to a *K-nearest neighbors* (K-NN) approach.



(a) Comparison with the time-based protocol (b) Comparison with the distance-based protocol (c) Comparison with dead reckoning

Fig. 7: Energy performance of iSense relative to various position update protocols

A. System Setup

To collect the required data samples, several experiments were carried out on the second floor of the IPVS institute, University of Stuttgart, Germany. We utilized a commercial off-the-shelf hardware that can execute C/C++ code and also supports the monitoring mode. The testbed consists of five APs (Linksys WRT 54GL) which were connected together with a back-end server via an Ethernet LAN 8-ports switch. In each AP, the detection and data collection algorithm has been implemented to collect RSSI values from a Galaxy Nexus (GT-I9250) smart phone with a dual-core CPU running at 1.2 GHz. To run our algorithm on the APs, we utilized the `OpenWRT` firmware, which is an open source Linux distribution for embedded devices. We installed all the packages that are required to run the application such as `pthread`, `libstd`, and `libpcap`.

To construct the radio map, we divided the area into a set of cells where each cell comprises a single RP. The distance between the neighboring RPs were set to two meters. The mobile device was carried by a volunteer who freely moved between the RPs. The radio map has been generated at 35 RPs from different orientations (i.e. $0°, 90°, 180°, 270°$). We set the packet threshold ψ_p to 20 and the time threshold ψ_t to 10 second. The data collection procedure was repeated four times for each RP till all RPs in the geographical area were scanned. After constructing the radio map, a set of 17 test points were selected at random locations to evaluate the localization accuracy. The RSSI samples at such test points were collected by capturing the probe request packets and the normal traffic, i.e. web browsing.

B. Energy Consumption

In this section, we quantify the energy consumption of the mobile devices participating in an indoor crowd-sensing application. As baseline methods, we compare the energy performance of iSense to three position update protocols namely time-based protocol (TBP), distance-based protocol (DBP), and dead-reckoning (DR) [8]. The former protocol relies on periodically updating the servers with the current position. The main shortcoming of this approach emerges from the lack of updated position information at the servers during these periods. Alternatively, the distance-based protocol triggers a new position update if the Euclidean distance between the current location and the last reported location exceeds a certain threshold δ_d. However, the profit of adopting this method highly depends on the mobility pattern of the mobile devices. Finally, the dead-reckoning approach is an optimization of the distance-based protocol where the position at the server evolves with time. In other words, the servers estimate the current position of the mobile devices based on their old positions, speed, and direction. Alternatively, iSense entirely sidesteps the need to continuously active the sensors to collect the position information.

For the energy measurements, we adopted two strategies while considering the aforementioned four scenarios, i.e. two mobile (S1, S2) and two stationary (S3, S4). In the stationary scenarios, we utilize our `Monsoon` power monitor tool[7] to collect the energy data of the Nexus smart phone. In the mobile scenarios, it was hard to connect the device to the power monitor tool. Therefore, the energy measurements were achieved through logging the battery's current and voltage. During the execution of each method, these current and voltage values associated with timestamps were recorded every 100 ms. The energy measurements were performed while no other background activity existed. The screen energy (at 50% brightness level) has been subtracted from the total energy consumption to consider only the energy overhead of running the algorithms. We ran each experiment for 30 minutes with repeating them four times and the resultant values are then averaged.

In the three baseline methods, we measure the energy consumption of sensing the position and sending the update messages. In iSense, we measure the energy consumption of sending probe requests during the silence periods. Figure 7a depicts the energy consumption due to localization while adopting iSense and the TBP approach. Since the TBP approach solely relies on the time difference, we selected to perform the comparative study in the two stationary scenarios, i.e. S3 and S4. It is worth to mention that the TBP approach schedules the position sensing task directly before sending the updates. As it can be seen in the figure, iSense consumes much

[7]https://www.msoon.com/LabEquipment/PowerMonitor/

less energy than the TBP approach (at least by 55.5% for S3 and 86% for S4) with a maximum variance of 0.65. We also note that iSense in S4 consumes on average 68% less energy than its consumption in S3 due to the existence of large WiFi traffic in S4.

Figure 7b depicts a comparison between the energy consumed by iSense and the DBP approach in S1 and S2. For the DBP approach, we selected to sense the position every 10 seconds and then check the violation of the distance threshold δ_d. Due to frequently sensing the position, the DBP approach consumes on average 91.8% more energy than iSense with a maximum variance of 1.15. In fact, iSense in S1 and S2 exploits the device mobility which increases the WiFi traffic thanks to the continuous handover between the neighboring APs. Similarly, Figure 7c compares the energy consumption for iSense and the DR approach. In the DR approach, the mobile device senses its position whenever the difference between the predicted position and the last measured position exceeds a threshold δ_r. Again, iSense outperforms the DR approach where it significantly reduces the energy consumption (by at least 82.5% in S1 and 94% in S2) with a maximum variance of 0.8.

Finally, Figure 8 depicts the fraction of forced probe requests which were triggered by the APs during the silence periods. The figures show these forced requests in the four tested scenarios. As expected, the APs did not request many probe requests in S1 and S2 due to the mobility of the mobile devices as well as running the activities. Alternatively, the APs triggered more probe requests in S3 and S4 due to maintaining the connection to a single AP. However, we notice that even the worst case in S3 and S4 has a minor impact on the energy consumption, as depicted in Figures 7b and 7c.

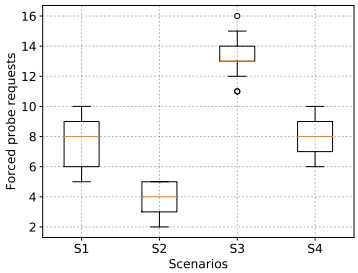
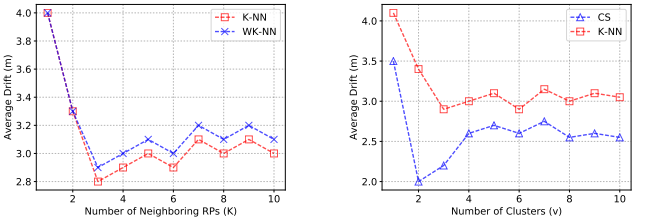


Fig. 8: Fraction of the forced probe requests

C. Localization Accuracy

In this section, we evaluate the two-step position sensing algorithm in terms of the average localization drift estimated as the Euclidean distance between the actual location of the volunteer and the estimated position. To clearly understand the results, we compare the CS-based method with a localization method based on the k-nearest neighbors (K-NN) algorithm [9]. In the K-NN algorithm, the mobile device location is determined by matching the online RSSI vector ψ_r to K RPs in the partial radio map $\widetilde{\Psi}$. Specifically, the location is estimated as the centroid point between K RPs which have the minimum Euclidean distance. Figure 9a demonstrates the localization drift obtained while adopting various K values. This figure serves as a calibration step to obtain the most "well-suited" number of nearest neighbors K. Furthermore, the figure also compares the performance of the naïve K-NN algorithm and a weighted (WK-NN) version in which higher weights are given to the closest RPs. As it can be seen in the figure, both K-NN and WK-NN have approximately similar performance. The figure also shows that setting $K = 4$ achieves—in our settings—a stable performance with small localization error.



(a) Estimating the most proper k value for K-NN algorithm

(b) Performance of the CS approach relative to K-NN

Fig. 9: Localization accuracy

Figure 9b quantifies the accuracy of the CS-based position sensing method relative to the K-NN method for a different number of radio clusters. The figure shows that increasing the number of clusters reduced the localization drifts till $v = 4$. After this value, the localization drift steadily remains stable in both adopted methods. In addition, the figure demonstrates that the CS method reduces the average error by at least 18%. Specifically, the CS-based method achieves a localization error of 2.5 m when the number of cluster increases beyond 4. It is also important to mention that increasing the number of APs highly improves the localization performance. To summarize, we found that the CS method achieves better localization relative to the K-NN method for different numbers of APs.

D. Discussion

Apparently, the evaluation results show that iSense outperforms the baseline methods in terms of the energy overhead on the mobile devices. It is important to also mention that iSense keeps the position information up-to-date at the crowd-sensing servers. Alternatively, the servers have to wait for the position updates in the baseline methods. The main shortcoming of iSense emerges owing to detecting and tracking the MAC address of the participating mobile devices. In this paper, we assumed that the crowd-sensing servers are trusted. However, this assumption may not hold in other situations. To overcome such a challenge, the user identity can be protected via increasing the time between subsequent data sampling. In other words, adopting an on-demand strategy to solely sample the WiFi data whenever a sensing query is to be distributed. Furthermore, modern devices will utilize MAC address randomization techniques to frequently change their

MAC addresses [10]. Once these randomization techniques become popular, the task of data sampling in iSense will not contradict with users' privacy.

VII. Related Work

In this section, we review the recent efforts for tackling the energy efficiency problem in crowd-sensing applications. In this realm, the previous efforts can be classified into two main categories: (1) reducing the update messages, and (2) reducing the energy waste of position sensing on the mobile devices. In the first category, the position updates can be reduced through piggybacking them with other processes or via adopting predictors [11]–[13]. In [11], we propose an opportunistic update protocol that appends the update messages with other messages exploiting the tail time of the cellular network interfaces. Although this approach showed a significant reduction in the energy burden, it is still limited to delay-tolerant crowd-sensing applications. Chen et al. [12] propose a simulation-based prediction model to reduce the update frequency. To this end, they model the road networks by graphs of cellular automata to predict the objects movements. Based on these predictions, the objects are grouped and only the central object in each group reports its position to the server. In contrast, iSense entirely moves this burden to the crowd-sensing server while maintaining nearly up-to-date position information.

In the second category, the ideas mostly revolve around adopting low-power sensors as well as deactivating the sensors in unnecessary situations. In [14], we propose—in an indoor mapping application—a framework that enables the mobile devices to deactivate their localization sensors, i.e. inertial navigation system, in areas that have been mapped already with high quality. Kjærgaard et al. [15] introduce a sensor management strategy that continuously utilizes low-power sensors, i.e. compass measurements, to sense the trajectory. Based on these measurements, the system evaluates the necessity to use high-power sensors, i.e. the GPS sensor. However, running the low-power sensors continuously consumes a considerable amount of energy. For instance, the gyroscope sensor embedded in IPhone 4 consumes, on average, 175.6 mW [16]. In contrast to all these approaches, iSense thoroughly moves the positioning burden to the back-end servers. Hence, the mobile devices avoid wasting their energy while being involved in crowd-sensing tasks.

VIII. Conclusion & Future Work

In this paper, we presented iSense, an energy-aware framework that relieves the energy burden on the mobile devices participating in crowd-sensing applications. To this end, iSense exploits the already-existent wireless traffic to enable the crowd-sensing servers from detecting the available mobile devices in the sensing area. We demonstrated the applicability of iSense through several traffic analysis experiments. We discussed how iSense improves the localization accuracy by detecting both associated and unassociated mobile devices even if the mobile devices are inactive. Furthermore, sensing the position at the crowd-sensing servers allowed the adoption of complex localization methods like the CS-based localization. The experimental results showed a significant reduction of the energy consumption of the mobile devices while limiting the localization error below 2.5 meter. So far, we investigated some QoS measures such as energy consumption and localization accuracy. In the future, we will analyze the privacy issues to sidestep revealing the users identity. Furthermore, we plan to investigate other localization methods to further improve the positioning accuracy. Finally, we will extend our evaluations via considering the outdoor scenarios.

References

[1] D. Wang, T. Abdelzaher, and L. Kaplan, *Social sensing: Building Reliable Systems on Unreliable Data*. Elsevier Science, 2015.

[2] D. Philipp, P. Baier, C. Dibak, F. Drr, K. Rothermel, S. Becker, M. Peter, and D. Fritsch, "Mapgenie: Grammar-enhanced indoor map construction from crowd-sourced data," in *2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, March 2014, pp. 139–147.

[3] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *IEEE Communications Magazine*, vol. 48, no. 9, pp. 140–150, Sept 2010.

[4] G. Welch and G. Bishop, "An introduction to the kalman filter," Chapel Hill, NC, USA, Tech. Rep., 1995.

[5] C. Feng, W. S. A. Au, S. Valaee, and Z. Tan, "Received-signal-strength-based indoor positioning using compressive sensing," *IEEE Transactions on Mobile Computing*, vol. 11, no. 12, pp. 1983–1993, Dec 2012.

[6] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *science*, vol. 315, no. 5814, pp. 972–976, 2007.

[7] K. Bryan and T. Leise, "Making Do with Less: An Introduction to Compressed Sensing," *SIAM Review*, vol. 55, no. 3, pp. 547–566, 2013. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/110837681

[8] A. Leonhardi and K. Rothermel, "A comparison of protocols for updating location information," *Cluster Computing*, vol. 4, no. 4, pp. 355–367, Oct 2001.

[9] P. Bahl and V. N. Padmanabhan, "Radar: an in-building rf-based user location and tracking system," in *Proceedings of the IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, vol. 2, 2000, pp. 775–784.

[10] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. C. Rye, and D. Brown, "A study of MAC address randomization in mobile devices and when it fails," *CoRR*, vol. abs/1703.02874, 2017. [Online]. Available: http://arxiv.org/abs/1703.02874

[11] P. Baier, F. Dürr, and K. Rothermel, "Opportunistic position update protocols for mobile devices," in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '13. ACM, 2013, pp. 787–796.

[12] J. Chen, X. Meng, B. Li, and C. Lai, "Tracking network-constrained moving objects with group updates," in *Proceedings of the 7th International Conference on Advances in Web-Age Information Management*, ser. WAIM '06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 158–169. [Online]. Available: http://dx.doi.org/10.1007/11775300_14

[13] S. Foll, K. Herrmann, and K. Rothermel, "Energy-efficient update protocols for mobile user context," in *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, March 2012, pp. 120–127.

[14] P. Baier, D. Philipp, F. Dürr, and K. Rothermel, "quality-based adaptive positioning for energy-efficient indoor mapping.," Institute for Parallel and Distributed Systems, Universität Stuttgart, Tech. Rep., 2014.

[15] M. B. Kjærgaard, S. Bhattacharya, H. Blunck, and P. Nurmi, "Energy-efficient trajectory tracking for mobile devices," in *Proceedings of the 9th International Conference on Mobile systems, Applications, and Services (MobiSys)*. ACM, 2011, pp. 307–320.

[16] I. Koenig, A. Q. Memon, and K. David, "Energy consumption of the sensors of smartphones," in *Proceedings of the 10th International Symposium on Wireless Communication Systems (ISWCS)*. Ilmenau, Germany: IEEE, August 2013, pp. 1–5.