# CrowdWiFi: Efficient Crowdsensing of Roadside WiFi Networks*

Di Wu[†‡], Qiang Liu[‡], Yuan Zhang[‡], Julie McCann[†], Amelia Regan[‡], Nalini Venkatasubramanian[‡]
[†]Department of Computing, Imperial College London, UK
[‡]Department of Computer Science, University of California, Irvine, US

## ABSTRACT

In this paper, we present `CrowdWiFi`, a novel vehicular middleware to identify and localize roadside WiFi APs that are located outside or inside buildings. Our work is motivated by the recent surge in availability of open WiFi access points (APs) that are enabling opportunistic data services to moving vehicles. Two key elements of `CrowdWiFi` that provide vehicles with opportunistic WiFi access include (a) an online compressive sensing component and (b) an offline crowdsourcing module. Online compressive sensing (CS) techniques are primarily used to for the coarse-grained estimation of nearby APs along the driving route; here, the received signal strength (RSS) values are recorded at runtime, and the number and locations of APs are recovered immediately based on limited RSS readings. The offline crowdsourcing mechanism assigns the online CS tasks to crowd-vehicles and aggregates answers using a bipartite graphical model. This offline crowdsourcing executes at a crowd-server that iteratively infers the reliability of each crowd-vehicle from the aggregated sensing results and refines the estimation of APs using weighted centroid processing. Extensive simulation results and real testbed experiments confirm that `CrowdWiFi` can successfully reduce the number of measurements needed for AP recovery, while maintaining satisfactory counting and localization accuracy. In addition, the impact of `CrowdWiFi` middleware on WiFi handoff and data transmission applications is examined.

## Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

## General Terms

Algorithms, Design, Experimentation

## Keywords

localization, crowdsensing, vehicular networks

## 1. INTRODUCTION

Roadside WiFi networks are increasingly being tapped into by end users with WiFi interfaces in vehicular networks opportunistically for a broad range of applications including ad hoc data dissemination and low-cost Internet access. These networks use fixed access points (APs) that provide improved higher bandwidth connectivity due to better signal propagation characteristics and their ability to exploit spare spectrum. This is especially the case in locations with limited cellular coverage and/or in environments vulnerable to the obstruction of satellite signals by buildings and is typical in both urban environments (with significant built infrastructure) and in rural areas (where cellular connectivity may be sparse). To support smooth continuous Internet operation in the presence of dynamics caused by vehicle mobility, we argue that a middleware that supports accurate real-time identification of roadside APs is critical; something which current mobile devices and vehicular middleware cannot offer. Since APs are deployed in a dynamic and unregulated manner, the efficient in-network lookup of roadside APs is key to mobile vehicles seamlessly finding WiFi connections. The desired service must support multiple functionalities including lookup of APs, identification of APs in near range, their count and location; the design of such as middleware presents unique opportunities and challenges.

The AP lookup feature has multiple benefits in roadside WiFi networks. For example, in conditions where a popular AP is congested, the mobile vehicle can switch to other candidate APs in its communication range. Accurate lookup of roadside APs, deployed outdoors or inside buildings, is also an important step towards understanding the topologies and network characteristics of large scale WiFi networks, *e.g.* network density, connectivity, interference properties, etc, in urban areas. Furthermore, the lookup of APs may reveal interesting social aspects of vehicular networks so that mobile vehicles can be involved in location based services and mobile cloud support.

However, enabling accurate lookup in dynamic settings is however not straightforward.

1) War-driving [5] techniques proposed for AP lookup in mobile scenarios assume relatively low moving speeds.

*This work was partially supported by the Intel Collaborative Research Institute for Sustainable Connected Cities, the University of California Transportation Center, and the National Science Foundation under Grant No. 1063596 and 1059436.


*Middleware'14,* December 08–12 2014, Bordeaux, France.

http://dx.doi.org/10.1145/2663165.2663329.

With fast-moving vehicles, lookup results based on fingerprinting WiFi beacons usually yield rough location estimates with errors in tens of meters - this is often due to the fact that only a small number of beacons can be collected by fast-moving vehicles. Accurate location estimates of roadside APs are critical for the mobile vehicle being able to associate with the best-available AP. We require solutions to improve AP location estimates by an order of magnitude in highspeed vehicular networks, given sparse signal collection opportunity.

2) High mobility in vehicular networks also impacts the connection between mobile vehicles and roadside APs. Ideally, vehicles can obtain a list of APs from the server along its path – such lookup results from existing wardriving databases, *e.g.* Skyhook [15], are simplistic and error-prone since the server side lacks efficient methods to evaluate the accuracy of the information contributed by various mobile users from the same geographic area. Meaningful learning schemes are needed to generate more accurate lookup results by fusing multiple estimates and inferring the overall reliability of the aggregate values.

To address the above challenges, we propose `CrowdWiFi`, a crowdsensing middleware (See Section 3) specifically designed for vehicular networks. It consists of two major components to enable efficient lookup on roadside WiFi networks: an online compressive sensing component and an offline crowdsourcing component. The online compressive sensing component running at the vehicle end coarsely counts and localizes nearby APs in real-time while driving, using sparse signal collection capabilities. The Offline crowdsourcing component running at the server end assigns online compressive sensing tasks to some mobile vehicles, then aggregates the online sensing results uploaded by these vehicles, and produces a fine-grained estimation of AP distribution.

In-network localization algorithms require a large number of RSS (Received Signal Strength) readings; this is impractical with fast-moving vehicles. We exploit the use of compressive sensing (CS [3]) techniques to reduce complexity since they allow the recovery of sparse signals with far fewer noisy measurements than that predicted by the Shannon-Nyquist sampling theorem. Unlike several existing solutions [7, 19], we aim to provide an online CS scheme to recover sparse signals by reading and handling dynamic amounts of noisy measurements at runtime in vehicular networks. Feng et al [7] used CS to localize only *one* mobile target from multiple stable reference nodes, which is vastly simpler than the problem we attempt to solve, which requires looking up *multiple targets* from a single mobile vehicle. Our scheme implements CS using $\ell_1$-minimization [3], which can be solved in polynomial time, to count and localize APs in a sparse network online without a priori knowledge of their number and location. We apply our CS scheme in a situation where RSS values are recorded by an *RSS-collector* online, and the number and location of APs must be recovered immediately based on only a few noisy RSS readings, so that efficient estimations can be made in the presence of network dynamics. Upon receiving the measurements, the recovered information includes the number of nearby APs and their coarse-grained locations on a grid.

Crowdsourcing [10] refers to the outsourcing or sharing of tasks among loosely defined resources, typically workers, *crowd-vehicles*, in our case. `CrowdWiFi` uses geographical participation allowing servers to assign AP lookup tasks to crowd-vehicles to run the online CS component, and gain information from aggregated answers. A major problem of this crowdsoucing scenario is that the answers are often unreliable and diverse, mainly because it is difficult to monitor the performance of a large collection of crowd-vehicles under various communication environments and mobility situations. In the extreme, there may exist "spammers", who submit random rather than good-faith answers. Efficient aggregation methods should take into account the differences in the reliabilities of crowd-vehicles' answers. A common strategy to improve aggregation is to add redundancy. `CrowdWiFi` uses a bipartite graph to assign each task to multiple workers and then aggregate the resulting answers. In addition, we address offline crowdsourcing by transforming the aggregation problem into an iterative inference problem on the graphical model, and obtain the reliability of each crowd-vehicle. The reliability information is used to refine the estimation of APs using weighted centroid processing. The crowdsourced result can be further used for WiFi topology analysis, or downloaded and shared by other vehicles that will move into these road segments and need Internet access, data dissemination or other infrastructural supports.

To the best of our knowledge, `CrowdWiFi` is the first mobile middleware using the concept of crowdsensing to localize roadside APs in vehicular networks. The rest of this paper is organized as follows. Section 2 presents the related work on localization. Section 3 gives a system overview of `CrowdWiFi` middleware. Section 4 describes the online compressive sensing in `CrowdWiFi`. Section 5 addresses the offline crowdsourcing issues in `CrowdWiFi`. Section 6 evaluates `CrowdWiFi` performance using simulations and real testbed experiments. Section 7 concludes our paper.

## 2. RELATED WORK

We describe related work on localization in vehicular networks using RSS-based and crowdsourcing-based approaches.

Several in-network localization approaches use RSS-based collection since it is straightforward to implement in wireless environments and has no extra requirement on hardware, For instance, grid based target lookup algorithms [20] have used a Gaussian mixture model and an expectation-maximization method to derive the number and location of wireless sensors, by enumerating the probabilities associated with each grid point estimate. Multidimensional scaling (MDS) based approaches [9] have been designed to analyze the dissimilarities between pairs of WiFi APs from radio scans, then produce a geometric configuration of WiFi APs. Place Lab [4] presented a ranking scheme to sort RSS collected from wardriving, then apply $k$-nearest neighbor (KNN) based fingerprinting to localize WiFi infrastructure. Techniques have also been proposed to improve the robustness of RSS-based localization algorithms using probabilistic models and calibration enhancements [6, 16]. Compressive sensing based localization has been addressed in [7, 19] for sparse target counting and positioning via $\ell_1$-minimization program.

In recent years, crowdsourcing based solutions for WiFi fingerprint localization have been attracting much attention. Zee [12] is an indoor localization system that makes the calibration zero-effort, by enabling training data to be crowdsourced without any explicit effort on the part of users. FreeLoc [18] can extract accurate indoor fingerprint values

from short RSS measurement times and achieve calibration-free positioning across different devices in crowdsourcing based systems. [17] proposed a centroid method to crowdsource heterogeneous access points for routing switch and data dissemination in hybrid networks. A major problem of crowdsourcing-based localization is that the qualities of the answers are often unreliable and diverse. A common strategy to improve reliability is to add redundancy, such as assigning each task to multiple workers, and aggregate the workers' answers by some method such as majority voting [14]. This problem also can be addressed by building probabilistic models to assign tasks and process answers using standard inference. [8] gave a message-passing style algorithm for deciding which tasks to assign to which workers and for inferring correct answers from the workers' answers.

Note that RSS-based localization approaches have been primarily designed for localizing wireless client nodes, and not the infrastructure. In this paper, we consider the opposite problem – localization of infrastructure nodes (APs) for roadside WiFi lookup. In addition, given sensing capabilities of fast-moving vehicles, we use crowdsourcing to improve the accuracy of the lookup results, where we focus on analysis of crowd-vehicles' reliabilities and effective aggregation models for fine-grained estimation of roadside APs.

## 3. SYSTEM OVERVIEW

In this section, we present the architecture of `CrowdWiFi`, a crowdsensing middleware to look up roadside APs.

`CrowdWiFi` has two key elements – an online compressive sensing component which counts and localizes nearby APs in a coarse-grained way along the driving route; and an offline crowdsourcing component which evaluates the reliability of each crowd-vehicle after aggregating sensing results and then produces fine-grained estimation of the APs' locations.

As shown in Fig. 1, the `CrowdWiFi` middleware is designed to operate as a service on a vehicular/mobile platform to provide intelligent transportation applications executing in a distributed setting with information about APs. The online compressive sensing component executes on the client/vehicle OS, while the offline crowdsourcing component executes on the server side. Client-server interaction protocols enable mobile vehicles to upload the coarse-grained online sensed results to the server, and download the fine-grained offline crowdsourced results from the server. The AP lookup results generated by `CrowdWiFi` middleware are provided through a service interface to many software applications in vehicular networks, *e.g.* WiFi handoff operation, WiFi topology analysis, and other location-based services.

Three crowdsensing parties are actively involved in `CrowdWiFi` system for AP lookup, as shown in Fig. 1, with following specific functions:

- **Crowd-vehicle**: this party plays the role of a worker in the `CrowdWiFi` system. Typical candidates for crowd-vehicles come from public transportation, *e.g.* buses, official vehicles, *e.g.* patrol cars that have regular driving routes and schedules to provide WiFi sensing services in certain geographical areas. Private cars also can sense and upload roadside WiFi information to the crowd-server, possibly in return for a small reward.

- **Crowd-server**: this party assigns AP lookup tasks to crowd-vehicles to run online compressive sensing in some road segments, and then aggregates answers with different reliabilities by offline crowdsourcing. The refined crowdsourced result can be further used for WiFi topology analysis, or downloaded by user-vehicles that will present in these road segments and need roadside WiFi access.

- **User-vehicle**: this party downloads fine-grained AP lookup results from crowd-server in advance, and uses this information for further opportunistic connection with nearby WiFi APs.

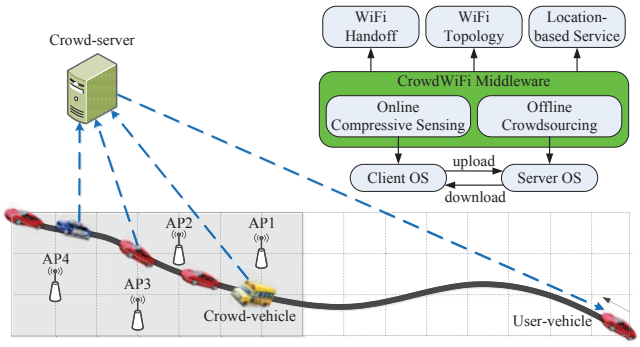



**Figure 1: Crowdsensing of roadside WiFi and CrowdWiFi middleware.**

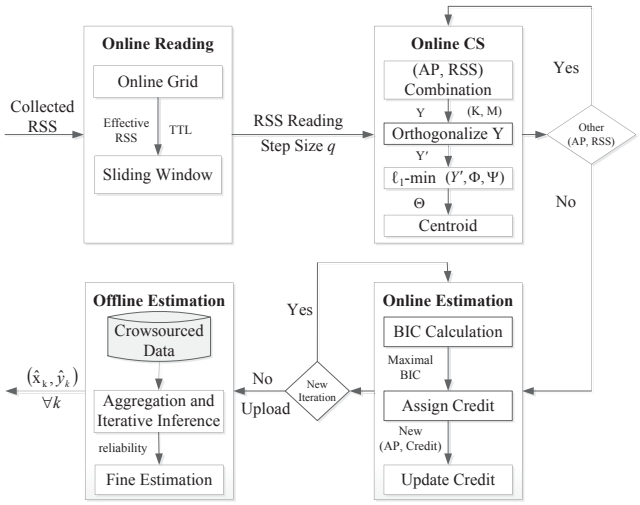



**Figure 2: Workflow (upload case) in CrowdWiFi middleware.**

The workflow of `CrowdWiFi` middleware is illustrated in Fig.2. We use an upload case from the crowd-vehicle to the crowd-server to explain its operational steps. In order to further enhance the processing speed and derive the number and location of APs while the vehicle is moving, we propose an iterative approach for online CS, based on a sliding window and additional steps over the RSS data series collected in a driving grid. Through online CS, nearby APs will be located on grid points with coarse-grained estimation. However, crowd-vehicles that execute the compressive sensing task have different reliabilities, due to various com-

munication environments (*e.g.* signal interference, malicious attack), and processing capabilities (*e.g.* CPU, memory). Besides assigning *lookup tasks* to crowd-vehicles for online CS, the crowd-server also assigns *mapping tasks* to crowd-vehicles in a bipartite graph to aggregate AP lookup results, and analyzes the reliability of each crowd-vehicle using an iterative inference approach. Finally, the reliability information will be used to refine the estimation of APs by using weighted centroid processing. The individual blocks of the system model in Fig.2 will be further explained in detail in the following sections.

# 4. ONLINE COMPRESSIVE SENSING

We formulate roadside AP lookup tasks on crowd-vehicles as a compressive sensing problem and propose corresponding online strategies for efficient estimates in `CrowdWiFi`.

## 4.1 Fundamentals of Compressive Sensing

Recently, research has shown that CS can reconstruct a sparse signal with a much lower sampling rate than Nyquist' Shannon sampling theorem. Let $s$ be a $N \times 1$ column vector. Given an $N \times N$ orthogonal basis $\Psi = [\Psi(1), \Psi(2), \ldots, \Psi(N)]$ where each $\Psi(i)$ being a column vector, $s$ can be expressed by: $s = \Psi\theta = \sum_{i=1}^{N} \theta_i \Psi(i)$, where $\theta$ is the sequence of coefficients needed to represent $s$ in the domain of the basis $\Psi$. Signal $s$ is $k$-sparse if it is a linear combination of $k$ basis vectors. If $k \ll N$, compressive sensing aims to reconstruct $s$ by taking a set of measurements $M$ much smaller than $N$ by finding a minimal solution to: $y = \Phi s = \Phi\Psi\theta = A\theta$, where $y$ is an $M \times 1$ vector, $k < M \ll N$, $\Phi$ is an $M \times N$ measurement matrix, and $A$ is an $M \times N$ matrix.

- $\ell_1$-minimization: For an $N \times 1$ vector $\theta$, its solution is obtained from the following $\ell_1$-minimization [3], which can be solved in polynomial time, to reconstruct the sparse signal: $\hat{\theta} = \arg\min_{\theta \in A^N} \|\theta\|_1$, s.t. $y = A\theta$, where $\|.\|_1$ is the $\ell_1$-norm. If the measurement $y$ include some noise $\varepsilon$, *e.g.* additive white Gaussian noise, then the $\ell_1$-minimization to reconstruct $\theta$ is $\hat{\theta} = \arg\min_{\theta \in A^N} \|\theta\|_1$, s.t. $y = A\theta + \varepsilon$.

$\ell_1$-minimization can be used to recover $\theta$ exactly if $\theta$ is $k$-sparse, or compute an approximation of $\theta$ that is at least as good as if it is computed from the values and locations of the $k$ most significant coefficients of $s$.

## 4.2 Compressive Sensing in AP Lookup

### 4.2.1 Channel Model

The relationship between RSS and the distance between vehicle and AP is usually defined by a log-distance path loss model [13], as: $r = t - l_0 - 10\gamma \log_{10}(\frac{d}{d_0}) - S, d \geq d_0$. $t$ and $r$ are the transmitted and received signal power in dBm, respectively; $d$ is the distance between the transmitter and receiver; $d_0$ is the reference distance; $l_0$ means the path loss in dBm at $d_0$; $\gamma$ refers to the path loss exponent; and $S$ is the log-normal shadow fading in dB.

However, in reality, the RSS-collector receives signals from multiple APs, so each RSS measurement could potentially come from any of the sources in a probabilistic way. To capture such a fact, we collect a series of RSS measurements at a time, denoted as $R = \{r_1, r_2, \ldots, r_n\}$, which are assumed to be mutually independent, and the probability of RSS measurement series $R$ coming from the mixture of $K$ APs can

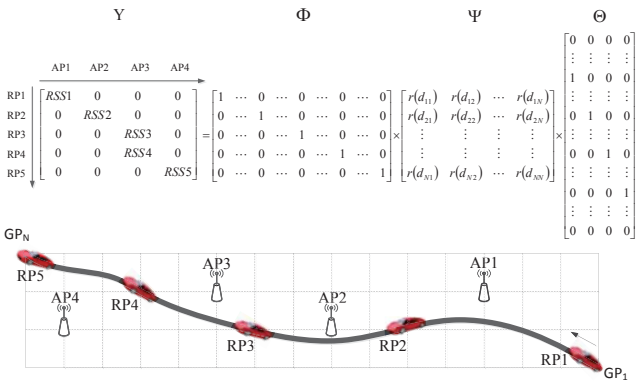



**Figure 3: Compressive sensing in roadside WiFi networks.**

be described using Gaussian mixture model (GMM), as:

$$p(R) = \prod_{i=1}^{n} \sum_{j=1}^{K} \frac{w_{ij}}{\sqrt{2\pi}\sigma_{ij}} \exp\left[-\frac{(r_i - \mu_{ij})^2}{2\sigma_{ij}^2}\right] \quad (1)$$

where $\mu_{ij}$ represents the expected value of the $i$-th RSS measurement, which can be computed by the distance between the $j$-th AP and the RSS-collector at each RSS measurement point $i$, using the log-distance path loss model, as: $\mu_{ij} = t - l_0 - 10\gamma \log_{10} d_{ij}$. $\sigma_{ij}$ is the standard deviation of the Gaussian model. For convenience, we set $\sigma_{ij} = b\mu_{ij}$, where $b$ is a constant.

The weight $w_{ij}$ of each RSS measurement $r_i$ depends on the Cartesian distance $d_{ij}$ between the AP $j$ and the RSS-collector at the $i$-th RSS measurement. It is computed by: $w_{ij} = e^{-d_{ij}} / \sum_{j=1}^{K} e^{-d_{ij}}$, which places more weight in receiving RSS measurements from the closer APs than from the farther ones, thus enforcing a myopic policy to filter the RSS data and make our model reliable in real environment.

Our goal is to find the optimum $K$ AP locations such that the probability $p(R)$ in Eq. (1) is maximized. In order to decide the right number of mixtures, we will use a penalty for introducing too many Gaussian components based on Bayesian information criterion (BIC) [6] in Section 4.3.5.

### 4.2.2 Problem Formulation

Fig. 3 illustrates an example with $K = 4$ APs deployed in a vehicular area divided into a discrete grid with $N = 64$ grid points (GPs), while the number and locations of these APs are unknown to a mobile vehicle. On-board wireless devices take the drive-by RSS measurements from these APs at $M = 5$ arbitrary reference points (RPs) over the grid. The goal is to determine the number and locations of these APs efficiently, using only a small number of noisy RSS measurements. Since $K \ll N$, and the number of measurements $M \ll N$, the APs lookup problem can be well formulated as a sparse matrix recovery problem in the discrete spatial domain using compressive sensing as follows:

$$Y = \Phi\Psi\Theta + \varepsilon, \quad (2)$$

where:

- $\Theta_{N \times K}$ is a $N \times K$ matrix denoting the locations of the APs over the grid, and $\Theta = [\theta_1, \theta_2, \ldots, \theta_K]$. Each $\theta_k$ is an $N \times 1$ vector with all elements equal to zero except

$\theta_k(n) = 1$. $n$ is the index of the grid point at which the $k$th AP is located.

- $\Psi_{N\times N}$ is the sparsity basis, under which the measured signals have sparse coefficients $\Theta$, as defined above. Assume that the transmitted signal power of an AP is $t$ (dBm). The received signal power $r(d_{ij})$ follows the log-distance path loss model. $[\Psi]_{ij} = r(d_{ij})$ records the RSS reading on grid point $i$ from the AP located at grid point $j$, for all $1 \le i \le N$, $1 \le j \le N$.

- $\Phi_{M\times N}$ is the measurement matrix to record the locations of the vehicle to collect the RSS. Vehicle can know its location from widely used GPS or other positioning systems. Only a small number of RSS measurements are collected by the mobile vehicle on several arbitrary grid points, referred as RPs. Thus, each row of $\Phi$ represents the location of each RP, with an element of 1 to indicate the grid point at which the RP is located.

- $Y_{M\times K}$ includes the compressive noisy RSS measurements from $K$ APs and collected by the mobile vehicle on $M$ RPs. Each row vector indicates one measurement value, since the vehicle only can receive one RSS measurement at a time. The number of measurements obeys $M = O(K\log(N/K))$, with $M \ll N$.

- $\varepsilon$ is the measurement noise.

The specific application of above CS operations is illustrated in Fig. 3 by the matrix $Y$, $\Phi$, $\Psi$, and $\Theta$.

Since the sparsity basis $\Psi$ and the measurement matrix $\Phi$ are coherent in spatial domain, we apply an orthogonal operation on matrix $Y$ so that we can use the compressive sensing.

PROPOSITION 1. *Assume compressive noisy measurement matrix $Y$ with size $M \times K$. $Y = \Phi\Psi\Theta + \varepsilon$, where $\Theta$ is a $K$-sparse matrix in a $N$-dimensional space, and $M = O(K\log(N/K))$. Let $Y' = TY$, $A = \Phi\Psi$, and $Q = orth(A^T)^T$, where $T = QA^\dagger$ and $A^\dagger$ is a pseudo-inverse of matrix $A$. Then, $\Theta$ can be well recovered from $Y'$ via an $\ell_1$-minimization program.*

PROOF. See Appendix A. □

## 4.3 Online Counting and Localization

While in theory $\ell_1$-minimization is solvable in polynomial time [3], it becomes computationally expensive when $N$ is large. When a vehicle moves in a vehicular network, it keeps reading the RSSs from different APs, so the noise measurement matrix $Y$ is increasing fast online, as shown in Fig. 3. Meanwhile, the size of $\Phi$ and $\Psi$ will be updated as well in order to get an online estimate of $\Theta$.

In order to further enhance the processing speed and derive the number and location of the wireless APs while the vehicle is moving, we propose an online CS approach in `CrowdWiFi`, using a sliding window and an iteration step over the collected RSS data series. In comparison with traditional CS localization using orthogonal noisy RSS measurements and $\ell_1$-minimization as introduced in Section 4.2.2, the proposed online CS in `CrowdWiFi` applies additional steps to online counting and localization based on limited RSS measurements, and can achieve low cost on CS computation for efficient estimation by several iterations. The workflow of online CS to count and localize APs is illustrated in Fig.2. We explain its key components in details as follows.

### 4.3.1 Grid Formation

Vehicle trajectories typically cross a large geographic area. These long trajectories result in a large number of grid points and large sparsity bases $\Psi$ in the offline CS operation. However, many of these grid points provide irrelevant or redundant information. Therefore online AP lookup from current grid points in reachable region is more practical for vehicle users.

Our online grid formation in `CrowdWiFi` is designed based on the dynamic time-dependent route in each round of online CS. In the $n$-th round of `CrowdWiFi` with inputs $R_n$, we count and localize the APs by a grid structure on current driving area. We derive the boundaries of the driving area according to the input RSS measurements $R_n$ and their corresponding RP location information. Simply, the boundaries of the fixed area is defined by a rectangle with $(x_{\min} - T_m, y_{\min} - T_m)$ and $(x_{\max} + T_m, y_{\max} + T_m)$ as the lower-left and upper-right corners' coordinates, and $x_{\min}$, $y_{\min}$ and $x_{\max}$, $y_{\max}$ are the minimum and maximum $x$ and $y$ coordinates of the series of RPs' locations, respectively. $T_m$ is the communication radius of the RSS-collector in the vehicle.

Given the area definition, we draw a grid structure on the area. Depending on the accuracy and computation cost of the lookup algorithm, the edge length of each lattice in the grid structure can be determined.

### 4.3.2 Sliding Window based RSS Readings

Based on the driving route and its time-dependent grid area, the mobile vehicle will collect a series of RSS measurements from APs. In order to further enhance the processing speed in `CrowdWiFi` and look up APs while the vehicle is moving, we take an iterative approach using a sliding window concept over the collected RSS data series. Each RSS is tagged with a time stamp and TTL (Time to Live). Since old RSS data can not provide valuable information for online lookup, they will be expired and then removed from the data set after the TTL.

As the RSS-collector gathers RSS information, we group the most recently collected RSS data series into a small data set for current estimation of nearby AP number and location. Suppose the length of the current collected RSS sequence is $k$. We use a sliding window with a length of $s(s < k)$ to extract the input data sequence from current collected RSS sequence. The iteration step size is set to $q(q < s < k)$. Then, the set of the input RSS sequence in the $n$-th round of iteration is $R_n = \{r_{q(n-1)+1}, r_{q(n-1)+2}, \ldots, r_{q(n-1)+s}\}$. In each round of iteration, `CrowdWiFi` searches the likely number and locations of APs on a grid structure, which maximizes the probability of the data series $R_n$ by a way introduced in Section 4.3.5. After several iterations of selecting $q$ RSS values from data set $R_n$, `CrowdWiFi` consolidates these estimates as in Section 4.3.6, and then refine the APs' number and locations through offline crowdsourcing introduced in Section 5.4.

The sliding window based RSS readings can reduce the number of measurements for AP lookup. It makes online CS operation in `CrowdWiFi` fast and low-complexity, while achieving accurate lookup.

### 4.3.3 Combination of AP and RSS Data

In the AP lookup application, we assume there is no information to indicate how many APs are present, and which RSS reading comes from which AP. In each iteration of sliding window based RSS reading, we put an extra step to make a classification of collected RSS data into different APs, and then apply it to the CS operation before obtaining good lookup results.

`CrowdWiFi` first makes an estimation of the AP number of $K$. Given a small size of RSS readings collected from different RPs on the continuous vehicle trajectory, the upper bound of the AP number should be same as the number of collected RSS by $M$. Suppose we have $M = 5$ RSS values, we can classify these RSS values under different estimates of the AP number $K$, where $K = 1, 2, \ldots, 5$. The matrix $Y \in \mathbb{R}^{M \times K}$ in Fig. 3 shows an example on the classification and combination of five RSS values when `CrowdWiFi` assumes there are five APs.

PROPOSITION 2. *The problem formulation cannot state the number of APs, and which RSS reading comes from which AP. If there are $K$ APs and $M$ RSS measurements, for all the possibilities of AP number $K, (K = 1, 2, \ldots, M)$, the complexity of enumerating combinations of $(K, M)$ is $\Omega(M^M)$.*

PROOF. See Appendix B. □

From Proposition 2, we know that the number of combinations exponentially increases with the number of RSS readings. If we estimate APs based on too much RSS data, it will induce a large cost to run $\ell$-1 minimization in the CS, which is not practical for online counting and localization. Therefore, it is necessary to take the sliding window based RSS reading to reduce the number of (RSS, AP) combinations for online computation. Given the limited number of RSS, the capability of CS to recover sparse signal still can give good coarse-grained estimate of the APs.

#### 4.3.4 Centroid Processing

For each $\hat{\theta_k}$, the output of the $\ell_1$-minimization for CS operation turns out to be a $N \times 1$ vector with all elements equal to zero except one element equal to one, which exactly indicates the grid point at which the AP is located. This means that if the APs are exactly located at the grid points, then the recovery can be precise.

However, the recovered location $\hat{\theta_k}$ does not turn out to be an exact 1-sparse vector, but has a few non-zero coefficients. In order to compensate for the error induced by the grid assumption, a centroid processing procedure is conducted. We choose the dominant coefficients in $\hat{\theta_k}$ whose values are above a certain threshold $\zeta$, and take the centroid of these grid points as the location indicator. Let $\mathcal{S}_k$ be the set of all indexes of the elements of $\hat{\theta_k}$ such that: $\mathcal{S}_k = \{n | \hat{\theta_k}(n) > \zeta\}$.

These are potential candidate points for the estimate of the location of the $k$th source. Each $n \in \mathcal{S}$ represents a point in the two dimensional space $(x_n, y_n)$. The location of source $k$ can be estimated by finding the centroid of the candidate points using weighted mean in (3). The weight for each $(x_k, y_k)$ is its corresponding value $\hat{\theta_k}$ in that grid point. We have,

$$(\hat{x_k}, \hat{y_k}) = \frac{1}{\sum_{k \in \mathcal{S}_k} \hat{\theta_k}} \sum_{k \in \mathcal{S}_k} \hat{\theta_k}(x_k, y_k). \quad (3)$$

#### 4.3.5 Bayesian Information Criterion

In maximum likelihood estimation, the more Gaussian components are estimated, the more accurate the maximum likelihood of the GMM (see Section 4.2.1). In order to decide the right number of mixtures, we introduce a penalty for introducing too many Gaussian components. The Bayesian information criterion (BIC) is commonly used to select model parameters [6].

As shown in Fig. 3, we enumerate the (AP, RSS) combinations from the sliding window based RSS readings, and estimate the number and locations of APs in $\Theta$ after the CS operation. Based on these estimates, we use BIC to pick the best estimate in the $n$-th round of iteration, which provides the maximum likelihood $\log p(R_n)$ and lead to BIC estimate under current AP number $K$.

Given $v$ as the number of free parameters to be estimated and $R$ as the data, we define BIC as: $\text{BIC} = 2 \max \log p(R|v) - v \log(m)$, where $\max p(R|v)$ is the maximum likelihood of the data $R$ given the number of parameters $v$, and $m$ is the number of data samples in the $n$-th round of iteration. In our models, the parameters to be estimated are the two-dimension coordinates of APs, thus $v = 2K$. To use BIC for model selection, we simply choose the model that leads to the maximum BIC. After certain number of combinations on (AP, RSS), `CrowdWiFi` is able to find the best number of APs that maximizes the registered BIC value, as well as the locations of the APs.

#### 4.3.6 Credit based Consolidation

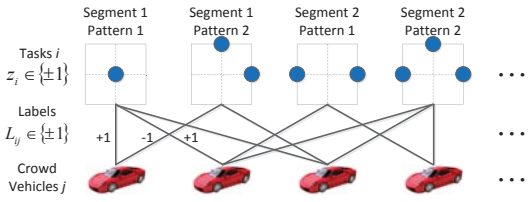
After computation of the maximum BIC value from each iteration of sliding window based RSS reading, we grant one credit to each of the estimated locations satisfying the maximum BIC. Because we maintain a dataset to record previous estimates and credits, we need to update the credit data once we have new assignment of credit.

`CrowdWiFi` takes a data cleansing step that consolidates the location estimates of the prior iterations. We compare the location estimates of current iteration with those of the previous iterations. If any location in the current estimates aligns with a previous location estimate, these two locations are merged, and the merged location gains the credit points of both new and prior location estimates. In addition, the coordinate of the merged location estimate is averaged by taking the centroid of the coordinates of the new and old points, proportional to their credits. If a location in the current estimate does not align with any prior location estimates, the location is added as a new AP location in the AP set.

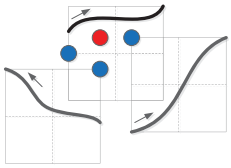
Finally, after a few iterations of online RSS reading and BIC computations, or when RSS data collection is complete, `CrowdWiFi` filters out the spurious location estimates that have few credits. The number of credits for filtering spurious estimates is cross-referenced by reality check, and is normally set at 1. That is, if a location estimate has only one credit, it is removed from the final AP set.

## 5. OFFLINE CROWDSOURCING

After online compressive sensing of roadside WiFi information, crowd-vehicle uploads the geographical distribution of nearby APs, with the number and location information on the corresponding grid formation, to the crowd-server for offline crowdsourcing. The offline crowdsourcing in `CrowdWiFi`




(a) Bipartite graph



(b) Fine-grained estimation

**Figure 4: Aggregation in offline crowdsourcing.**

assigns the AP mapping tasks to crowd-vehicles by a bipartite graph, and aggregates answers using iterative inference on the graphical model. `CrowdWiFi` also refines the locations of APs using crowdsourced coarse-grained locations of these APs on the grid points.

## 5.1 Spammer-hammer Model

In `CrowdWiFi`, each crowd-vehicle is assigned with multiple lookup tasks to sense APs along different road segments. We assume that all the tasks have the same level of difficulty, but that crowd-vehicles may have different reliabilities. Some crowd-vehicles can provide better AP estimates than others due to various communication environments and processing capabilities, while some other crowd-vehicles might be spammers.

We assume the reliability of crowd-vehicle $j$ is measured by a single parameter $q_j$, which corresponds to its probability of correctness. The values of $q_j$ reflect the diversity in crowd-vehicles' reliability, which are independent distributed random variables with a given distribution on [0, 1]. One typical example is *spammer-hammer* model where, $q_j \approx 1$ correspond to *hammers* that provide reliable answers, and $q_j \approx 1/2$ denote *spammers* that give random answers. We assume the $q_j$ of all crowd-vehicles are drawn independently from a common prior $p(q_j|\lambda)$, where $\lambda$ are the hyper-parameters of a prior distribution [10]. To avoid the cases when spammers overwhelm the system, it is reasonable to require that $\mathbb{E}[q_j|\lambda] > 1/2$. Typical priors in *spammer-hammer* model includes the discrete prior, where $q_j \approx 0.5$ or $q_j \approx 1$ with equal probability.

## 5.2 Graphical Model

To aggregate results from unreliable crowd-vehicles and determine their reliability $q_j$, we have designed a bipartite graph scheme for crowd-server to enumerate and assign mapping tasks to crowd-vehicles. As shown in Fig. 4 (a), the task assignment scheme can be represented by a bipartite graph where an edge $(i, j)$ denotes that the mapping task $i$ is labeled by the crowd-vehicle $j$. Each mapping task is a possible distribution pattern (combination of number and location) of APs, denoted by blue dots in a grid formation, given a road segment ID. Crowd-vehicles submit their lookup results to answer if these distribution patterns exist (labeling +1) or not (labeling -1). Initially, some AP distribution patterns are generated randomly by crowd-server for bootstrapping purpose. After that, more distribution patterns can be added into the mapping set by selecting the lookup results from crowd-vehicles, so that the crowd-server can avoid generating too many non-existent AP distribution patterns to save computation and assignment cost.

Specifically, assume there are $M$ crowd-vehicles and $N$ mapping tasks with binary labels $\pm 1$. The true label of task $i$ is denoted by $z_i \in \pm 1$, $i \in [N]$, where $[N]$ represents the set of first $N$ integers. $\mathcal{N}_j$ is the set of tasks labeled by crowd-vehicle $j$, and $\mathcal{M}_i$ is the set of the crowd-vehicles labeling task $i$. The labeling results form a matrix $L \in 0, \pm 1^{N \times M}$, where $L_{ij} \in \pm 1$ denotes the answer if crowd-vehicle $j$ labels task $i$, and $L_{ij} = 0$ if otherwise. Therefore, the reliability $q_j$ of crowd-vehicle $j$ is defined by the probability of correctness: $q_j = \text{prob}[L_{ij} = z_i]$. The goal is to find an optimal estimator $\hat{z}$ of the true labels $z$ given the observation $L$, minimizing the average bit-wise error rate $\frac{1}{N}\sum_{i\in[N]} \text{prob}[\hat{z}_i \neq z_i]$.

## 5.3 Iterative Inference

A naive approach to aggregate crowd-vehicles' labels $L$ is to use majority voting. Majority voting simply chooses what the majority of crowd-vehicles agree on. When there are many spammers, majority voting is error-prone since it weights all the crowd-vehicles equally. We use an iterative inference approach for `CrowdWiFi`, based on a message-passing algorithm proposed by [8]. Let $x_{i\to j}$ and $y_{j\to i}$ be real-valued messages from tasks to crowd-vehicles and from crowd-vehicles to tasks, respectively. Initializing $y^0_{j\to i}$ randomly from Normal(1, 1) or deterministically by $y^0_{j\to i} = 1$, iterative inference updates the messages at $t$-th iteration via

$$x^{t+1}_{i\to j} = \sum_{j'\in\mathcal{M}_{i\setminus j}} L_{ij'} y^t_{j'\to i}, \quad y^{t+1}_{j\to i} = \sum_{i'\in\mathcal{N}_{j\setminus i}} L_{i'j} x^{t+1}_{i'\to j}. \quad (4)$$

A crowd-vehicle message $y_{j\to i}$ represents the updated reliability of the crowd-vehicle $j$, and the labels are estimated via the sum of the answers weighted by each crowd-vehicle's reliability as: $\hat{z}^t_i = \text{sign}[\hat{x}^t_i]$, where $\hat{x}^t_i = \sum_{j\in\mathcal{M}_i} L_{ij} y^t_{j\to i}$. Note that the 0th iteration of iterative inference reduces to majority voting when initialized with $y^0_{j\to i} = 1$.

After transforming the labeling aggregation problem into an iterative inference problem on a bipartite graphical model, the joint posterior distribution of crowd-vehicles' reliabilities $q = q_j, j \in [M]$ and the true labels $z = z_i, i \in [N]$ conditional on the observed labels $L$ and hyper-parameter $\lambda$ is

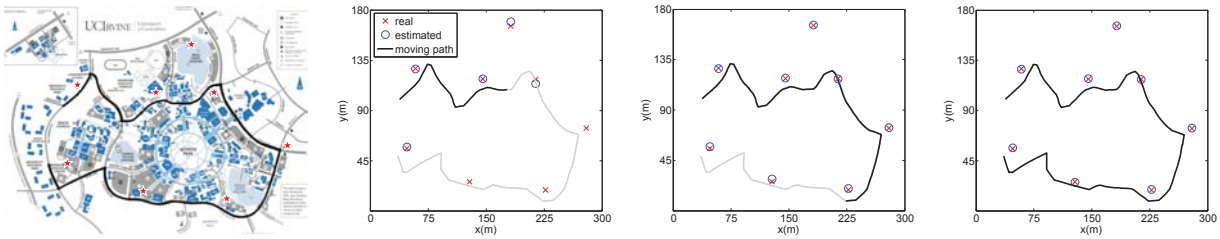
$$\begin{aligned} p(z, q|L, \lambda) &\propto \textstyle\prod_{j\in[M]} p(q_j|\lambda) \prod_{i\in\mathcal{N}_j} p(L_{ij}|z_i, q_j) \\ &= \textstyle\prod_{j\in[M]} p(q_j|\lambda) q_j^{c_j} (1-q_j)^{\gamma_j - c_j}, \end{aligned} \quad (5)$$

where $\gamma_j = |N_j|$ is the number of answers made by crowd-vehicle $j$, and $c_j := \sum_{i\in\mathcal{N}_j} \mathbb{I}[L_{ij} = z_i]$ is the number of $j$' s answers that are correct. By standard Bayesian arguments, one can show that the optimal estimator of $z$ to minimize the bit-wise error rate is given by

$$\hat{z}_i = \arg\max_{z_i} p(z_i|L, \lambda), \quad (6)$$

where $p(z_i|L, \lambda) = \sum_{z_{[N]\setminus i}} \int_q p(z, q|L, \lambda) \text{d}q$.

## 5.4 Fine-grained Estimation




(a) Simulation map (b) 60 data points (c) 120 data points (d) 180 data points

**Figure 5: Trajectory illustrations upon APs lookup in the UCI simulation scenario.**

Online compressive sensing finds the coarse-grained locations of the APs with coordinates all placed on the grid points. From iterative inference in offline crowdsourcing, we can obtain the optimal estimation of $z$ and the corresponding reliability of each crowd-vehicle. Based on the optimal results, we can further refine the locations of APs by edging into the true locations, using operations on crowdsourced grids weighted by the reliability of each crowd-vehicle.

According to the grid formation in Section 4.3.1, because of different moving paths, operation times and sensing locations, crowd-vehicles may form various grid structures from local views during online compressive sensing, even though they are in the same sensing area. As shown in Fig. 4 (b), the estimated locations of the same AP, denoted by three blue dots, are located on different grid points of three grid formations, due to three crowd-vehicles moving on the different paths. Since the three estimated AP locations are close in these overlapped driving grids, a centroid processing procedure is conducted by the crowd-server to merge the three estimates on different grid points to be one estimate on the true location, denoted by the red dot in Fig. 4 (b). Crowd-vehicles with higher reliability can present more accurate estimations, therefore the centroid operation of crowdsourced estimates is weighted by the reliability of each crowd-vehicle to compensate for the lookup error during online compressive sensing. The weighted centroid processing using a crowd-vehicle's reliability is similar to the centroid method illustrated by Eq. 3 in Section 4.3.4.

## 5.5 Crowdsourcing Platform

The crowdsourcing platform in `CrowdWiFi` consists of four components: (1) a web interface where vehicle can upload and download nearby APs information based on its location; (2) a crowd-server including a database for storing the crowdsourced AP information and for distributing the information to potential users; (3) a client application for the crowd-vehicle, which pulls available AP lookup tasks from the database, based on its given driving route and geographical information; and (4) a client application for user-vehicles to obtain AP information. Through this crowdsourcing platform, `CrowdWiFi` is able to virtually provide nearby AP information to vehicles requesting wireless access and data dissemination solutions at any time and anywhere.

The driving routes and locations of crowd-vehicles, their AP lookup results, and other relevant crowdsourcing factors are privacy information. Therefore, in `CrowdWiFi`, crowd-vehicles have the right to accept tasks to share these information for rewards, or deny the tasks to protect their privacy.

# 6. PERFORMANCE EVALUATIONS

We implemented `CrowdWiFi` and evaluated its performance on lookup accuracy by simulation and real testbed experiments. We also evaluated it as a vehicular middleware on handoff performance by some real trace studies. In our grid setting, some RPs located around grid points are selected to estimate the number and locations of APs, according to the formulation of online CS problem in Section 4.2.2.

We first formally define the localization error. For each grid $i$, let $k_i$ and $\hat{k_i}$ be the actual and estimated numbers of roadside APs, respectively. Assume there are $k$ actual APs with locations $\{(x_i, y_i)\}, \forall i = 1, 2, \ldots, k$, and there are corresponding $\{\hat{k_i}\}$ estimated APs with locations $\{(\hat{x_i}, \hat{y_i})\}$. Furthermore, let $k_{\min} = \min\{k, \hat{k}\}$, and thus we derive the localization error as the normalized relative distance: $(\sum_{i=1}^{k_{\min}} \sqrt{(x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2})/(k_{\min} l)$, where $l$ is the length of grid lattice. If the error is less than 100%, then it indicates that the estimated location is close to the real location as their distance is shorter than the grid diameter. Accordingly, we define the counting error in our evaluation, as: $(\sum_{i=1}^{N} |k_i - \hat{k_i}|)/(\sum_{i=1}^{N} k_i)$.

## 6.1 Simulation Results

The campus map of the University of California, Irvine (UCI) and the AP deployments on the map have been used here for AP lookup. There were 8 APs deployed at UCI, and an RSS-collector collected the RSS values of periodic radio packets from these APs along a path as shown in Fig. 5(a). We used NCTUns v5.0 [1] to model the vehicular network and scaled the UCI campus map over a $300m \times 180m$ rectangular area in our simulations. The distance between each pair of APs is more than $50m$, and the effective signal transmission radius of the APs is $100m$. The path loss at reference distance $1m$ is 45.6dBm and the path loss exponent is 1.76. The standard deviation of the shadow fading is set to 0.5dB. We run three sets of simulations based on Fig. 5 as follows to test online CS and offline crowdsourcing.

In order to demonstrate the AP lookup performance using online CS, we first set the 8 APs physically located on 8 grid points and then run online CS to estimate locations at three different moments, namely, those when the RSS-collector collected 60-th, 120-th, and 180-th RSS values in the simulation, respectively. The resulting location estimates are shown in Fig. 5(b), (c), and (d), respectively, where the crosses represent the actual AP locations, and circles are lo-

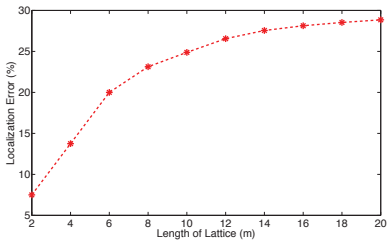
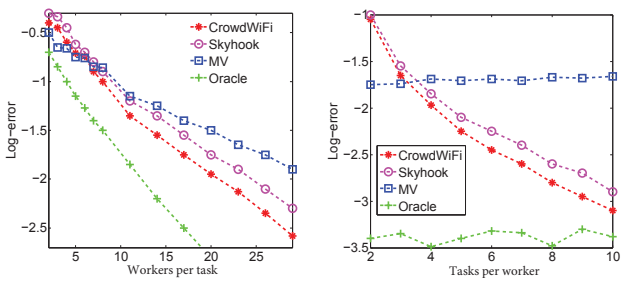
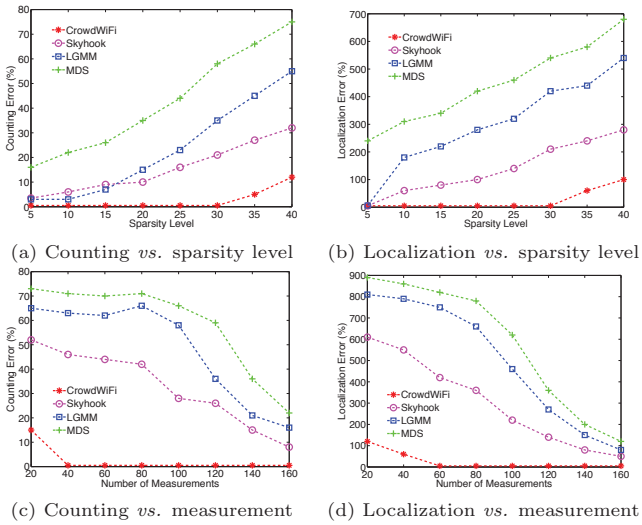



**Figure 6: Impact of lattice size on localization error.**




(a) $\ell$(fixed $\gamma = 5$) (b) $\gamma$(fixed $\ell = 15$)

**Figure 7: Performance of crowdsourcing in bipartite assignment.**




(a) Counting *vs.* sparsity level (b) Localization *vs.* sparsity level

(c) Counting *vs.* measurement (d) Localization *vs.* measurement

**Figure 8: Comparisons between CrowdWiFi and other algorithms on counting and localization errors.**

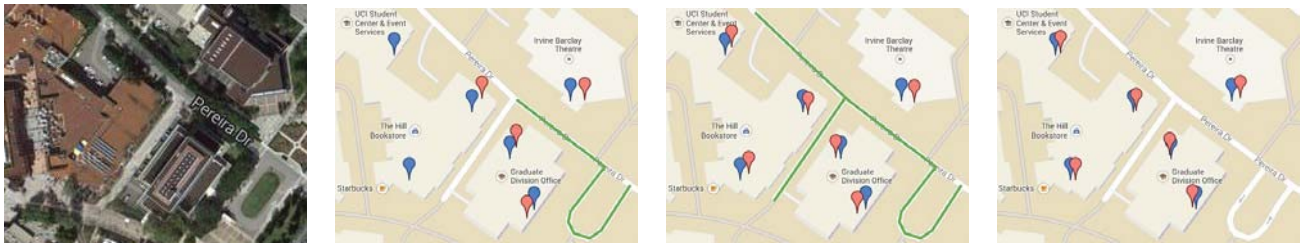
cation estimates by CrowdWiFi. In the proposed CrowdWiFi algorithm, the sliding window size for each computation iteration is set as 60, and the iteration step size is 10. That is, we re-run CrowdWiFi using the past 60 data samples when RSS-collector collects additional 10 RSS values. In order to test the robustness of our algorithm, we intentionally add Gaussian white noise $\mathcal{N}(0, \sigma^2)$ to the observation vector $y$ when running CrowdWiFi. We use SNR to quantify the signal to noise ratio and set SNR=30dB. The lattice size in grid structure is set as $8m \times 8m$.

As Fig. 5(b) to Fig. 5(d) show, when vehicles move around the trajectory, the number of RSS readings are increased, and CrowdWiFi can filter out the spurious AP locations and provide accurate estimations of the number and locations of APs in the Credit-based consolidation. Furthermore, Fig. 5(c) confirms that online CS successfully finds both the number and locations of APs when 120 data points collected. Then, Fig. 5(d) shows that after collecting all 180 data points, the CrowdWiFi algorithm accurately provides 8 estimated APs, successfully matching with the exact AP number and locations. Based on these three location estimates, the average estimation error reduces from 2.6157 meters (for the 60 points) to 1.8316 meters (for the 180 points), demonstrating that the presented online CS can correctly find APs within a satisfactory error bound.

The lattice size (or the number of grid points) in grid structure is an important parameter to run CS, and can determine the accuracy level of AP lookup that CrowdWiFi can achieve. Fig. 6 examines this impact of lattice size on localization error via CrowdWiFi, based on the simulation scenario of UCI map and 180 data points. The localization error is computed by above mentioned method times 100%. Fig. 6 shows that a smaller length of lattice (larger number of grid points) results in more accurate location estimation. When the length of lattice is less than or equal to 10 meters, CrowdWiFi can achieve a low estimation error less than 2 meters. When the length of lattice is around 20 meters, CrowdWiFi still can maintain the error at a certain level below 3 meters. Usually, the estimation error increases along with increasing the lattice length, and a very small length of lattice will result in a more expensive computation cost of the algorithm. According to the results, we learn that choosing a feasible lattice length between 5 to 10 meters can guarantee satisfactory localization performance. We also evaluated the impact of lattice size on counting error, and CrowdWiFi shows 0 counting errors for lattice lengths in the range between 2 meters to 20 meters.

Since online CS only can provide estimates of APs located on grid points, but most APs in reality are not physically exactly located on grid points, we set the true locations of 8 APs randomly on grid structure in the second set of simulations. Then we verified the performance of offline crowdsourcing on fine-grained estimation.

We generated the reliabilities $q_j$ from the spammer-hammer priors, that equals 0.5 or 1.0 with certain probabilities. The assignment graphs were randomly drawn from the set of $(\ell, \gamma)$-regular bipartite graphs with 1000 tasks. The left degree $\ell$ denotes the number of crowd-vehicles per task, and the right degree $\gamma$ denotes the number of tasks per crowd-vehicle. For comparison, we also calculated the majority voting (MV) that chooses what the majority of crowd-vehicles agree on [14], the Skyhook that compares relative rankings using the Spearman rank-order correlation coefficient [4], and the oracle lower bound (Oracle) that assumes the true $q_j$ are known. We terminate all the iterative algorithms at a maximum of 100 iterations or with $10^{-6}$ message convergence tolerance. All results are averaged on 100 random trials. As shown in Fig. 7 (a) and (b) in our experiments, the bit-wise error rate of the iterative inference in CrowdWiFi is shown to be lower than majority voting and Skyhook due to its reliability evaluation, and CrowdWiFi scales in the same manner as an oracle lower bound. Also, we show that the error rates of the crowdsourcing algorithms generally decay




(a) Testbed map (b) 20 data points, 45mph (c) 40 data points, 45mph (estimation error 3.6016m) (d) Crowdsourced result (estimation error 2.2509m)

**Figure 9: Roadside APs lookup and crowdsourcing in UCI testbed experiments.**

exponentially w.r.t. the degree $\ell$ and $\gamma$ of the assignment graph on a spammer-hammer model.

In the third set of simulations, we have tested the performance of offline crowdsourcing combined with online CS on AP lookup. To verify the strength of AP lookup in `CrowdWiFi`, we compared it with several state-of-art algorithms specifically developed for counting and locating WiFi APs. We implemented the Gaussian mixture model based grid algorithm [20], and multidimensional scaling (MDS) based radio scan algorithm [9], referred as "LGMM" and "MDS", respectively. We also implemented Skyhook [15] based on Place Lab's fingerprinting algorithm [4] (Skyhook's algorithm is proprietary, but similar to Place Lab [5]). The localization error and counting error are computed by above mentioned method times 100%, respectively. The simulation area is $250m \times 250m$, and the lattice size in grid is set as $8m \times 8m$.

Fig. 8(a) and (b) depict the counting and localization error *vs.* the sparsity level $k$ when $N = 900$, $M = 160$, where $k$, $N$ and $M$ indicate the number of WiFi APs, the number of grid points and the number of reference points, respectively. We also set SNR = 30dB. It can be observed that `CrowdWiFi` and Skyhook brings much lower error than other algorithms due to the adoption of crowdsourcing. In addition, because `CrowdWiFi` employs compressive sensing for sparse signal processing and reliability based crowdsourcing evaluation, it can achieve better performance on AP lookup than Skyhook. When $k = 30$, both counting and localization errors are almost zero, while other algorithms produce a counting error of at least 21% and a localization error of more than 200% under the same scenario. Even when $k$ is as small as 10, the localization error of LGMM, MDS, and Skyhook is still above 60%, though their counting errors are relatively small (0.03 or higher). When $k = 40$, `CrowdWiFi` achieves a counting accuracy of around 10% and a localization error of less than 100% (inside a grid) while other algorithms result in a much lower accuracy.

Fig. 8 (c) and (d) depict the counting and localization error *vs.* the number of measurements $M$ when $N = 900, k = 10$ and SNR = 30dB. As an overall trend, the larger the $M$ is, the smaller the error for all algorithms. Note that both the counting error and localization error of `CrowdWiFi` are almost zero when $M \geq 40$, while other algorithms yield much higher errors when $M < 100$. This clearly indicates that `CrowdWiFi` does not need a large number of measurements to precisely estimate the number and location of the APs, and thus has considerable values in practical scenarios.

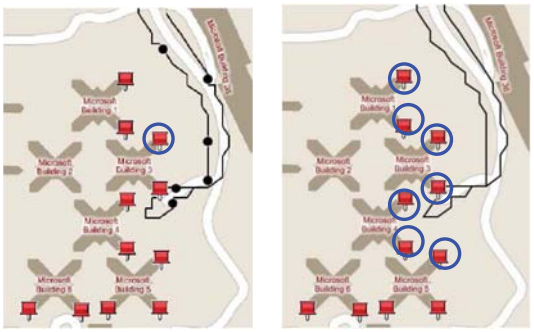
## 6.2 Testbed Experiment Results

In our real testbed experiments, we used Open-Mesh wireless mesh nodes that run the IEEE 802.11b/g standards as APs [11]. Each node is an integrated access point, mesh gateway and repeater, all in one tiny reliable package. The node model is OM1P and its price is as cheap as 30 US dollars. Our `CrowdWiFi` system and results of these experiments can similarly apply to other kinds of roadside APs.

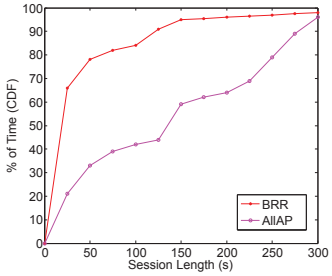
We deployed six Open-Mesh nodes at six different locations: two in Graduate Division Office, one in Irvine Barclay Theatre, one in The Hill Bookstore, one in Starbucks and one in UCI Student Center, over a $100 \times 100$ square meters area on UCI campus as shown in Fig. 9. The blue icons indicate the real location, the red icons indicate the estimated location, and the green solid line indicates the moving path of our vehicle. The lattice size in grid structure is set as $10m \times 10m$. As the RSS-collector, a ThinkPad X61 Laptop with Intel(R) PRO/Wireless 3945ABG Network Connection was used to collect the RSS values along the moving path of our vehicle. The transmission radius of the Open-Mesh nodes is approximately 30 meters.

For crowdsourcing of APs lookup results, the vehicle collected RSS values from nearby APs at three different average moving speeds around $20mph$, $35mph$, and $45mph$. We present the location estimation results for each moving speed at two different moments of the experiment, namely those when the RSS-collector collected 20-th and 40-th RSS samples, as shown in Fig. 9(b) and Fig. 9(c) for the case with speed $45mph$. The offline crowdsourcing platform aggregated the AP lookup results of the three moving speeds, and obtained corresponding reliability of the crowd-vehicle. After the centroid processing of crowdsourced AP information weighted by crowd-vehicle's reliability, the crowdsourced result shows more accurate estimation than individual vehicle. `CrowdWiFi` can look up all six Open-Mesh nodes to match with actual locations, and the average estimation error in the real testbed experiments is $2.2509m$, as shown in Fig. 9(d).

We also downloaded the Skyhook software, which is an existing system built upon fingerprinting and crowdsourcing, from the Internet to compare it with `CrowdWiFi`. Skyhook relies on war-driving to create current radio map, and then updates lookup results according to its previous records within university campuses and other driving paths. We tested Skyhook in the same area and the estimation error is $11.6028m$, which is less accurate than `CrowdWiFi`, due to disadvantages both in its localization scheme and crowdsourcing evaluation.

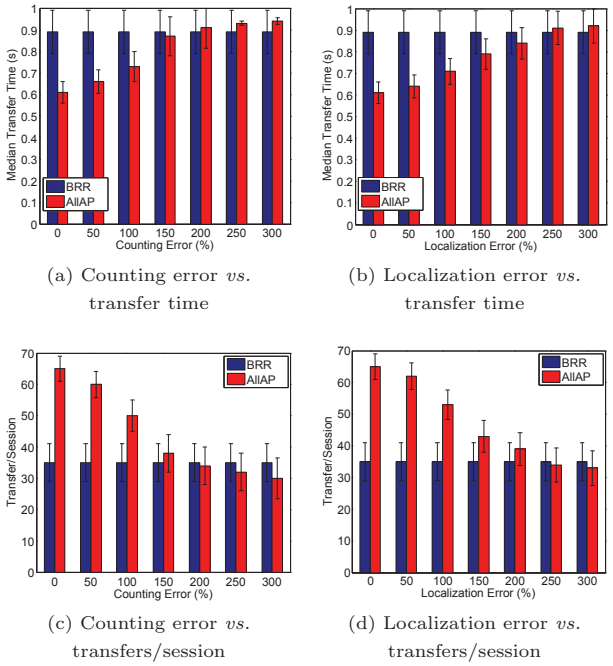


(a) BRR (b) AllAP




(c) Distribution of session length

**Figure 10: APs lookup in VanLan trace studies.**




(a) Counting error *vs.* transfer time

(b) Localization error *vs.* transfer time

(c) Counting error *vs.* transfers/session

(d) Localization error *vs.* transfers/session

**Figure 11: Impacts of AP lookup accuracies on the network connectivity.**

## 6.3 Network Handoff and Connectivity

We use VanLan [2] traces, consisting of 11 APs and 2 vans performing as crowd-vehicles. User-vehicles use the crowdsensed lookup results to connect nearby APs. APs are deployed across five buildings in the Microsoft campus, as illustrated in Fig. 10, covering a $828m \times 559m$ area. All vehicles travel with a speed limit of $25mph$. Each crowd-vehicle visits the region with AP deployment about ten times a day. All APs and vehicles are equipped with Atheros 5213 chipset radios. Their output power is around 26.02 dBm. All vehicles are equipped with a GPS unit that outputs location information once every second. During the evaluation, each AP and vehicle broadcasts a 500-byte packet at 1 Mbps every 100 $ms$. For the VanLan dataset in our experiment, there are 12544 RSS data associated with the beacon messages collected by the moving crowd-vehicle from nearby APs. Since compressive sensing can recover sparse signals from a small number of measurements, we chose 300 RSS data from the dataset to evaluate the lookup performance of `CrowdWiFi`.

We also propose two handoff policies to evaluate the impacts of `CrowdWiFi` on the network connectivity,

- **BRR**, where the vehicle connects to the AP with the highest exponentially averaged beacon reception ratio. It is a hard handoff policy, since the vehicle can only communicate with the associated AP.
- **AllAP**, where the vehicle opportunistically uses all APs in the vicinity. A transmission by the vehicle is considered successful if at least one AP receives the packet. The probabilities that a vehicle associates with nearby APs are related to their locations.

Moving vehicles are often within the communication range of multiple APs, and packet loss events are usually bursty and generally independent across senders and receivers. Since in the `CrowdWiFi` framework, user-vehicle can download the crowdsensed AP lookup results from the crowd-server in advance, AllAP can fully exploit multi-user diversity between the user-vehicle and the set of nearby APs. Therefore, AllAP is significantly more effective than BRR scheme that uses only one AP even when that AP is judiciously chosen.

Fig. 10 shows the behavior of BRR and AllAP during an example path segment in VanLan. Black lines represent regions of adequate connectivity, *i.e.*, more than 50% reception ratio in a one-second interval. Dark dots represent interruptions in connectivity. Blue circles represent the associated APs in two schemes. As shown in Fig. 10(a), BRR scheme contains several regions of inadequate connectivities due to the hard handoff to only one AP. In contrast, AllAP scheme in Fig. 10(b) performs far better since it uses multiple APs to further reduce the number of interruptions. It makes full use of all nearby APs, and the average localization error is just $2.0658m$. This effect becomes clearer in Fig. 10(c) that compares the two handoff policies with regard to cumulative time users spend in an uninterrupted session of a given length. Cumulative distrusting probability (CDF) of time is used to describe the probability that the real uninterrupted length is less than or equal to the given length. Based on it, we can correspondingly derive a probability that the real uninterrupted length is more than the given length. For median session length given in this scenario, we see the probability of AllAP is seven times more than that of BRR, suggesting its superiority over hard handoff on wireless connectivity.

We next conduct an experiment transferring a 10 KB file over TCP among user-vehicles and APs, to evaluate the network connectivity in terms of median transfer time and throughput, under various counting and localization errors. Transfers that make no progress for 10s are terminated and re-started afresh.

Fig. 11 (a) and (b) show the median time to complete a transfer under various counting and localization errors. The results show that given the accurate AP lookup results, the median TCP transfer time of AllAP scheme is about 0.61s, i.e., 50% improvement over BRR. Even under certain counting and localization errors, AllAP still outperforms BRR, due to the multi-user diversity gain achieved from nearby APs. Fig. 11 (c) and (d) show the throughput, or average number of completed transfers per session, under various counting and localization errors. It is observed that AllAP achieves almost twice the throughput of BRR if there are no counting and localization errors. However, the superiority does not diminish even if the system experiences tolerable estimation errors. Therefore, given the typically shorter transfer periods, users of the AllAP scheme will experience fewer disrupted transfers and better performance for individual transfers.

# 7. CONCLUSION

`CrowdWiFi` is vehicular middleware for roadside AP lookup using online CS and offline crowdsourcing techniques. The CS-based coarse-grained lookup approach includes completed online steps to make the CS operation efficient and effective for crowd-vehicles to recover sparse APs along the driving route. Online CS also reduces the number of RSS readings needed in `CrowdWiFi`, while maintaining good estimation results. The crowdsourcing-based fine-grained lookup can generate accurate estimation of APs based on an efficient aggregation of sensing results using bipartite graph, and provide feasible analysis of reliability of each crowd-vehicle using an iterative inference. Through extensive simulation and real testbed results, we have showed the superiority of `CrowdWiFi` over existing approaches with respect to lookup errors (*e.g.* around 80% improvement on localization in comparison with the state-of-the-art Skyhook). We also verified the efficiency of `CrowdWiFi` middleware on some vehicular applications, such as WiFi handoff and data transmission.

# 8. REFERENCES


[1] Network Simulator and Emulator, http://NSL.csie.nctu.edu.tw/nctuns.html.

[2] http://research.microsoft.com/en-us/projects/vanlan.

[3] E. Cands and M. Wakin. An Introduction to Compressive Sampling. *IEEE Signal Processing Mag.*, 25(2):21–30, 2008.

[4] Y.-C. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy characterization for metropolitan-scale Wi-Fi localization. In *ACM MobiSys*, pages 233–245, 2005.

[5] I. Constandache, R. R. Choudhury, and I. Rhee. Towards Mobile Phone Localization without War-Driving. In *IEEE INFOCOM*, 2010.

[6] M. Ding and X. Cheng. Fault Tolerant Target Tracking in Sensor Networks. In *ACM MobiHoc*, 2009.

[7] C. Feng, W. Au, S. Valaee, and Z. Tan. Compressive Sensing Based Positioning Using RSS of WLAN Access Points. In *IEEE INFOCOM*, 2010.

[8] D. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. In *Neural Information Processing Systems (NIPS)*, pages 1953–1961, 2011.

[9] J. Koo and H. Cha. Autonomous Construction of a WiFi Access Point Map Using Multidimensional Scaling. In *ACM PERVASIVE*, pages 115–132, 2011.

[10] Q. Liu, J. Peng, and A. Ihler. Variational inference for crowdsourcing. In *Neural Information Processing Systems (NIPS)*, pages 701–709, 2012.

[11] Open-Mesh. www.open-mesh.com.

[12] A. Rai, K. Chintalapudi, V. Padmanabhan, and R. Sen. Zee: Zero-Effort Crowdsourcing For Indoor Localization. In *ACM MOBICOM*, 2012.

[13] T. Rappaport. *Wireless Communications: Principles and Practice*. Pearon Education, 2nd edition, 2001.

[14] V. Raykar, S. Yu, L. Zhao, G. Valadez, L. B. C. Florin, and L. Moy. Learning from crowds. *The Journal of Machine Learning Research*, 11:1297–1322, 2010.

[15] S. Wireless. http://www.skyhookwireless.com.

[16] D. Wu, L. Bao, and R. Li. Robust Localization Protocols and Algorithms in Wireless Sensor Networks Using UWB. *Ad Hoc & Sensor Wireless Networks*, 11(3-4):219–243, 2011.

[17] D. Wu, Y. Zhang, L. Bao, and A. Regan. Location-Based Crowdsourcing for Vehicular Communication in Hybrid Networks. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):837–846, 2013.

[18] S. Yang, P. Dessai, M. Verma, and M. Gerla. FreeLoc: Calibration-Free Crowdsourced Indoor Localization. In *IEEE INFOCOM*, 2013.

[19] B. Zhang, X. Cheng, N. Zhang, Y. Cui, Y. Li, and Q. Liang. Sparse Target Counting and Localization in Sensor Networks Based on Compressive Sensing. In *IEEE INFOCOM*, 2011.

[20] Y. Zhang, L. Bao, S. Yang, M. Welling, and D. Wu. Localization Algorithms for Wireless Sensor Retrieval. *The Computer Journal*, 53(10):1594–1605, 2010.


# APPENDIX

## A. PROOF OF PROPOSITION 1

From the assumption, $Y'$ can be written as: $Y' = QA^{\dagger}Y = QA^{\dagger}A\Theta + QA^{\dagger}\varepsilon = Q\Theta + \varepsilon'$. Since $Q$ is an orthogonal matrix, $\Theta$ can be well recovered from $Y'$ via an $\ell_1$-minimization based on the CS theory.

## B. PROOF OF PROPOSITION 2

The matrix $Y$ of compressive noisy RSS measurements is not given. If we were to test exhaustively for the value of $Y$, we would need to examine each of the possible combinations of $K$ AP and $M$ RSS measurements. Each RSS value can belong to only one AP, and thus $K \leq M$. If $K$ was known, for a given $K$ and $M$, there are $K^M$ possible combinations in total. In the worst case $K = M$, and there can then be $M^M$ combinations. If $K$ is not known, then in the worst case we must exhaustively test each assignment of $K = 1, 2, \ldots, M$. There are $\sum_{K\in 1:M} K^M$ such combinations. Since it is clear that $\sum_{K\in 1:M} K^M < \sum_{K\in 1:M} M^M$ and $\sum_{K\in 1:M} M^M = M \times M^M = M^{M+1}$, then we have: $\sum_{K\in 1:M} K^M$, and it is of complexity $O(K^{M+1})$. Furthermore, since the last term of the combination sequences is $M^M$, the calculation complexity of combination on (AP, RSS) is $\Omega(M^M)$.