Institute of Parallel and Distributed Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit Nr. 0202-0013

Design and Implementation of an Occupancy Monitoring Method for Indoor Public Sensing Applications

Mohammed Qaid Abdul-Razzaq Mohammed

Course of Study:	INFOTECH
Examiner:	Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel
Supervisor:	Mohamed Abdelaal, M.Sc.
Commenced:	June 1, 2016
Completed:	December 1, 2016
CR-Classification:	C.2.4

Abstract

Recently, new architectures of the modern smart phones come with integrated powerful platform sensors. Moreover, indoor environments are typically rich with such smart devices. By considering these advantages, the mobile phones can be exploited for collecting measurements through a *public sensing* (PS) system. As a motivation for the mobile users to participate in the data sensing process, we should restrict the energy drawn by the sensing tasks. Several energy consumers may negatively affect the participating mobile devices while collecting measurements. Among these consumers are: (1) data acquisition, (2) measurements uploading, (3) overhearing superfluous sensing queries, and (4) position fixing and updating. Actually, a significant amount of energy is dissipated, on the mobile devices, due to updating the PS server with the current position. Several research work has devoted to reducing this overhead through either piggybacking the update messages or through reducing the number of position updates. However, there exists still an overhead on the mobile devices due to estimating the position and reporting it to the corresponding PS server.

In this thesis, we consider an alternative approach to overcome this energy overhead. We propose to opportunistically exploit the already-existent WiFi traffic to monitor the occupancy of a certain area by the participating mobile devices. Accordingly, we move the burden of estimating the position and reporting it to a PS server, from the mobile devices to the PS servers. At the outset, we investigate the applicability of our proposed method through experimentally studying the WiFi traffic. The traffic analysis confirmed that plenty of WiFi messages are exchanged between mobile devices and the WiFi access points (AP). Based on these findings, the proposed occupancy monitoring method is divided into two scenarios: (1) multiple APs localization scenario; which is applied when there are enough uplink traffic from the mobile devices, hence the user's location can be estimated using RSSI from multiple APs; (2) a single AP localization scenario when there is no enough WiFi traffic but the server still receives the RSSI measurement of mobile device from the associated AP. Specifically, two algorithms constitutes our proposed algorithm, namely *device detection and data collection* algorithm and *position estimation* algorithm. The former runs at the AP level to detect the existing devices in their coverage area whereas the latter algorithm runs on the server side to gather the RSSI measurement and to calculate the current user location. We adopt the fingerprinting strategy as our position fixing method. Generally, fingerprinting has two phases including an offline and an online phase. For the offline phase, we adopt affinity propagation to cluster the collected RSSI measurements. For the online phase, we provide a comparative study between adopting the *K-nearest neighboring* algorithm and the *compressive sensing* algorithm. For evaluating the performance, we construct a testbed and several experiments have been carried out. The findings show a localization accuracy of circa two meters which is achieved via adopting compressive sensing.

Acknowledgment

I would like to thank the following people for all their effort, motivation, and support during the thesis time. I own my gratitude to Prof. Dr. Kurt Rothermel for the opportunity he provided me to write my thesis in the Institute of Parallel and Distributed Systems. In addition, I would like to express my deep thanks to my supervisor, Mohamed Abdelaal for offering valuable advices, for his support during the whole period of this thesis, and for his patience, guidance, and help during the writing process. I also thank Dr. Frank Dürr for introducing me to the topic.

I would to thank all my friends who helped me during the thesis time where a special thank goes to Tareq Al-maamary for his great assistance during doing the experiments and during writing the thesis. Furthermore, I would like to thank Prof. Dr. Francesco Gringoli, from the University of Brescia, for providing me with the source code (Open-FWWWF) and his help to establish the environment.

I would like to send my deep thanks to my sponsor, the Sana'a University and to my colleagues there. I also would like to thank my advisors at Fraunhofer institute, Jonathan Masior, Stephan Schuele, Jacoba, and Mehmet for their understanding and giving me time off work to complete this thesis. Last but not least, I would like to thank my parents and brothers for their love and support. Without their encouragement, I would have never given myself the chance to continue my education. I thank the one above all of us, the omnipresent God, for everything.

Contents

List of Figures

List of Algorithms

1 Introduction

1.1 Motivation

With the recent advances in semiconductor processes and technologies, the hardware architecture of the modern mobile device enables it to be more powerful while at the same time consume less energy. For instance, modern smartphones make use of the latest generation of multi-core processors for higher computational and processing capabilities. Additionally, they embed larger and faster memory chips for better multitasking performance, and integrate a variety of smart sensors such as accelerometers, dual cameras (3D sensors), microphones, and *Global Positioning System (*GPS*)* receivers to provide a wide range of capabilities.

In fact, modern mobile devices support newest versions of the communication technologies such as the latest generations of WiFi and Bluetooth standards. With these modern communication technologies, a mobile device can establish faster and more reliable wireless communications while consuming less energy. For example, smartphones can communicate with nearby wireless devices, such as wearable devices (medical devices, smart watches, smart glasses, and fitness trackers) to obtain the user's blood pressure, stress level, and body temperature or other health and environment parameters using the *Bluetooth Low Energy (*BLE*)* technology. These hardware, software, and communication aspects of modern mobile devices make them an indispensable part of our daily life, paving the way for a new concept called Public Sensing (PS).

Generally speaking, the PS systems exploit crowdedness of mobile devices to collect measurements from the surrounding environment. Specifically, they obtain data from a set of embedded sensors. One important aspect of PS systems is to convince users to participate in the PS system. Users typically are concerning with the lifetime of their batteries. Hence, reducing the energy overhead due to PS tasks can motivate mobile devices' owners to participate in the PS systems. In particular, sources of energy consumption in PS systems comprise data acquisition, localizing the mobile devices and reporting these data to the PS servers.

In this thesis, we target relieving the energy consumption due to localizing mobile devices and reporting the position information. Specifically, we consider a subset of the

PS applications in which the sensing tasks are entirely scheduled in indoor environments. In such cases, indoor localization methods are to be utilized to identify the users' current position. For the PS servers to localize the participating mobile device, position queries and reports have to be frequently exchanged between each mobile device and the PS servers. To this end, there exist several technologies and proposed solution to the user positioning problem as alternative to the GPS technology which is ill-suited for indoor environments. Examples of these indoor positioning methods include: ultrasound-based positioning systems, infrared-based positioning systems, and Radio frequency based positing system [FNI13].

Mostly, position updates are recurring due to the varying mobility pattern. Hence, manifold problems emerge like the excessive energy overhead on the mobile device, the networking overhead, and the extended update latency. Unfortunately, mobile devices have typically limited energy sources, therefore indoor positioning methods have to preserve the allocated energy. Several endeavors have been undertaken aiming at addressing this issue from different perspectives. For instance, the authors in [LR01] propose numerous methods to reduce the number of update messages. In [KLGT09], sensing management strategy to improve the energy efficiency by scheduling and controlling the power on/off of sensors devices for reading the position barrier is proposed. In [BDR13], the authors introduce an opportunistic position update protocol. They investigate the energy characteristics of the 3G mobile network interface. Accordingly, they piggybacked the position update messages to other messages generated by various mobile applications.

Although the above solutions managed to reduce the energy consumption due to positioning and reporting the position fixes, there still exists remarkable energy burden on every mobile device. Furthermore, these proposed methods did not consider the tradeoffs between energy efficiency and other characteristics of position update such as latency and accuracy. For instance, *Dead Reckoning (*DR*)* is a well-known method used to estimate the current location based on the last estimated location and speed of the mobile device, as will be explained later. In the DR method, identical predictors have to be implemented on both of the mobile device and the PS servers. As a decision rule of reporting the position, DR defines a threshold deviation between the real position and the predicted position. Hence, a small threshold value leads to an increment in the update messages which significantly affects the energy efficiency.

To overcome these challenges, we propose to convert the research problem from developing energy-efficient position update protocols into designing an occupancy monitoring method through which sending position updates by mobile devices are sidestepped. Such a method is completely achieved on the PS server side without any intervention from the mobile devices. Considering an indoor environment with Wi-Fi availability, each mobile device frequently sends probe messages to connect to or to ensure connectivity to Wi-Fi access points and as we showed in our traffic studies that even if the mobile inactive but still sending frames frequently. Accordingly, the traffic analysis gives us the motivation to propose an approach that exploits this already-existent traffic for enabling the PS server to determine the occupancy of certain areas by mobile devices.

Aside from relieving a significant energy overhead from mobile devices, this proposed method has several other advantages. For instance, it handles different mobility patterns. Either the mobile devices are stationary or frequently moving, the occupancy monitoring method can detect them whether the mobile devices are communicating with the scattered Wi-Fi access points or not. Additionally, the incurred delays due to position update messages are entirely avoided. In our case, the PS server has approximately an instantaneous map of the mobile devices within the building. The main shortcoming of the proposed method is the user privacy. However, the user identity is still difficult to be revealed due to the long time between subsequent messages sampling. Additionally, some Apple devices are capable of frequently changing their *Media Access Control (*MAC*)* addresses. This feature will be integrated in all upcoming mobile devices. As a result, the proposed occupancy monitoring method does not contradict with users' privacy.

1.2 Public Sensing Overview

Public Sensing (PS) is a new technology to build low-cost wireless networked sensors. The main idea behind PS systems is to sense the environment via exploiting the powerful capabilities and features of modern mobile devices. In particular, smartphones became indispensable part of our daily routine. These mobile devices exist everywhere and cover a large geographical area where they can inexpensively report the sensed data by means of the existing wireless network infrastructure such as WiFi and 3G networks. The concept of PS systems was first proposed by Abdelzaher et al. in [WAK15]. Since 2007, several research work has devoted to develop new PS applications and to optimize the associated *Quality of Service (*QOS*)* metrics. Nowadays, most mobile devices are equipped with a large variety of sensors such as accelerometers, proximity sensors, microphones, magnetic field sensors, GPS senors, and cameras. Furthermore, some vendors come with specific sensors. For instance, a new model of Apple's iPhone come with a smart sensor such as barometers and gyroscopes as well as IBeacon (BLE modules). Similarly, wearable devices, such as smart watches and smart glasses can be easily integrated into the PS systems.

In fact, PS systems can be classified according to the human intervention into opportunistic sensing and participatory sensing. In the former class, the PS servers passively collect sensor data without including the mobile devices owners in the sensing tasks. Thus, sensor data is gathered as background operations such as position fixing using

IBeacon and occupancy monitoring, and detecting the orientation. The second approach is a participatory sensing approach; sensor data is collected by interacting with the mobile device. PS system asks the user by sending a request to a running application on the mobile device that can process the request and respond to it. This thesis takes the opportunistic sensing into consideration. In which, the sensor data collection and the position estimation are performed by the server without the need to modify or install any software on the mobile device.

1.2.1 Architecture

A typical PS system comprises a set of participating mobile devices, servers, communication networks, and client applications. Figure 1.1 shows the general architecture of PS system. A PS server is responsible for serving all queries from different applications in the system. A server manages the sensing query, controls the sensing processes and coordinates the communication between the served clients and the participating mobile devices. Specifically, the PS server has a *query interface* which makes the communication between a client and a server easier and more flexible. The query interface is used by a client to send queries and to receive the required data. In addition, the PS server contains a *basic sensing system* which is responsible for managing the received queries. The basic sensing system decodes the received queries and then forwards it to the set of participating mobile devices. To this end, a *position update protocol* keeps track of the mobile devices in the sensing area.

Similarly, the participating mobile devices run a basic sensing system which is responsible for managing the received sensing queries. The basic sensing system on the mobile device has two units, query listener and sensing engine. The query listener waits for any query and forwards it to the sensing engine. Subsequently, the sensing engine manages the sensing process by controlling the sensors. For example, activating or deactivating a sensor to perform a measurement, such as sampling the temperature in a specific area and fixing the position of the mobile device. The sensing engine answers the query by sending the requested sensor data back to the server. Additionally, an update position protocol frequently senses the position of the mobile device and sends an update to the PS server. More details about positioning can be found in Section 2.3.

1.2.2 QoS Requirements

There are several *Quality of Service* (QoS) requirements for the PS systems. For instance, the PS system should guarantee that the whole geographic monitoring area is covered by leveraging the availability of the mobile devices. Moreover, it should guarantee the

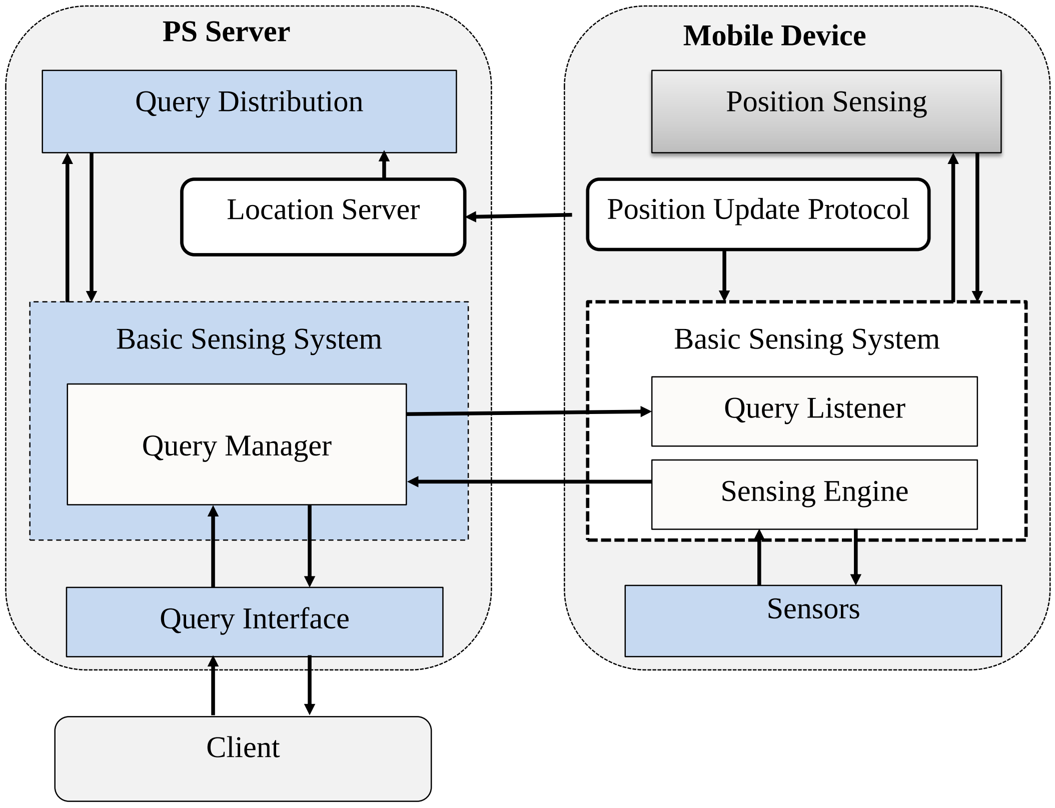


Figure 1.1: General architecture of PS systems [Bai15]

quality of the gathered data. In other words, the accuracy and precision of gathered data that could be the impact of unreliable data sources. Furthermore, it should motivate and attract the users so that they accept to participate in the data gathering process. To ensure the user's acceptance, the PS systems have to be highly energy efficient so that the energy resource on the participants mobile devices is preserved. An important QoS metric is the user provacy which has to be considered while desiging the PS system. Finally, the PS system should guarantee the usability by implementing and offering convenient *Application Programming Interface (*API*)*s [Kur15].

1.2.3 PS Applications

Due to their powerful capabilities, the PS systems can be applied in different areas, like environmental monitoring, advertising, indoor mapping applications, and traffic monitoring. PS opens the door for developing new powerful and smart applications and services for both big and small companies. PS enables an application to make the right decision and interact with the users based on data that is gathered in real-time. Below, we present some examples of the existing PS applications.

1.2.3.1 Environmental Monitoring

The PS system has many applications in the observation and analysis of the environmental changes. It can collect data from different sources such as from sensors installed in buildings or from sensors that are embedded in mobile devices. One such application is collecting noise pollution data using the microphone of a mobile device. For instance, the authors in [RCK+09] introduce the earphone system which is a sensing system that measures the noise pollution using the microphone built in the mobile devices of volunteers and sends the sensor data to a server. From this data, a reconstructed noise map is generated by exploiting the comparative sensing algorithm to manipulate the missing samples.

1.2.3.2 Indoor Mapping

The PS system can be used to automatically derive the floor plan of a building. For example, the ComNSense project – developed and implemented by Stuttgart University [**comnsense**] – aims to develop a universal method for the automated generation of interior models. It takes advantage of the mobile devices equipped with sensors and the fact that the indoor environment is crowded by such devices. ComNSense tracks the mobile devices in order to automatically derive indoor floor plans. Its major goals are:

- automatically derive 2D floor plans from the data collected through indoor tracking of mobile devices or foot-mounted-sensors.
- derive 3D indoor models from point clouds and 2D images samples using the recently-released Google Tango tablets.

Many contributions have been done in the project. One of those contributions is the development and implementation of an indoor grammar approach for building the interior models. This is used to manipulate the noise and draft-based of the collected data. The grammar approach shows a significant increase and improvement in the accuracy and robustness of the indoor plane floor generation. [Kur15]

1.3 Research Objectives

In this thesis, the author aims to achieve the following research objectives:

- conducting a thorough investigation of the work done in the field of energy-efficient indoor localization;
- developing and Implementing an indoor occupancy monitoring method for public sensing; and
- evaluating the proposed approach via building a testbed and performing real experiments.

1.4 Contribution

The outcome of this thesis contributes in overcoming problems of the existing position update protocols. In other words, the primary goal is to develop and to implement an indoor occupancy monitoring method on the PS servers by exploiting the existing IEEE 802.11 traffic. The implementation of the concept goes through two phases, namely: detection and data collection phase; and localization phase. The concept uses a Wi-Fi based technology, namely the *Received Signal Strength Indicator (*RSSI*)*. For the detection and data collection phase, a detection method is developed and implemented to discover all the devices available in a certain geographical area. For the localization phase, RSS fingerprint based method is selected which is recommended for the indoor positioning. To estimate the exact location and construct a map of the mobile devices, this thesis implements and compares between two methods, namely: K-Nearest Neighbors (KNN) and Comparative Sensing, which are well-suited candidates for a RSS fingerprint-based method.

1.5 Document Structure

The rest of the document is organized as follow. In Chapter 2, the system model and the problem statements are presented, some fundamental concepts, such as position update protocols and indoor localization technologies as well as the related work are elaborated. In Chapter 3, the WiFi traffic experiments are explained and the obtained results are discussed. Chapter 4 gives a clear description of the concept of our approach and each part is explained in more detail. Chapter 5, the implementation of our approach is discussed. specifically, the description of the hardware and software components of the detection and data collection method are explained. Then, the implementation of affinity propagation, KNN and compressive sensing is given. Afterward, the evaluation of the approach is explained where the real experiments are described and the performance of KNN and compressive sensing are evaluated. Finally, Chapter 6 concludes the thesis and gives some suggestions for future work.

2 System Overview

As discussed in the previous chapter, energy consumption is a significant design parameter of all PS systems. In other words, PS designers have to ensure a minimum energy overhead on the participating mobile devices. To tackle this problem, we identify the main energy consumers in most PS systems. Due to the requirements of the ComNSense project, we consider only a subset of PS systems which are designed for indoor operation. In this chapter, we explain the system model before we formalize the attacked problem of reducing the energy overhead on the mobile devices. Subsequently, we present an overview of the existent indoor localization methods.

2.1 System Model

In this section, we describe the system under investigation in this thesis. Moreover, we introduce our assumptions that have to be considered while resolving the energy overhead problem. As discussed earlier, the thesis targets a subset of PS systems in which the entire system operations are performed in indoor environments. For instance, comNSense project targets autonomously generating 3D interiors model through crowd sensing. Accordingly, assuming a building is equipped with n WiFi *Access Point (*AP*)*s, each AP covers an area whose radius is r meters. These areas are deliberately overlapping to guarantee full coverage. For simplicity, we assume a single PS server which is physically connected to the scattered APs. The building has m mobile devices users who are freely traversing the various floors. Figure 2.1 depicts the architecture of the proposed method. As it can be seen in the Figure, the system consists of main four components as follows:

- A set of mobile devices, which send association requests and exchange normal traffic (e.g. web browsing, video streaming) with the associated AP.
- A set of APs, which detect and read measurement data.
- A localization server, which is deployed on the top of thePS server.
- A communication network, which allows the mobile devices to communicate with the APs.

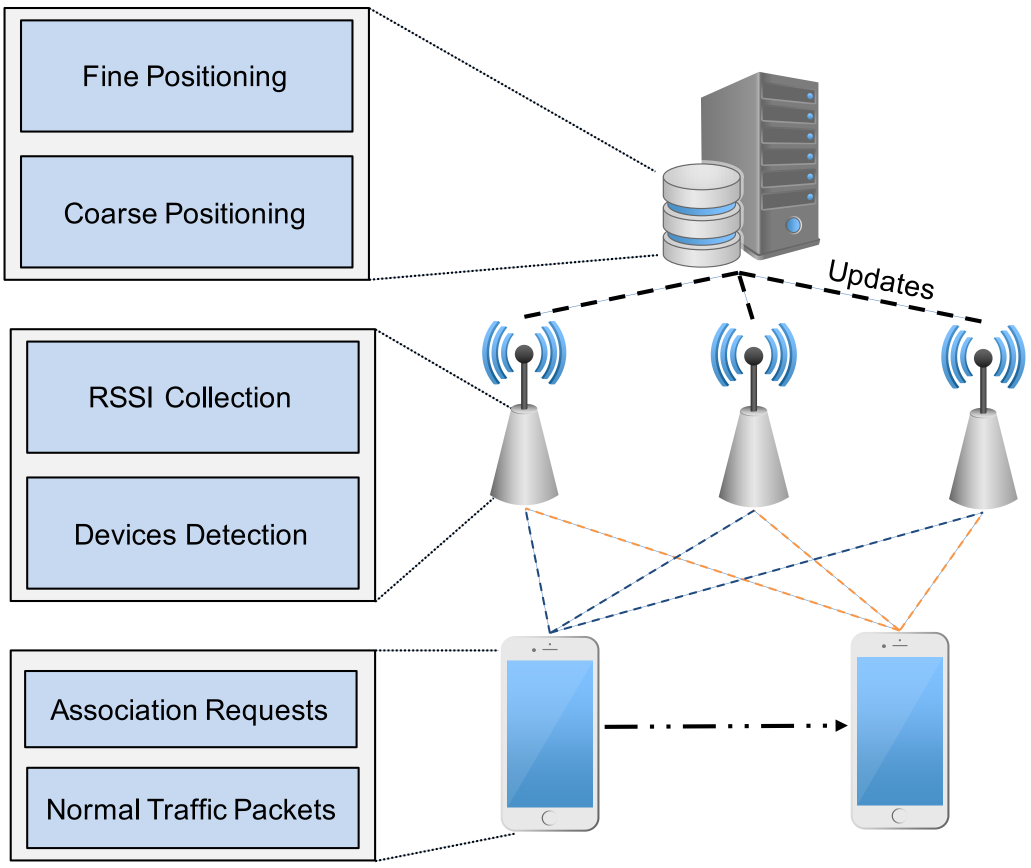


Figure 2.1: Sytem model architecture

In the sequel, we briefly describe the role of each component:

Mobile Devices The main goal of this thesis is to localize the mobile devices in a certain geographic area to construct a map of the available mobile devices. We assume that the mobile devices enable their WiFi and they frequently send probe messages and enable client roaming mode during the movement of the user between the APs. Thus, our proposed method does not require to install any extra software on the mobile device.

Access Points In general, most of indoor environments have a set of APs that are already installed to wirelessly communicate with mobile devices. We assume these APs support a lightweight embedded Linux distribution such as OpenWRT and DDR-WRT. Accordingly, we are able to run an application which is installed on each AP to monitor the surrounding area. Each running application detects all mobile devices that can be seen from a certain AP. It also reads measurement data such as the *received signal* *strength indicator* RSSI of each detected mobile devices and then sends an update to the server.

Localization Server In our approach, the localization processing is performed on the server side. A localization algorithm is responsible for gathering the sensing data from all APs and for calculating the current device's position. The estimated positions are used to update a map of mobile devices on the server.

Generally, several components contribute to the energy consumption problem, including: (1) sensor data acquisition, (2) data uploading, (3) position fixing and server updating with new locations, and finally (4) overhearing sensing queries. Activating the sensing module of a mobile device to sample data is energy-costly, especially for frequent data logging. For instance, the gyroscope sensor embedded in IPhone 4 consumes approximately 175.6 mW per each data sample [KMD13]. Similarly, reporting the measurements to the PS servers is extremely "energy-expensive". It is known that sending one bit of information consumes energy similar to consuming for processing one thousand line of codes. Another fact is most modern transceivers is the same of amount of energy consumed for data transmission and data reception. Therefore, receiving sensing queries outside the sensing area represents wasting energy due to the reception of superfluous queries.

As described in the previous chapter, the PS servers have to be aware of mobile devices in the sensing area. Hence, the participating mobile devices are asked to send their position. This implies that each mobile device has to frequently activate its location sensing module and its transceiver to report the current position. In this thesis, we consider a different approach in which we remove the burden of localization and position update from the participating mobile devices to the PS servers. Below, we describe the system model before we formalize the investigated problem.

2.2 Problem Statement

The traditional position update protocols face many problems such as energy consumption on mobile devices, latency update, network overhead and low accuracy. Several approaches have proposed and addressed those issues. Due to the tradeoff between the energy efficiency and other properties of indoor positioning, implementing an energy-efficient localization system is one of the biggest challenges. The main objective of this thesis is to tackle the problems with the existing *indoor positioning systems* (IPSs) through designing and implementing of an indoor occupancy monitoring. In this section, we will define the problems of typical IPS systems in terms of the energy overhead on mobile devices due to frequently sense their location, and the incurred latency.

2.2.1 Energy Consumption

Mobile devices typically consume a large amount of energy during the process of updating their position. The source of energy consumption can be divided into three sources: First, the IPS requires frequently to sense the position in order to update its current position which leads to power consumption by the used sensor and some localization algorithms use many sensors to navigate on the map. In terms of energy consumption, sensors can be categorized into two categories: 1) sensors that consume higher power such as the Wifi module and the GPS sensor, and 2) sensors with lower power consumption such as the magnetometer sensor which is used to measure the orientation, and the accelerometer sensor which is used to measure the speed of the object. Both of the accelerometer and the magnetometer sensors are used to for localization.

Most approaches focus on the energy consumption of the energy-expensive sensors and on reducing the need for reading data from high-power sensors by exploiting the map information, floor plan and using the low-power sensors and predicate the next positions. Few approaches that take into consideration the power consumption on the low-power sensors (e.g. [KBBN11]). The second source of energy consumption is that IPS frequently sends current position as an update message to the PS server. The power consumption due to sending the update message is higher than others operations. Thus, many researchers strive to reduce this energy consumer. For example, *dead reckoning* is an efficient approach that tackled this issue via employing predictions.

Finally, the power consumption due to the computational operations is another source of energy dissipation. Compared to the two previous consumers, the computation's energy consumption can be ignored. Since energy consumptions is an important metric that has to be considered during the design of a position update system, several research work has been performed from different perspectives. For example, Leonhardi and Rothermel [LR01] study and classify a class of update position protocols which reduce the energy consumption by reducing the number of update messages. The authors in [KBBN11] propose a management sensor strategy in order to reduce the power consumption via controlling and scheduling the tasks of running the sensors on mobile devices to fix the position. Alternatively, the authors in [BDR13] describe an opportunistic update strategy. The proposed strategy reduces the power consumption that is consumed on a mobile device by utilizing a Markov algorithm. This algorithm predicts the next coming traffic in order to send the update message with normal traffic instead of activating the 3G network module to send the update message. To sum up, many efforts have been achieved to address the problem of high power consumption on mobile devices. However, mobile devices still have to consume an amount of energy update their position. In this thesis, we propose an opportunistic algorithm which exploit the already-existent WiFi traffic to enable the PS server from localizing the participating mobile devices. Accordingly, the burden of localizing mobile devices is completely relieved from the mobile devices side.

2.2.2 Latency Update

In particular, clients send queries to the PS server asking for data. If the server has no fresh position data, it can not determine the possible candidates to perform the required sensing task. Then, the PS server waits the available mobile devices for a certain time to send position updates. If we consider our proposed solution, we find that the PS server continuously perform localization, assuming unlimited energy source allocated for the server. Hence, the server will always retain fresh information about the users distribution in the sensing area. Formally, we can define the attacked problem as follows.

$$\text{minimize} \sum_{i=1}^{m} (E_c(i) + E_s(i) + E_u(i)) \tag{2.1}$$

$$\text{minimize} \sum_{i=1}^{m} (D(i)) \tag{2.2}$$

subject to

$$P_s = P_{current} \quad \forall m \tag{2.3}$$

The terms E_c, E_s, and E_u denote the energy consumption required to perform localization computations, position data acquisition, and transmitting the position to the PS server, respectively. Thus, Equation 2.1 describes the main objective of reducing the energy consumption due to the different sources. This reduction should be achieved for the m mobile devices participating in the sensing task. Similarly, Equation 2.2 denotes the objective of reducing the incurred delay D due to waiting for the mobile devices to send their position updates. These two goals should conform with a quality condition, give by Equation 2.3. In this equation, we declare that the position of a mobile device stored at the server P_s should be the same as the actual current position $P_{current}$ of this mobile device. Below, we explain the well-known indoor positioning methods before we introduce our proposed method in the next chapter.

2.3 Indoor Positioning Systems

*Indoor Positioning System (*IPS*)* is a user location tracking system of indoor environments like such as in a university or a hospital. As illustrated in Figure 2.2, the indoor positioning system consists of three components. The first component is a *Localization Sever (*LS*)* which contains a secondary copy of location information for all mobile objects. The secondary copy is used by the LS to answer the incoming queries by the clients. A second component is a mobile device which has the primary copy. The primary copy is read directly from the position sensors. To sense a position on the mobile device, several techniques are used in indoor position systems as explained in the subsection 2.3. The server and mobile devices run a position update protocol to sense the position on mobile devices and an update message is sent to update the secondary copy on the server. The network communication, for instance, between the mobile device and the existing WiFi network is used to send an update message in order to update the primary copy. The last component is the client application which inquires about the position information of a mobile device.

The huge amount of update messages causes many problems such as consumption of the energy resources on the mobile device and the network overhead. Hence, the key challenge of IPS is to overcome these problems and achieve a higher accuracy. In the following subsection, the components and the general structure of the IPS as well as the update protocols will be presented.

2.3.1 Location Servers

As illustrated in Figure 2.2, the LS runs a position update protocol to manage the location information of the mobile devices in a certain geographical area. The location server stores a secondary copy that comprises the location information about mobile objects such as the speed of the object, the “(X,Y)” location coordinates as well as information about the region such as name and address. Moreover, LS is deployed on the cloud and provides an API to allow clients to query the current position of a mobile object. With Public Sensing system, location server is deployed on the top of PS server to construct the map of available mobile devices in a certain geographical area. In order to synchronize the update process between the secondary copy on the LS and the primary copy on the mobile object, several approaches have proposed and implemented a position update protocol. The next subsection describes the classes of the update protocols.

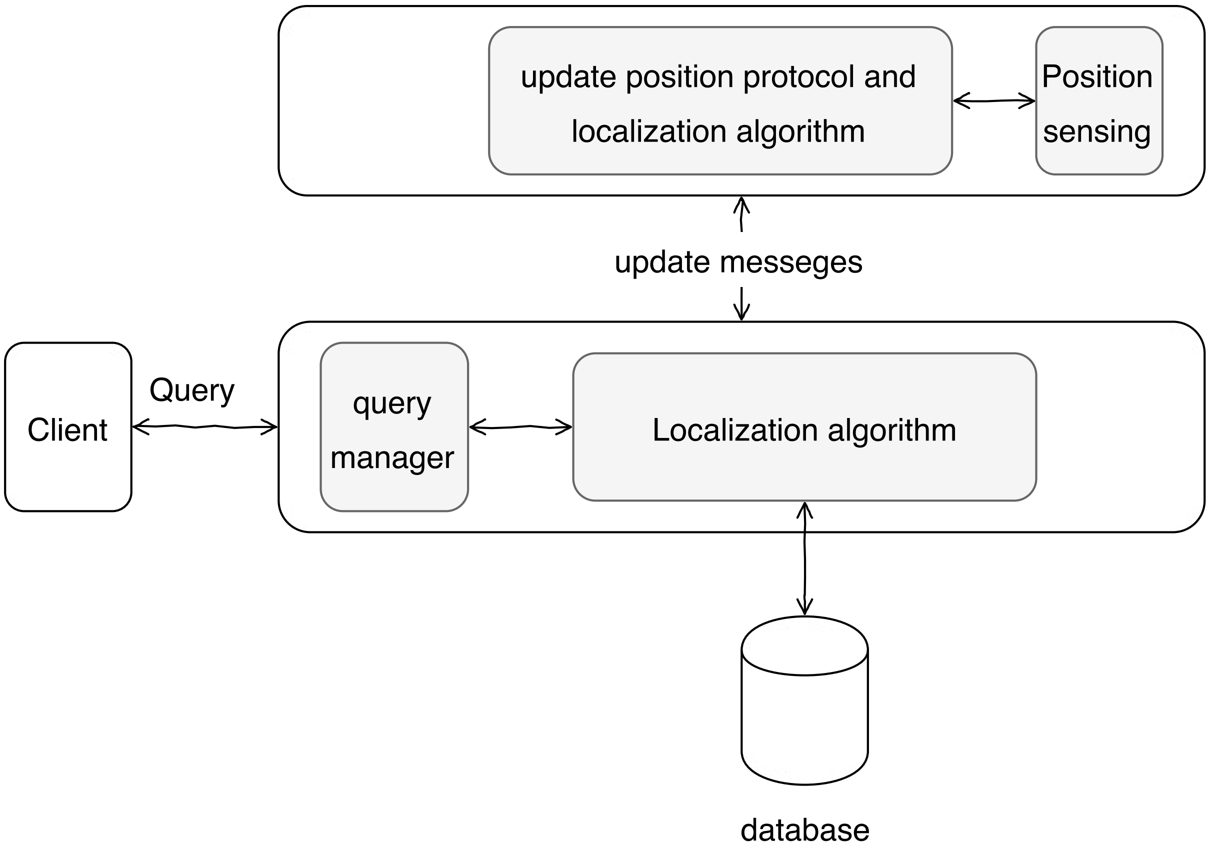


Figure 2.2: General Structure of the IPS

2.3.2 Update Position Protocols

The mobile object senses its location (primary copy) and updates a secondary copy stored on a location server. A mobile object sends an update frequently to make sure that the secondary copy on the location server is up-to-date, to guarantee a higher accuracy and lower latency delay. Sending the update for each sensing position results in more energy resource consumption on mobile devices and consuming the network bandwidth. According to [LR01], the update position protocols are classified into three types: (1) querying update protocol which the LS asks the mobile device for sending update whenever the queries about information location of the mobiles are received, reporting update protocols which the mobile device frequently send updates to the LS and hybrid update protocols which both aforementioned types are combined.

2.4 RSS-Based Technique

A radio wave travels from a source to a destination in all directions and in a straight line in the free space. The traveling radio wave may be reflected by barriers in indoor environment which cause signal path loss and signal fading. Hence, the power of the signal is lost during the traveling between the source and the destination. The lost power varies depending on the distance and the barrier in the way of the signal to the destination. To model the energy loss of the received signal, the term *Received Signal Strength (*RSS*)* is used. The advantage of using RSS that it supports and can be measured by the most of the commercial wireless network devices. Therefore, the RSSI is used for various purposes. For example, RSSI is used to measure the quality of network link. Moreover, RSSI value is specified in – db which is used to implement a localization system.

The RSS-based method has a few disadvantages: (1) RSSI has a slight linear relation between the RSSI and distance, (2) The measurement of RSSI is unstable and (3) The RSSI at the same point can be changed from time to time due to the multipath and presence of the barriers in an indoor environment such as walls, closed doors, human bodies and furniture. Thus, key challenge to use RSS-based for the indoor position is to overcome the problems of the RSS-based technique by utilizing the vectors of RSSI that are collected from multiple APs. The difference of the RSSI measurement from multiple APs can be utilized to characterize and to describe a certain physical location. There are two methods to model the RSSI in order to describe the information of the user position, including the *radio propagation modeling* methods and the *fingerprint-based* method.

2.4.1 Radio Propagation Modeling

The distance between a mobile object and a known position reference point (e.g. AP) can be presented as a model. Such a model describes the relation between the RSSI and the distance. The basic model of radio propagation modeling is given as denoted by Equation 2.4.

$$\begin{aligned} (d) = P(d_0)[dBm] + 10log(\frac{d}{d_0}) = P_T - \text{RSSI} \\ d = d_0 * 10log(P - \text{RSSI} - \frac{A}{10n}) \end{aligned} \tag{2.4}$$

where $P(d_0)$: the RSSI value at a reference distance that is taken during the calibration phase, typically 1 meter,

d: the distance between the source and destination,

d_0: reference distance $A = Pathloss(d_0)$, usually 1 meter

P_T: the transmitted power,

n : path loss exponent, typically 2 for free space,

RSSI: the runtime RSSI at the unknown position.

Most of the WLAN cards have omnidirectional antennas. As illustrated in Figure 2.3, the estimated distance between a sender and a receiver represents the radian of the circle (r) where the position of the mobile target is estimated as a point located on the perimeter of the circle. The estimated distance can be formulated as

$$r = X^2 - Y^2. \tag{2.5}$$

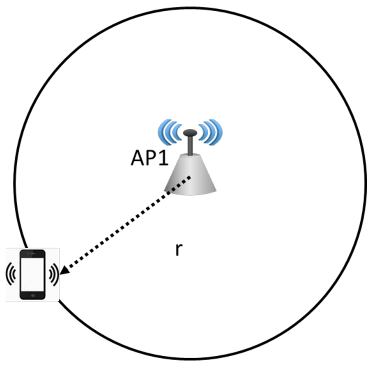


Figure 2.3: 1D estimated position

To estimate the exact position of a mobile device, the radio propagation modeling needs at least three nodes. In general, there are two methods to estimate the position which are triangulation and trilateration methods which is which an alternative method to the triangulation methods.

2.4.1.1 Triangular

Triangulation method uses the geometric knowledge to estimate a node position. As shown in Figure 2.4, the method requires at least three reference nodes whose position is known (N1,N2 and N3) and a fourth point (target point T) whose location needs to be calculated. The distances from T to all three reference nodes are measured. Each distance from T to Ni is presented as a circle. The exact location of the target point is estimated as the intersection of three circles.

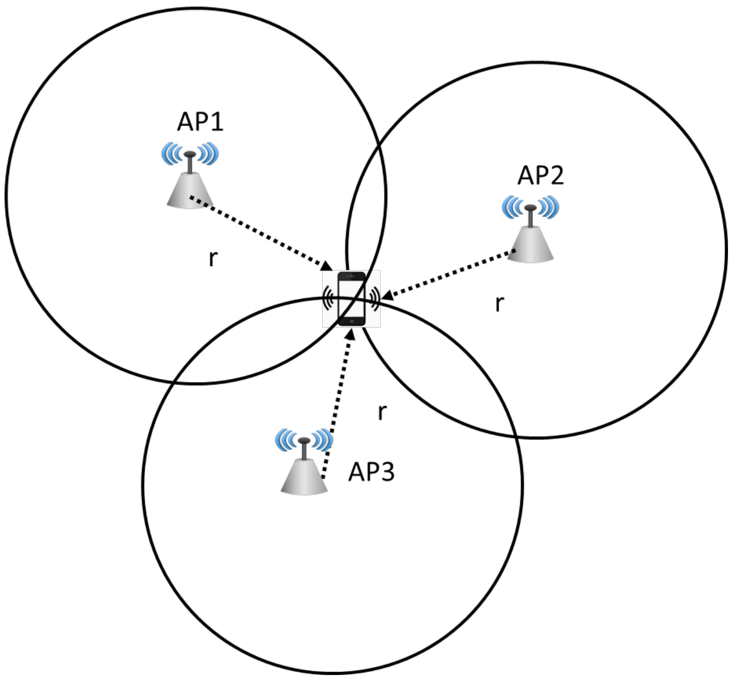


Figure 2.4: triangulation estimated position method

Moreover,the triangulation method has been extended to trilateration method where more than three nodes are used for calculating the user location which shows improving in the accuracy of localization. For more details about trilateration [Far11].

The drawbacks of the triangulation method,that a *Line Of Sight (*LOS*)* is required to be able to measure the distance or the angle. Hence, the triangulation cannot be used in the real world because of the presence of *Non-Line Of Sight (*NLOS*)* in an indoor environment. Thus, alternative method is proposed which is Location Estimation Based on Location Fingerprinting as explained in the next section

2.4.2 Fingerprinting-Based Localization

Location estimation based on Fingerprinting-based location is one of the recommended method for indoor positioning. This method is implemented as a non-line of sight mitigation approach for an indoor positioning. The implementation of this method goes through two phases: (1) offline phase (training phase) and (2) on-line phase.

Off-line Phase: During the off-line phase, the radio map of a certain geographic area is constructed by dividing the area into small cells. Inside each cell, a reference point (RP_i) is calibrated by the average of RSS values of surrounding APs which is known as a fingerprint of *Reference Point (*RP*)*. For each RP_i, an RSSI vector is stored in the database known as a fingerprint table or radio map. $RP_i = \langle \psi_{i0}, \psi_{ij}, \psi_{iN}, \theta_i \rangle$, N=1,2, . . . , n. Where *n* is the number of APs; ψ_{ij} is the average of RSSI from AP_j, θ_i contains information about the calibration point such as the (x, y) physical coordinates location and the orientation which can integrate the direction (north,south,east and west).

Online phase: During the on-line phase, the mobile collects the average RSSI values of the available APs. The RSSI values vector (Y) is collected similar to the offline phase.

$$Y = \langle \psi_0, \psi_1, \dots, \psi_n \rangle \tag{2.6}$$

The user position can be estimated by comparing the on-line measurement Y with RPs in the radio map. There are several approaches are proposed for calculating the user position such as KNN which is a deterministic-based approach for calculating the user location by calculating the centroid of the k nearest neighbors. More details are discussed later in Chapter 4.

2.5 Related Work

In this section, some outstanding approaches for occupancy detection and localization are discussed. An approach – which significantly affects the energy-efficient of the position update protocol – utilizes the human daily activities, map information. For example, dead-reckoning is one of the efficient positioning updating protocols with prediction function for reducing number of update messages. This approach draws research interests to improve the performance of the prediction function. For instance, the authors in [LNR01] describe a map-based dead-reckoning protocols for updating location information where both the mobile device and the server navigate on the map. The map dead-reckoning predicts the future course on the map by utilizing the information of the map. The information includes the intersection and information of the geographic location, as well as the last update current status of the user such as the estimated direction and speed. By utilizing map information, the performance of the prediction function is improved, however, the number of update messages decreases, while the error threshold value increases. This means the energy efficient is improved but localization accuracy are decreased.

Other approaches which studied the behaviors and the activities of human in order to improve the predication function. For example, the authors in [AGC13] described an approach for improving the performance of the prediction function. The authors uses Markov processing for predicting the user location for a future time by utilizing the human daily activities. The approach achieves in 8 hours, 69% accuracy for predicting the user location and significantly improves the energy efficient. However the accuracy is not taken into account.

The approaches mentioned above attempt to improve the energy consumption of positioning update protocol by reducing the number of the update messages. However, those approaches do not consider the burden of energy consumption on the position sensor of the mobile device during the position update operation. Thus, several energy-efficient update position protocols have been proposed to address the energy consumption of the positioning sensors. The main principle of improving the energy efficient is by reducing the need of querying the position from high power positioning sensors such as GPS , GSM, and WiFi. For example, [CCR10] exploited the low power sensors such as the accelerometer sensor and the compass sensor. However, the approach shows an improvement in the energy efficiency, but the energy burden on the low power sensors is still not ignorable as the sensors are turned on for a long time. Thus, the authors in [KLGT09] describe a sensor management strategy which fully controls and schedules the power on and power off of the sensors. Besides that, it describes a robust movement-aware strategy and heading-aware strategy. Comparing to the other approaches, this approach achieved a higher energy efficiency. In general, the performance improvements of the mentioned approaches are based on the prediction algorithm and the predefined error threshold. For example, the number of update messages decreases as the threshold value increases but the localization accuracy is decreased and vice versa.

To overcome the trade off between the energy efficiency and the localization accuracy, occupancy monitoring of the building and localizing the mobile devices have to be been considered. For example, the authors in [BXN+13], present an occupancy-based HVAC actuation using existing WiFi infrastructure within commercial buildings where the occupancy of the HVAC zone is monitored by capturing the traffic from mobile devices at the AP level. The approach also utilizes the RADIUS server which is as a part of the WPA2802.1x protocol for acquiring information about the mobile devices. RADIUS server supports the information about the MAC address and indicates when the mobile device moves from AP to AP. The accuracy of the approach was 85% of the time, the estimation was true.

Another occupancy monitoring is presented in [MRNC11] which namely calls implicit occupancy sensing where existing IT infrastructure. The approach requires additional hardware and software in order to monitor the occupancy of a geographic area by the mobile devices. The approach consists of three tiers. In tier I, the MAC and IP addresses are gathered from the existing IT such as routers and AP. In tire II, additional software is required for monitoring the activities on the existing IT, such as the keyboard usage and the mouse movement and the PC activity monitor. In tier III, additional hardware is installed, such as webcams and other sensors. In this thesis, we strive to improve the occupancy monitoring in approach [BXN+13] by improving the detection function to be able to detect both associated mobile devices and un-associated mobile devices. Afterward, the exact user position is calculated instead of predicting the existing of the mobile device. Moreover, we avoid the utilization of additional hardware devices.

3 WiFi Traffic Analysis

As earlier highlighted, the primary contribution of this thesis is the exploitation of the already-existent WiFi traffic to monitor the occupancy of a certain area. Hence, we investigate, in this chapter, the WiFi traffic to check the applicability of the proposed method. To better understanding the network traffic in wireless networks, an overview of WLAN standards are explained, before we introduce the experiments of measuring the WiFi traffic. Afterward, we classify this collected traffic into: (1) association messages and (2) normal traffic emerges from the typical mobile Apps such as Youtube and Facebook.

3.1 WLAN standards overview

The IEEE 802.11 standard is an implementation of WLAN which specifies the media access control (MAC) and the physical layer (PHY) of the WLAN. IEEE 802.11 supports various frequency bands (e.g. the data rate at 2.4 GHz is up to 2 Mbps). In general, IEEE 802.11 uses several modulation techniques such as Frequency Hopping Spread Spectrum (FHSS), *Direct Sequence Spread Spectrum (*DSSS*)*, and Infrared (IR). There exist several variants of the IEEE 802.11 standard, most of them have 14 channels. The availability of such channels differ based on the country regulation [Pri13]. For instance, all channels are operable in Japan while most other countries permit the use of all channels except channel 14. Below, we briefly present two examples of these variants.

3.1.1 IEEE 802.11a Protocol

Technically speaking, IEEE 802.11a uses the orthogonal frequency-division multiplexing (OFDM) modulation technique. Furthermore, it employs the *Wired Equivalent Privacy (*WEP*)* and the WiFi Protected Access (WPA) security protocols. It has a data rate up to 54 Mbps in the 5 GHz band. The 802.11a protocol uses 12 non-overlapping channels in the USA and 19 non-overlapping channels in Europe [Ban13].

3.1.2 IEEE 802.11b Protocol

Specifically, the IEEE802.11b protocol is the most popular variant of the IEEE 802.11 standard that is adopted and built into many mobile devices nowadays. This fact arises to numerous advantages of the IEEE 802.11b protocol, including the lowest cost, the relatively long signal range (i.e. 115 feet), and being not easily obstructed. IEEE 802.11b uses high rate DSSS modulation technique. The maximum data rate of IEEE 802.11b is 11 Mbps per AP which is less that provided by the 802.11a protocol. Moreover, the 802.11b protocol uses only three out of 14 non-overlapping channels [Ban13].

Recently, IEEE released two variants, namely 802.11g and 802.11n. Both protocols shows better performance in terms of the signal range and the offered data rate. However, we confine our discussion, in this thesis, to 802.11b protocol which has been utilized during our experiments. Generally, the 802.11 standard supports two modes, including the authentication mode and the non-authentication mode. In fact, there are three types of 802.11 frames, namely *management frames*, *control frames* and *data frames*.

Management Frames The IEEE 802.11 supports two scan modes to discover the nearby APs, including *passive* and *active* scan modes. In the former mode, the station (STA) keeps listening for periodic beacon frames which are broadcasted by the surrounding APs. The mobile device selects an AP with the strongest signal. Subsequently, the mobile device negotiates to associate with the selected AP. Authentication frames and association frames are exchanged during the association. After association with an AP is successfully done, the STA can use both active and passive scan. In the active scan mode, a client WiFi (e.g. mobile device, laptop) broadcasts *probe request* frame on every channel to scan the nearby APs. The frame is also used in client roaming when the client is moving between the APs. The AP then responds with *probe response* frame to the client. The negotiation to associate is started. While walking between APs inside a building, only *re-association request* frames and *re-association response* frames are sent between the station and a new AP as well as *disassociation frame* is sent to the old AP.

Control Frames The 802.11 control frames are used to manage the traffic and to deliver the data between a STA and an AP. For instance, to avoid the packet Collision, the *Request To Send (*RTS*) / clear To Send (*CTS*)* mechanism was proposed. The STA or AP sends RTS to ask the destination whether it is possible to receive the data or not. Then, the destination station replies with CTS to inform that it is ready to receive the data. After receiving the CTS, the sender forwards the data frame whereas the receiver acknowledges the received data frame by using immediate *Acknowledgment (*ACK*)* frame.

Data Frames The data frame is the last types of 802.11 frames which carries the payload of higher level layer in the frame body. Important to realize that the data frame differ according to its function. In the sequel, we perform an experimental study to investigate the various WiFi frames in practice. Several experiments have been done to check the amount of daily traffic, including the three classes of WiFi frames.

3.2 WiFi Traffic Study

For studying the network traffic, several experiments have been carried out using an IEEE 802.11b WiFi network. In each, the wireless network traffic between a mobile device and the surrounding APs has been measured. The study focuses on specific WiFi frames which are used by the different localization methods. For example, in the time-based localization, frames such as RTC/CTS and DATA/ACK have to be collected. Several research articles investigated the use of such frames to design a positioning method [HW08]. In this work, four ways RTS/CTS and DATA/ACK are used to calculate the round time trip. Knowing that most of the WiFi devices can measure the RSSI, the following study gives more attention to all the up-link traffic from a mobile device, including the normal traffic and the broadcast traffic.

3.2.1 Experimental Setup

To capture and analyze the network traffic in the low-level layer, the IEEE *radiotap* header – which is a format mechanism to provide more information about the various IEEE802.11 frames such as *MACtimestamp* and RSSI – is parsed. Important to realize that these headers are not part of the standard 802.11 frame format, but are additional information added at the time of capture to provide supplementary data about the frames captured. Indeed, the IEEE radiotap header is more flexible than other headers like Prism and AVS header format [Ber16]. In order to capture radiotap header, the IEEE 802.11 wireless card has to work in the monitoring mode which also allows sniffing all the packets in the surround area. The monitor mode is inherently supported in some wireless network interface card (WNIC) where a machine that operates in a monitor mode is running as a listener.

For this study, we used the Wireshark tool installed on an Apple MacBook. The Wireshark tool is an open source packet sniffing and network analyzing tools, and offers visualization analyze as well. The MacBook laptop has been configured to work in monitoring mode. Furthermore, several experiments have been carried out in a building of three floors (Informatics building 38, Stuttgart University) where the APs are deployed. We

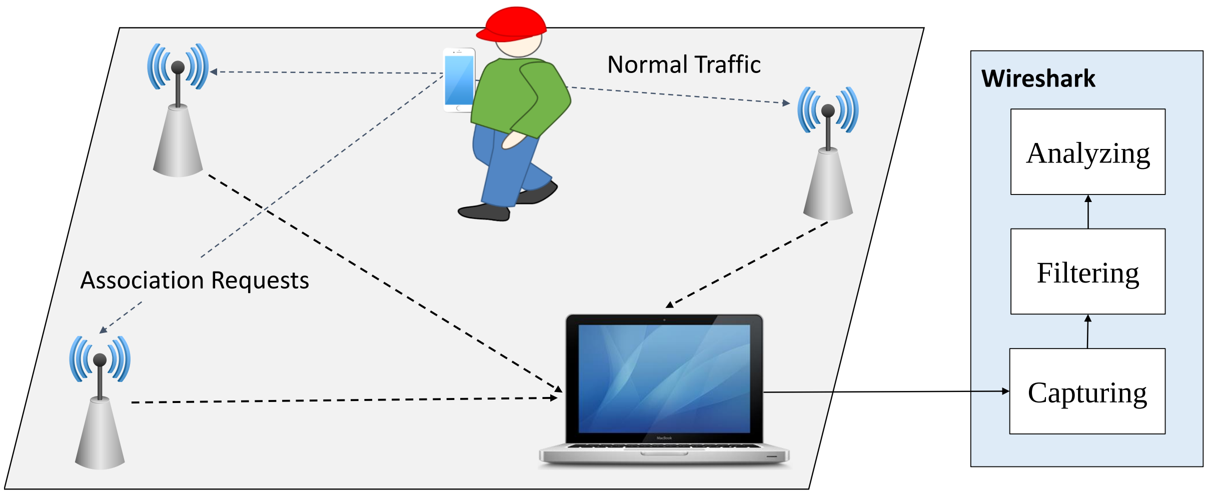


Figure 3.1: Experimental setup

employed a Google Nexus 4 smartphone as the targeted mobile device. The MacBook acts as a third party device which listen to the traffic between the mobile device and the various APs. Figure 3.1 shows the scenario that have been used in the experiment to perform the network traffic study.

During the experiments, several scenarios have been examined in terms of the mobility of the mobile device and the activities running on the mobile device. For studying the network traffic, four scenarios have been considered under different conditions.

- **Scenario 1**: mobility without activities (only background services).
- **Scenario 2**: mobility with activities (e.g. web browsing, video streaming).
- **Scenario 3**: stationary without activities.
- **Scenario 4**: stationary with activities.

In the stationary scenario, we collect uplink packets while the mobile device is fixed. Alternatively, the second scenario consider collecting WiFi packets during the movement of the mobile device. Normal activities were running on the mobile device such as browsing Internet (e.g. YouTube, Facebook) and some services were also running in the background. The capture period continues for ten minutes for each experiment. All the traffic in the air has been captured by the MacBook laptop. Later, the collected traffic has been filtered using the MAC address of the Google mobile device. This filtering step is crucial to consider merely the incoming and upcoming traffic from/to our mobile device while discarding traffic from other sources. Specifically, the experiments have been repeated four times. In each round, we consider one of the aforementioned scenarios. In the following section, we analysis the collected packets.

3.2.2 Traffic Analysis

In this section, the captured traffic during the experiments are analyzed. After filtering the traffic, Wireshark statistics I/O Graph is used to visualize the results. For each experiment, we delineate two figures to show the amount of probe request and the normal up-link traffic as the following. Figure 3.2 depicts the association packets (also known as probe requests) when the mobile device is moving and no activities are running. As it can be seen in the figure, there exist plenty of wireless message due to mobility. The movement of a mobile device forces continuous handovers between the neighboring APs to maintain the wireless connectivity. Figure 3.3 demonstrates the normal uplink traffic from the mobile device to the various APs with no activities are running. Despite disabling the typical mobile applications, we found that tens of packets are transmitted to the application.

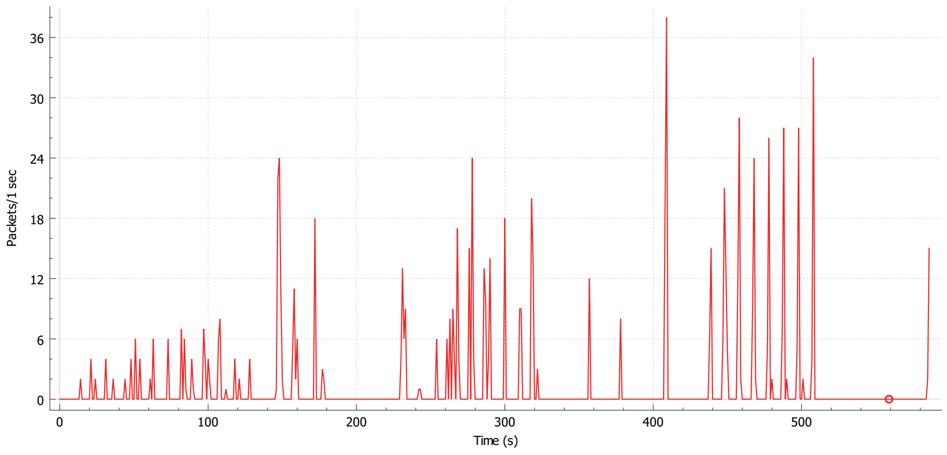


Figure 3.2: Scenario 1: probe requests

Figure 3.4 demonstrates the probe requests when the mobile device is moving while activities are running. The figure shows plenty of messages for association even during delivering normal traffic. Figure 3.5 depicts the normal uplink traffic from mobile with running activities. Again, we found hundreds of messages transmitted to the APs due to web browsing and video streaming.

Figure 3.6 shows the probe request when the mobile device is stationary and no activities are running. As expected, the number of messages is dropped thanks to keep the connection to a single AP. However, we note also the periodic behavior of the probe request in this scenario. As shown in the figure, the probe requests are frequently transmitted every approximately three minutes. Figure 3.7 depicts the normal uplink traffic from the mobile device when no activities are running and mobile device is stationary. Similarly, the number of messages is smaller than that in Figure 3.5.

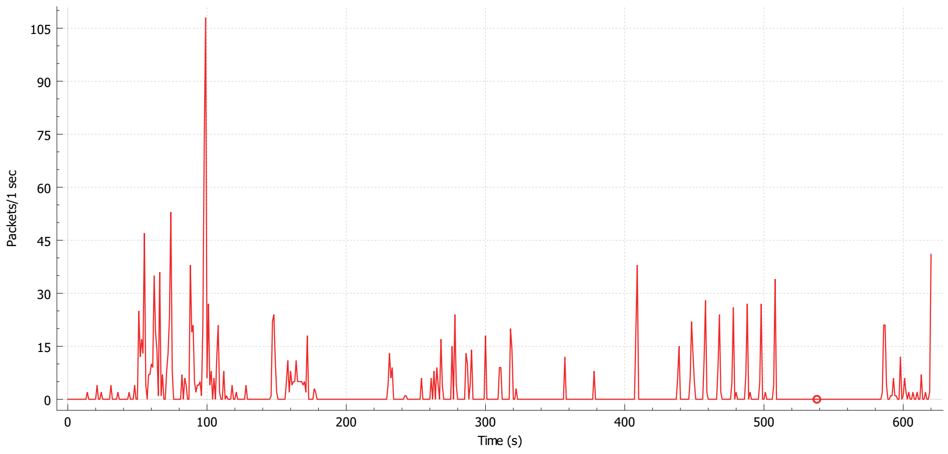


Figure 3.3: Scenario 1: normal uplink traffic

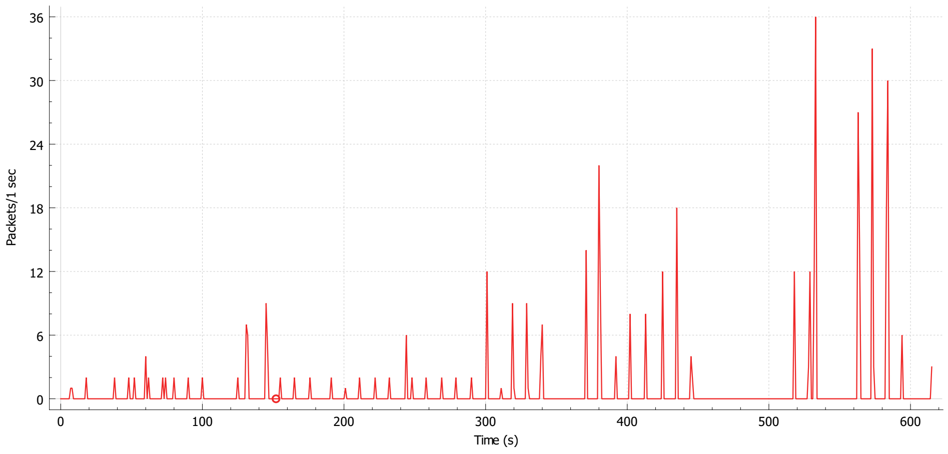


Figure 3.4: Scenario 2: probe requests

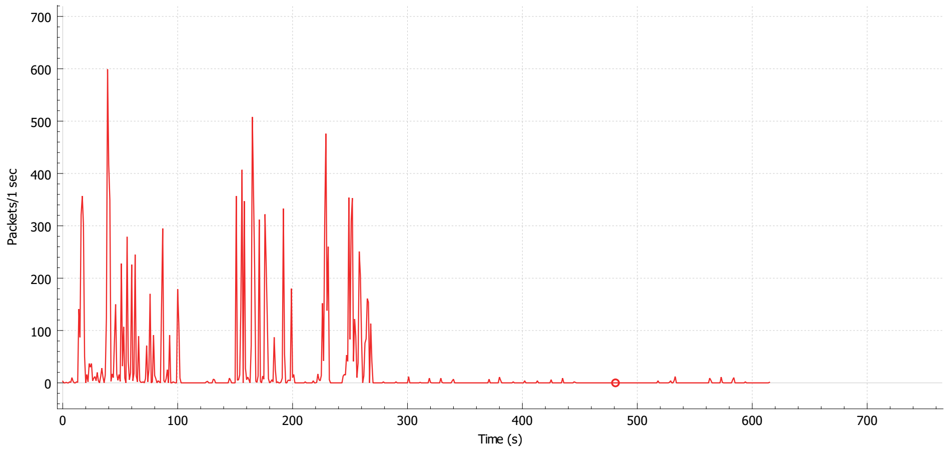


Figure 3.5: Scenario 2: normal uplink traffic

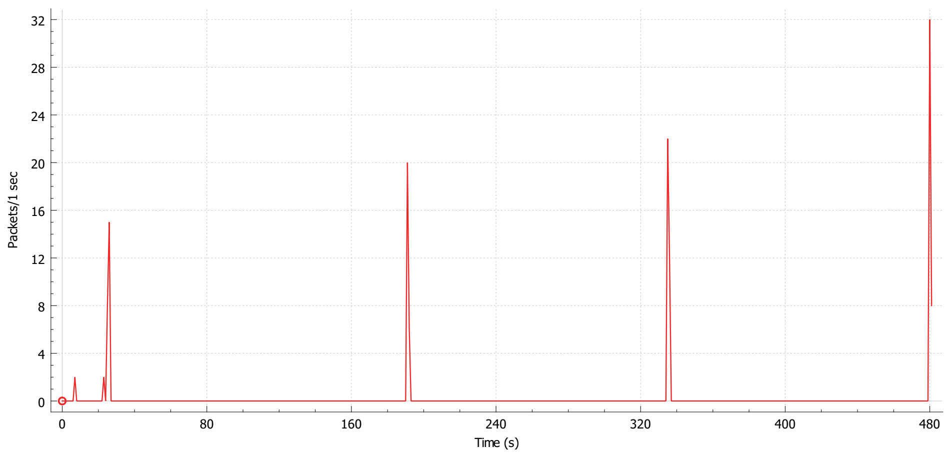


Figure 3.6: Scenario 3: probe requests

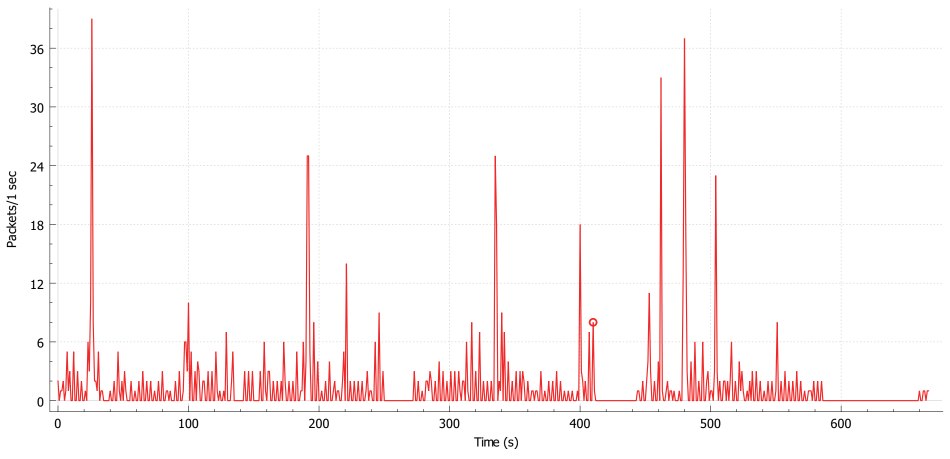


Figure 3.7: Scenario 3: normal uplink traffic

Figure 3.8 shows the probe requests when the mobile device is stationary and some activities are running. The figures show extreme reduction in the number of probe request due to being already connected to an AP. Alternatively, Figure 3.9 delineates the normal uplink traffic from the mobile device when some activities are running and the mobile device is stationary. We found also high traffic size due to uploading data related to the running mobile applications.

Figure 3.10 depicts the RTS messages that are sent from AP to the mobile device during mobility and running activities. As obvious in the figure, the peaks in the curve occur during running all activities simultaneously. Similarly, Figure 3.11 shows the RTS messages while the mobile device is fixed and no activities are running.

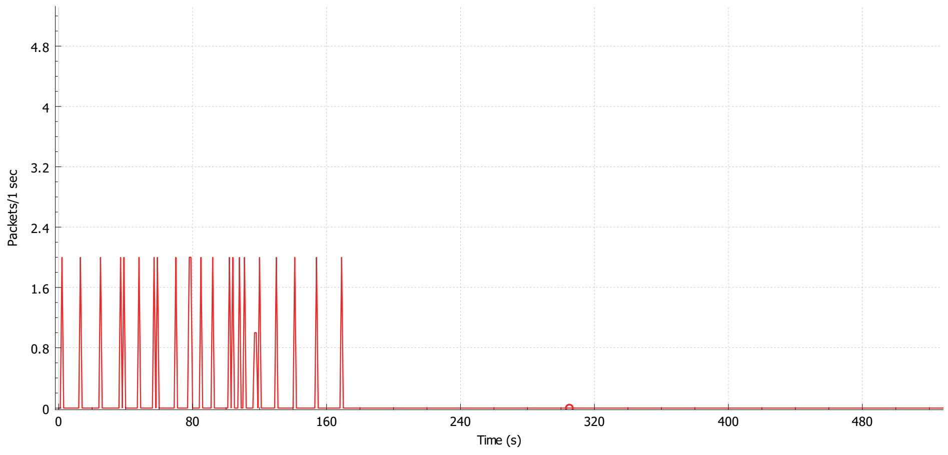


Figure 3.8: Scenario 4: probe requests

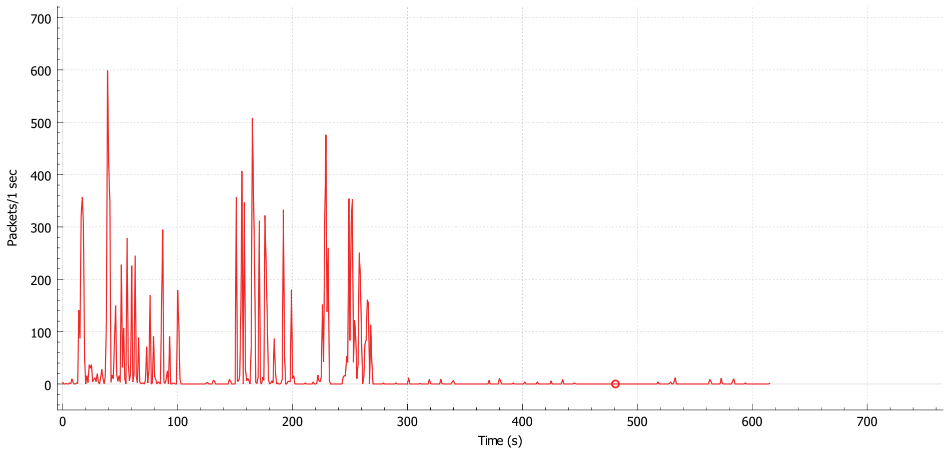


Figure 3.9: Scenario 4: normal uplink traffic

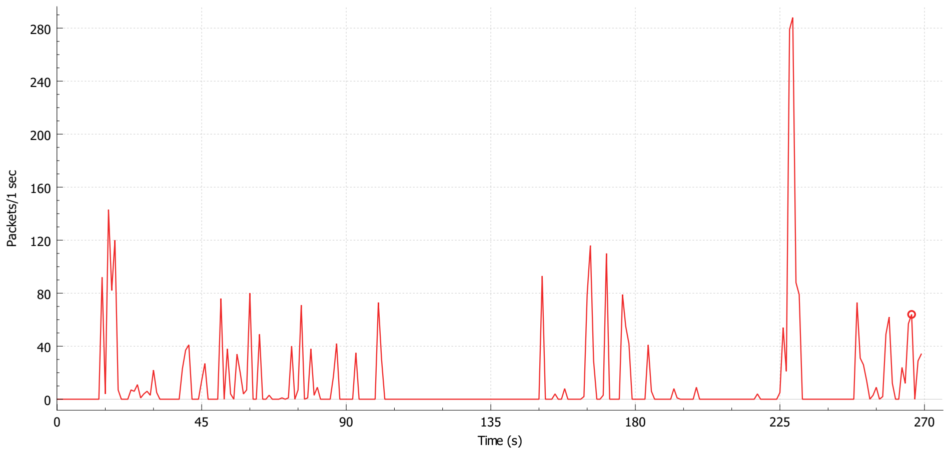


Figure 3.10: Scenario 2: RTS messages

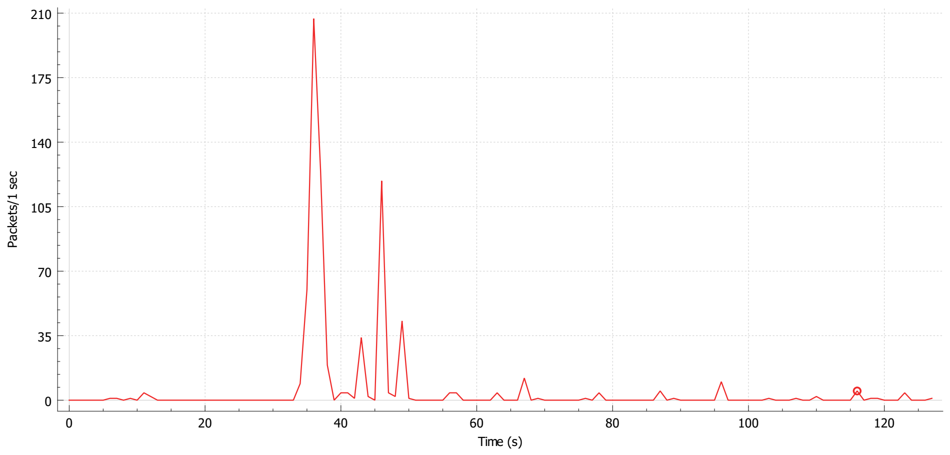


Figure 3.11: Scenario 4: RTS messages

3.2.3 Discussion

In general, the IEEE 802.11 control frame such as probe request are broad-casted from time to time as a bench of messages and it broadcasts on all channels. The amount of the probe request frames is varied based on the setting and the state of the mobile devices. The figures from the four experiments illustrate the amount of probe request that are transmitted by the mobile device to check the connectivity to the deployed APs. The traffic of probe request is increased if the user mobile moves between the APs and the client roaming mode are active as shown in Figure 3.2 and Figure 3.4.

Furthermore, the amount of normal uplink traffic that is sent from the mobile devices varied according to running activities on the mobile device. In the experiment, the normal traffic measured in both cases when the mobile device was running activities and without activities as depicted in the figures above. From the above figures and also by analyzing the traffic of random mobile devices which were existing in the building during running the experiment. The measurement results showed that in both cases when there were running activities or not, there is sufficient traffic which can be exploited to opportunistically monitor the occupancy of an area.

Important to mention that the obtained results of this study may be affected by several reasons. For instance, we have monitored solely one channel. Furthermore, the background activities running in the mobile devices can be different from a mobile device to another. Also, the background services and activities are different according to the OS, such as Android, IPhone and Symbian. For these reasons, we repeated the experiments three times. We found the average results show similar traffic size even when there are no activities running on the mobile devices.

To conclude, the normal traffic is shown to be sufficient to be utilized for developing an opportunistic occupancy monitoring method. There are many services and application that keep sending data when the mobile is not used. The result of the study shows the feasibility of localizing the mobile devices by exploiting the existing traffic. For example, as the dedicated results show that there is enough traffic that can be exploited to get a good accuracy of RSSI measurements by averaging the extracted value for the threshold duration such as 1 second. Therefore, the normal traffic can be captured from more than one anchor and then a localization method is applied to calculate the user position.

4 Monitoring Occupancy

After validating the existence of plenty of WiFi messages in different scenarios, we present, in this chapter, the proposed occupancy monitoring method for public sensing. First, an overview of the proposal method is described. Second, we explain the packets detection and data collection algorithms. Subsequently, we explain the estimation of coarse and fine grained locations of mobile devices in a certain building. Specifically, we strive to improve the accuracy of localization by applying different algorithms, including compressive sensing and the k-Nearest Neighbors algorithm.

4.1 Overview

The traffic analysis motivates us to propose an occupancy monitoring algorithm that exploits the already-existent traffic for monitoring the mobile device occupancy in certain geographic area. The primary goal of the approach is to overcome the energy consumption on the mobile device due to localization mobile during data collection process in public sensing applications. To this end, the burden of location estimation is moved from the mobile device to the server, as illustrated in Figure 4.1. As it can be seen in the figure, the proposed occupancy monitoring method is implemented on top of the basic sensing system and the localization process is entirely performed on the server.

According to the traffic analysis presented in Chapter 3 Referenceschapter:traffic, the monitoring occupancy method is divided into two scenarios, namely *fine-grained* and *coarse-grained* localization. The algorithm frequently checks if there are enough traffic that guarantees highly accurate RSSI measurement. In other words, the algorithm keeps observing the amount of collected RSSI measurements from the acquired data. If the collected amount during an interval is above a threshold D_{th}, which is calibrated during the offline phase, the first scenario is applied to estimate the user location. Otherwise, the second scenario is applied, when there is no enough traffic but the server still receives the RSSI only from the associated AP. This means that a single AP is used to estimate the current position. Given n access points deployed in the building, then the collected RSSI is expressed by $\psi_r = [RSSI_1, RSSI_i, \ldots, RSSI_n]$ where $RSSI_i$ is the average of RSSI that is collected by AP_i. The approach consists of two separately stages, including:

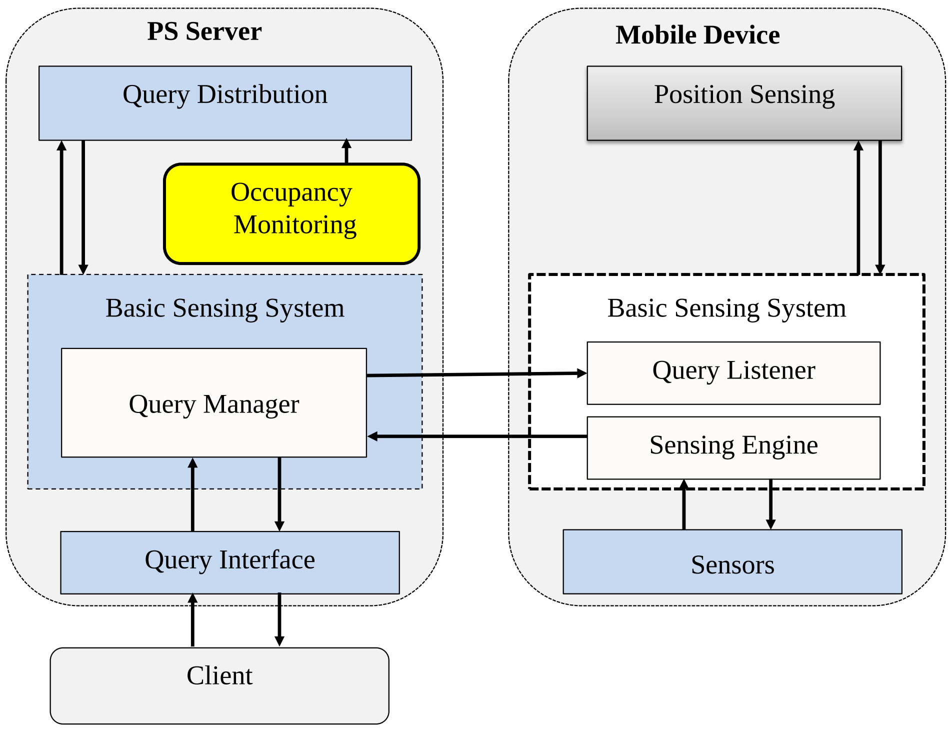


Figure 4.1: PS architecture with incorporating the occupancy monitoring method

(1) *detection and data collection* which runs at the AP level for monitoring the occupancy of the surroundings by mobile devices; and (2) *positioning estimation* which runs also on the server for calculating the device's location.

For better understanding the role of each algorithm, the approach is presented in hierarchy layer. In bottom layer where the detection and data collection are running, the monitoring occupancy is performed on each AP by detecting the mobile devices in the surround area of the AP and then collecting the RSSI measurement for each detected mobile device. In the upper layer, the RSSI measurement at the bottom layer are collected and then a localization method is applied for calculating the fine-grained position.

4.2 Detection & Data Collection

The algorithm – which runs on the AP – monitors the occupancy of an area by mobile devices where it checks whether the mobile devices are associated with the AP or not.

According to the connection type with the AP, the algorithm is designed to handle two types of mobile devices which are: (1) *associated* mobile device which are already in connection with an AP, and (2) *unassociated* mobile devices which are available in the coverage region of an AP.

Figure 4.2 demonstrates the diagram of the proposed algorithm. It mainly consists of three parts that includes the associated mobile device detection, the unassociated mobile device detection and the measurement pool update. In the sequel, the role of each part is described in more detail.

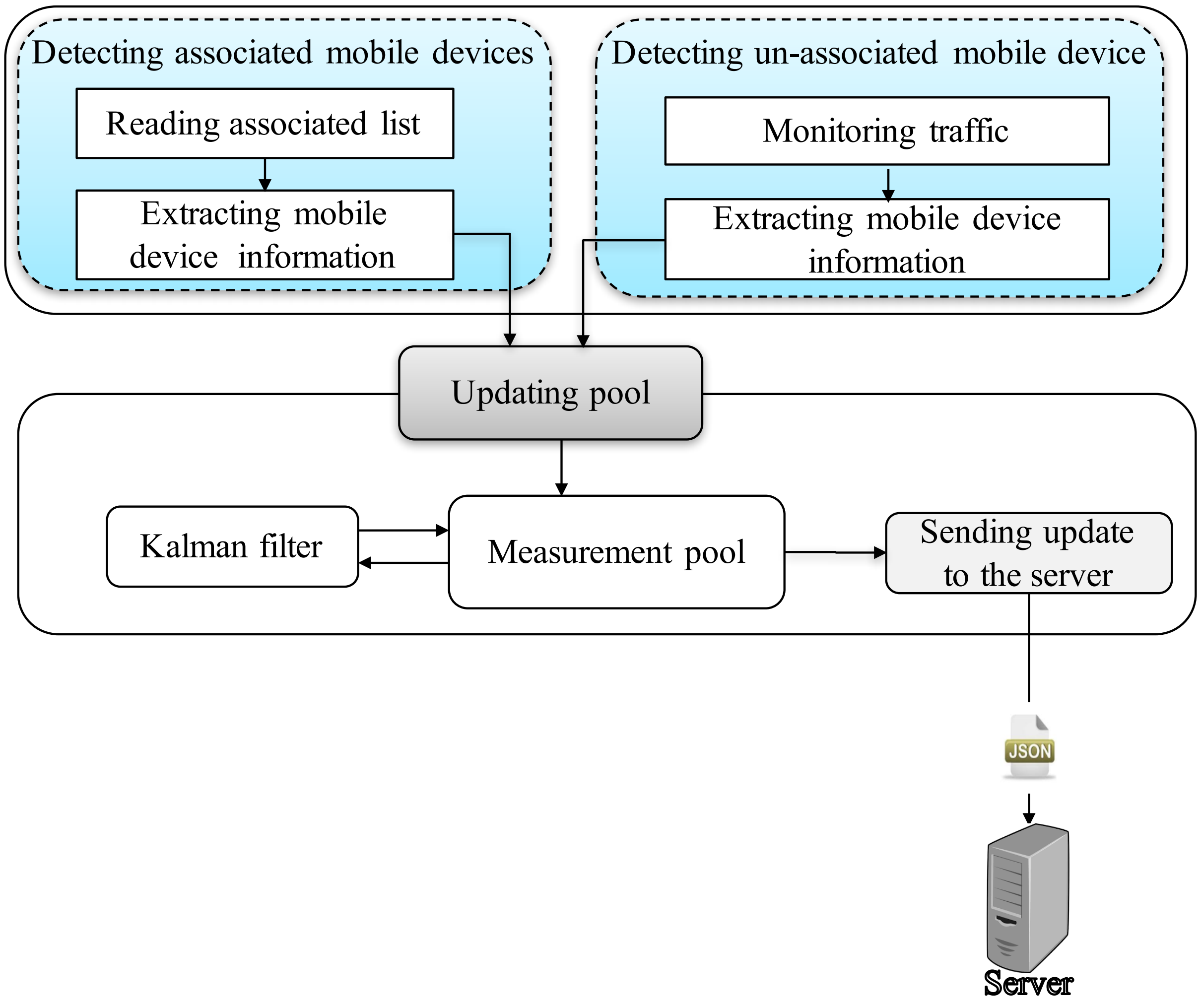

Detecting associated mobile devices
Reading associated list
Extracting mobile device information
Detecting un-associated mobile device
Monitoring traffic
Extracting mobile device information
Updating pool
Kalman filter
Measurement pool
Sending update to the server
JSON
Server

Figure 4.2: Detection of mobile devices and data collection

4.2.1 Associated Mobile Devices Detection

The role of this handler is to detect and gather the RSSI of the associated mobile devices with the AP by frequently reading the associated mobile device list on each AP where the information of Wi-Fi devices such as data rate, RSSI, mac address, and channel are found. The RSSI and MAC of the associated mobile devices are extracted by analyzing the content of associated list on the AP. Each detected mobile device is reported to the server by send an update message $m_{i,j} \in A_i$, where $m_{i,j}$ is a mobile device j that is detected by AP_i while A_i is the associated mobile devices list on AP_i. With only detecting the associated mobile devices, accuracy of the occupancy monitoring is in the space of the coverage area of the AP which is presented as circle. Thus, to improve the accuracy, the AP have to handle the unassociated mobile devices as well.

4.2.2 Unassociated Mobile Device Detection

In this section, the detection and data collection of unassociated mobile devices is described. The traffic analysis shows that there is enough normal uplink traffic and broadcast traffic from the mobile devices. This traffic can be exploited to detect and collect the RSSI of the unassociated mobile devices which can be seen by the AP. Thus, the detection of the unassociated mobile devices is achieved by dumping the entire flight packets in the surround area. Then, the dumped traffic is analyzed and filtered for extracting the MAC address and the corresponding RSSI value of each packet, as explained in detail in Chapter 5. The handler reports each detected mobile device to the server by sending update message $n_{i,j} \in U_i$, where $n_{i,j}$ is a mobile device j which unassociated mobile that is detected by AP_i. With knowing the RSSI measurement at more than two AP, the location of the occupant mobile device is estimated as coordinates locations in side the coverage area of the associated AP which is performed by the upper layer as discussed in the next sections

4.2.3 Measurement Pool Update

The last part aims to improve the quality of the collected data. The gathered data is preprocessed and then are sent to the server. For example, a noise filter, referred to as the Kalman filter, is applied. The Kalman filter improves the quality of measurements by filtering out the noise. It also tracks the changes of mobile devices based on the previous measurement. As discussed in Chapter 2, the location can be estimated as a point by using RSSI measurement value from one AP. To calculate the exact user position, the RSSI from more than two APs is required. To have better understanding, Figure 4.3

shows an example of how our approach can gather the RSSI for the mobile devices and sends them to the server.

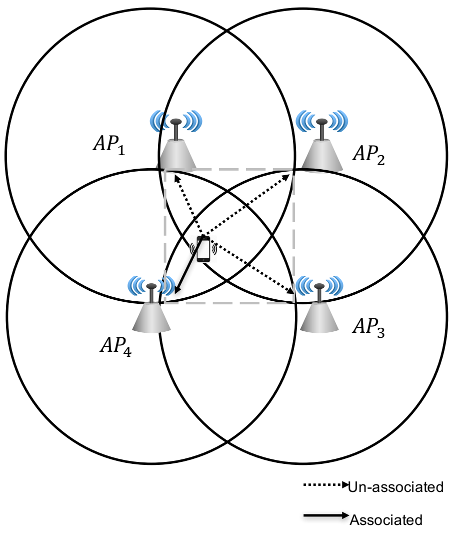


Figure 4.3: Example of the mobile devices detection

As it can be seen in the figure, each AP detects the mobile device and collects data about the mobile device in the range of their coverage which is represented as a circle. For instance, AP_1, AP_2, and AP_3 report the RSSI value of the mobile device T_t as unassociated device such as $n_{i,t}, i = \{1, 2, 3\}$. While AP_4 reports the RSSI of mobile devices as an associated device (e.g. AP_4 sends $m_{4,t}$) while $n_{i,t}$ and $m_{i,t}$ are the update message which hold the MAC address and the RSSI measurements. The server received all the update messages from the APs and then reconstructs them as a record which is known as runtime measurement $\psi_r = \{\psi_1, \psi_2, \psi_i, \cdots, \psi_n\}$, Where ψ_i is the RSSI that is reported by AP_i. In the next section, we will discuss the localization methods that use RSSI to calculate the user position.

4.3 Position Estimation

In this section, the positioning estimation algorithm is presented. This algorithm runs on the server for calculating the user location. To calculate the user position, in this thesis, the RSS-based *fingerprint* is applied which has accepted as an effective method in the indoor positioning comparing to the propagation modeling method. Figure 4.4 shows the architecture of the monitoring occupancy algorithm which consists of two phases. These phases comprises (1) an *offline* phase where the radio map is constructed and then is clustered into small clusters, and (2) an *online* phase where the data is gathered from multiple APs for calculating the exact position using the proposed localization method.

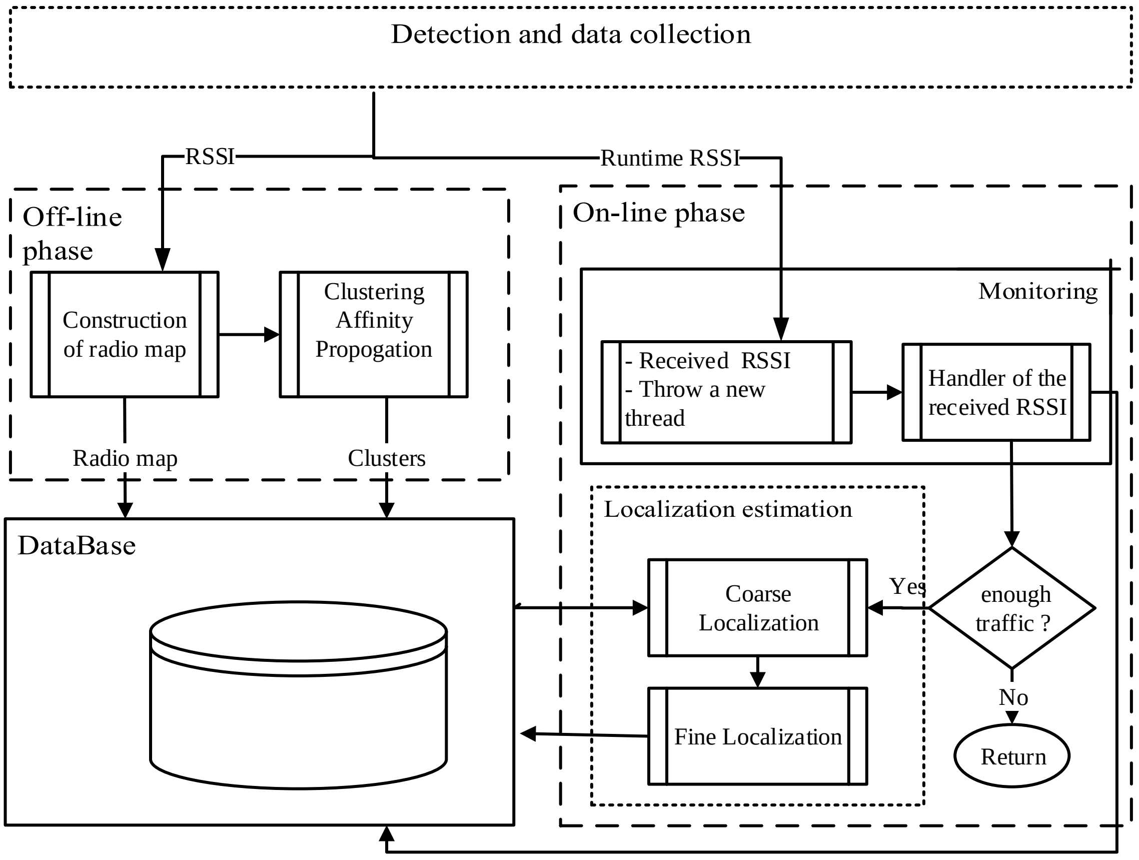


Figure 4.4: Architecture of the proposed occupancy monitoring method

4.3.1 Offline Phase

The aim of this phase is to construct the radio map and use it during the online phase. Figure 4.4 illustrates the offline phase which consists of two stages. These stages includes construction of the radio map and RSSI clustering using the *affinity propagation* algorithm for operating on the radio map in order to improve the accuracy by clustering the *reference points* (RPs) in the radio map based on the RSSI.

4.3.1.1 Radio Map Construction

In the offline phase, a radio map of the monitoring area is obtained. Figure 4.5 illustrates the diagram of radio map construction for our approach. The radio map of the indoor environment is constructed and stored in the database. First, the geographic area of the indoor environment is divided into a small area. Each small area has a RP which is identified by its coordinates (X, Y) as discussed in Chapter 2. The distance between each point is selected to be two meters. For example, if a coordinate location starts with (1,1), then it will increase to (1,3) and (1,5).

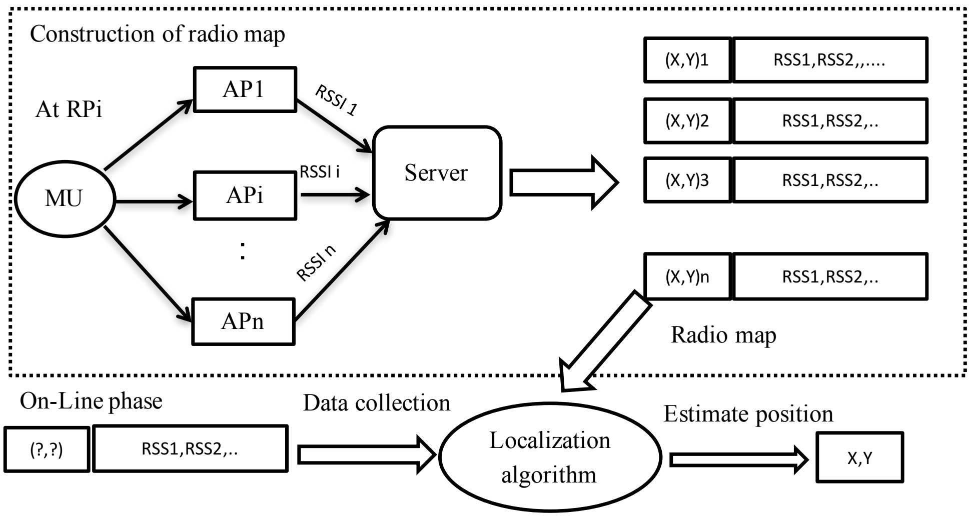


Figure 4.5: Radio map construction

In our approach, the APs measure the RSSI of mobile devices and send the measured RSSI to the server. For each $RP_{i,j}$, at the coordinates location (X_i, Y_j), the average of RSSI for each orientations is collected from the surround APs. Integrated radio and the orientation map has shown improving the accuracy compared to radio map with averaging the RSSI across different orientations. The integrated radio map takes into account the obstacles of the human body. The collected RSSI is constructed as a training record by adding information about the RP such as coordination location (X, Y), and the orientation. The collected average of RSSI has to be unique to characterise the RP in the radio map.

$$RP_i = X, Y, direction, \Psi_{i,k} i = 1, \ldots, N; k = 1, \ldots L \tag{4.1}$$

Where $\Psi_{i,k}$ is the average of RSSI at RP_i which is reported by AP_k; $k = \{1, \cdots, L\}$ and N is the number of RPs in the radio map, L is the number of APs in the training record.

The same procedures are repeated until all RPs in the radio map are visited. In the end of the training phase, the radio map is constructed as a set of training record such as

$$\Psi_{i,j} = \begin{bmatrix} \Psi_{0,0} & \Psi_{0,1} & \Psi_{0,j} & \dots & \Psi_{0,L} \\ \Psi_{1,0} & \Psi_{1,1} & \Psi_{1,j} & \dots & \Psi_{1,L} \\ \dots & \dots & \dots & \dots & \dots \\ \Psi_{N,0} & \Psi_{N,1} & \Psi_{N,j} & \dots & \Psi_{N,L} \end{bmatrix}. \tag{4.2}$$

The radio map is stored in the database and used for calculating the user location during the online phase.

4.3.1.2 Affinity Propagation

Affinity propagation is a clustering algorithm that was proposed in [FD07]. The algorithm is used for clustering the data points such as a set of a training data by exchanging the update messages between the data points. The aim of using Affinity propagation, in this approach, is to cluster the RPs in the radio map. This clustering improves the accuracy and reduces the number of RPs that will be used for calculating the user position during the online phase. In addition, using affinity propagation reduces the number of RSSI that is required during the online phase. Next, the procedure of the algorithm and how affinity propagation operates on radio map is described.

As an initialization, the constructed radio map ψ, in the above section, is first represented as a matrix of size $\psi \in \mathbb{R}^{N,L}$ where N is the number of RP in the radio map and L is the number of the APs in each RP. Second, the algorithm calculates the similarity between all RPs in the radio map and creates a square similarity matrix $S \in \mathbb{R}^{(N,N)}$ where N is the number of RP in the radio map.

$$S(i,k) = -||RP_i - RP_k||^2 \qquad \forall\ i,k \in \{1,2,\cdots,N\} \tag{4.3}$$

where

$$||RP_i - RP_k||^2 \tag{4.4}$$

The notation $||.||$ denotes the Euclidean distance which is used for measuring the similarity between the candidate exemplar and other RPs in the radio map. Third, the idea of affinity propagation is to label the RPs with a preference $P = s(i,i)$. The RPs with P are more likely to be selected as an exemplar. Thus, the algorithm requires a set of data points to be selected as center points for each generated clusters. So, if the center point is selected from the actual data points, the selected center is called “exemplar”. In this approach, the algorithm is initiated by assuming that all the RPs in radio map matrix as potential examples [TTZT13] which is done by initialing all RP with a preference value which is calculated by taking the median of similarity matrix S such as

$$P = s(i,i) = \underset{(i,j)\in 1,2,\ldots,N}{\text{median}} \{S(i,j)\} \tag{4.5}$$

Fourth, there are two update messages: (1) responsibility update message which is represented as matrix $R \in \mathbb{R}^{N,N}$ and (2) availability update message which is represented as matrix $A \in \mathbb{R}^{N,N}$. These messages follow update rules and recursively exchange between the RPs until the well-suited examples are selected as follows. The responsibility message $R(i,k)$ is transmitted by RP_i to each candidate potential exemplar RP_k. The responsibility message transmits the accumulative evidence for the RP_k to be selected as a fitness exemplar for RP_i with taking into account all the other potential exemplars. The responsibility message $R(i,k)$ is then calculated by following this update rule

$$R(i,k) = S(i,k) - \max_{k‘\neq k} \{A(i,k‘) + S(i,k‘)\} \tag{4.6}$$

where $A(i,k)$ is the availability update message of RP_k for RP_i. As explained later, $S(i,k)$ is the similarity matrix between the RP_i and exemplar RP_k. If the $i \neq k$, then the $R(i,k)$ is set to the input similarity between RP_i and RP_k minus the maximum similarity between the other RPs and exemplar RP_i pulse $R(i,k)$. With the first iteration, the initial value of $A(i,k)$ is equal to zero. In the later iteration, the availability message is decreased if the RP_i is assigned to other exemplars and may drop under zero. The negative $A(i,k)$ leads to remove the RP_k from the competence.

If $i = K$, the $R(i,i)$ is defined as self-responsibility that is set to the input preference P which is calculated by using Equation 4.5. The self-responsibility describes that RP_k is selected as well suited exemplar. The second message is the availability message $A(i,k)$ which is sent from each potential candidate exemplar RP_k to other RP_i. The availability message indicates how the exemplar RP_k is well-suited to be selected as exemplar for RP_i. Hence, The availability message is calculated by following this rule

$$A(i,k) = min\left\{0, R(k,k) + \sum_{i‘\neq(i,k)} max\{0, R(i‘,k)\}\right\} \tag{4.7}$$

The availability message $A(i,k)$ is set to the self-responsibility plus the sum of the positive responsibility update $R(i‘,k)$ that exemplar K receives from $RPi‘$. Similar to the self-responsibility, the availability message defines self-availability which reflects the accumulative evidence how well-suited RP_k to serve as exemplar for RP_i based on the positive responsibility that is sent by other RP_i. The self-availability is calculated by

$$A(k,k) = \sum \max_{i' \neq k} \{0, R(i', k)\} \tag{4.8}$$

The two messages recursively pass among the RPs and both mentioned update rules are followed until a suitable set of exemplars is found. In addition, Tian et al. [TTZT13] describe damping factor as another important factor. The damping factor is a number between 0 and 1. This factor is used for avoiding the numerical oscillations while updating the message above. So, the mentioned above messages are set to $(\lambda)*$ time the value and $(\lambda - 1)$ time the value from the previous iteration. The update is calculated, in this thesis, by using the previous update rule with the following that used in [TTZT13].

$$R_i = (1 - \lambda) R_i + \lambda R_{i-1} \tag{4.9}$$

$$A_i = (1 - \lambda) A_j + \lambda A_{j-1} \tag{4.10}$$

The update messages recursively are transmitted between the neighbor RPs until there is no change in the R_i and A_i or not exceeds 300 iterations according to Tian et al. [TTZT13]. In this thesis, the number of iteration is set to the maximum 300 iterations. Finally, after the iteration is done, the exemplars are selected by checking if the self-availability update message plus the self-responsibility are positive such that

$$a(i,i) + r(i,i) > 0. \tag{4.11}$$

Specifically, $RP_i \in H$ where H is a set of exemplar RP_i. Each exemplar is a center point for cluster C_i which holds the set of RPs. The RPs in the radio map is assign to the well-suited cluster by calculating the Euclidean distance between the RP and all exemplary $RP_i \in H$. Then, the RP is added to the cluster that its exemplar has a minimum distance. The output of this algorithm are set of exemplars and their corresponding cluster that are used during the online phase in coarse localization.

4.3.2 Online Phase

In this phase, the server collects the detected RSSI on each AP and then calculate the exact location. In this work, the positioning estimation algorithm initially defines two threshold values that include D_{th} which determines the number of RSSI which have been collected during the D_{th}^{period}. The selected threshold value have to guarantee enough accuracy of the RSSI. Hence, the algorithm frequently checks if the number of collected RSSI exceeds the threshold value, then the proposed localization method can be applied. To calculating the position for RSS-based fingerprint technique, Possible methods are classified into three approaches according to [TTZT13].

- machine learning approaches such as *K-Nearest Neighbor* (KNN) [Far11] (see the next section).
- probabilistic-based approaches such as Bayesian theory [SMRP] and kernel theory [KPV07].
- other approaches such as *Compressive Sensing* (CS) (see Section 4.3.2.2).

In this thesis, we consider both of the KNN and the CS methods, as explained in the next section.

4.3.2.1 K-Nearest Neighbor Method

The K-Nearest Neighbor method is used to estimate the user's location for the RSS-based fingerprint technique by matching the run-time RSSI vector and the RP in radio map. The KNN is an extension of the Nearest Neighbor (NN) algorithm where only one RP is used for estimating the position. Figure 4.6 illustrates the diagram of calculating the user position by using KNN algorithm. First, the distance between the run-time RSSI vector of the target mobile devices ψ_r and the RPs in the radio map is calculated by using a similarity method such as Euclidean distance.

$$RP = min||\psi_i - \psi_r|| \tag{4.12}$$

where ψ_i is RSSI vector at RP_i, ψ_r is the runtime RSSI vector. The term $||\psi_i - \psi_r||$ is the Euclidean distance which calculate by

$$d_i = \sqrt{\sum_{(j=0)}^{n} (\psi_{i,j} - \psi_{r,j})^2}. \tag{4.13}$$

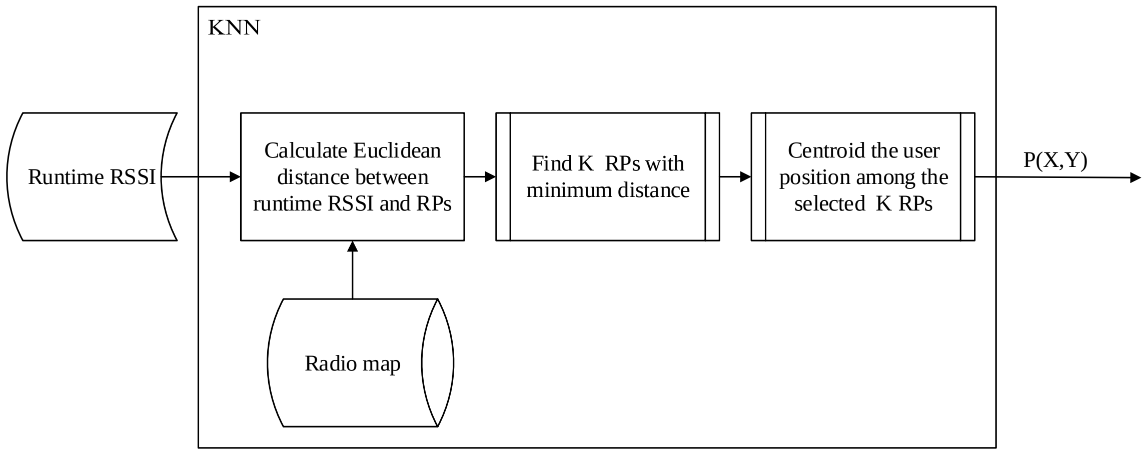

KNN
Runtime RSSI
Calculate Euclidean distance between runtime RSSI and RPs
Find K RPs with minimum distance
Centroid the user position among the selected K RPs
P(X,Y)
Radio map

Figure 4.6: The KNN algorithm

To improve the location-estimated accuracy, the KNN algorithm is proposed in which the accuracy is improved by selecting K RPs instead of a single RP and the user location is estimated by calculating the centroid location between the K RPs which have the minimum distance such that

$$(X,Y) = \frac{1}{K}\sum_{i=1}^{K} R(X_i, Y_i) \tag{4.14}$$

where R is a set of the K selected RPs that has the minimum distance with ψ_r. Second, K RPs with minimum distance is selected.

The key challenge of the KNN method is to select the well-suited K for the environment which achieves the highest accuracy. The well-suited K can be determined during the calibration phase as it depends on the environments. As investigated in [LSDR], the accuracy of KNN can improve by using the weight of RP instead of K which is implemented in the extension Weighted K-nearest neighbor (WKK) algorithm.

$$P = \frac{1}{\sum d}\sum_{i=1}^{K} R(X_i, Y_i) \tag{4.15}$$

Although the WKNN method improves the accuracy of the estimated location, but it is still not sufficient. Moreover, there are still several challenges and limitations to achieve high accuracy in the RSS-based fingerprint, including:

- a different RSS measurement for the same particular location, every time the measurement is taken.

- RSSI vectors sever from several noises such as missing data and change in the environment which highly affects the location-estimated accuracy.

Thus, To overcome this problem and improve the accuracy, a machine learning methods is applied in this thesis as discussed in the next section.

4.3.2.2 Compressive Sensing

Compressive Sensing (CS) is a signal processing technique that provides a novel framework to reconstruct sparse signals. The CS theory is used in different areas such as signal processing, image compression, and localization. In [FAVT10], the compressive sensing is used to improve the accuracy location-estimated by utilized the advantage that sparse location in space can be recover by using L1 minimization algorithm.

Figure 4.7 shows the CS localization algorithm has two phases including an offline and an online phase. The offline phase groups the RPs in the radio map into small clusters using affinity propagation algorithm. The online phase has two stages consisting of coarse localization and fine localization where the CS algorithm is applied for recovering the missing sparse data and estimating the position of the mobile users.

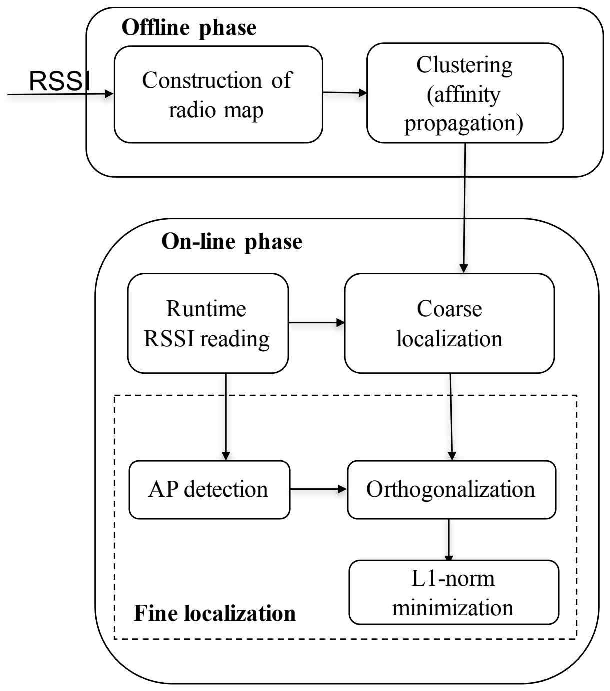


Figure 4.7: Compressive sensing diagram

During the online phase, the average of RSSI values for mobile devices at the unknown position are collected from each APs by the server as explained in section 4.2.

$$\psi_r = [\psi_{r1}, \psi_{ri}, \ldots, \psi_{rL}]^T \tag{4.16}$$

where ψ_{ri} is the average of RSSI on the AP_i.

Coarse Localization The primary goal of estimating a coarse localization is to reduce the set of RP that will be involved later in the fine localization. This leads to reduce the required memory to perform the fine localization. For instance. In [FAVT10], the coarse localization operates on mobile devices where the similarity between the set of exemplars H and run-time average RSS ψ_r are computed in order to select the close cluster. the selected cluster is send back to the server and then their corresponding clusters are downloaded from the server for calculating the user position during the fine localization as explained in the next section. In our approach, the coarse localization operates on the server. So, there is no need to wait for downloading the selected clusters.

Furthermore, the algorithm improves the accuracy by selecting more than one cluster by taking into account the case when the mobile device is close to the boundary of the clusters. The factor α is computed in order to select the well-suited number of clusters, and it is given by

$$\alpha = 0.95 \max_{l \in H} \left\{ S(r,n) - \min_{j \in H} \{S(r,j)\} \right\} \tag{4.17}$$

Now, the result of the coarse localization is donated as the following.

$$R = j : s(r,j) > \alpha, j \in H \tag{4.18}$$

$$[C = \bigcup_{jcs} C_j] \tag{4.19}$$

where C_j is the cluster that is corresponding to the exemplar J in S, C is a set of RP_i that their exemplars appear in S, the RPs in C is converted to matrix with size $N` * L$, $N`$ is the number of RP that appear in C $N` < N$, L is the total number of APs, the matrix is given by

$$\psi` = \psi_j \forall j \in C. \tag{4.20}$$

At the last step, the fine localization stage operates on the selected clusters are represented in $N` * L$ matrix.

Fine Localization The compressive sensing is used to recover the sparse signal. Fortunately, the localization has sparse nature where the RSS at RP is unique in the discrete spatial domain at a certain time [FAVT10]. To apply CS algorithm for the RSS-based fingerprinting, the collected data have to processed as follows. The term $theta$ is 1-spare vector of transformation coefficients, where $\theta \in \mathbb{R}^{N`}N`x1$ and the θ is equal to zero in all element except $\theta(n) = 1$, where n is the index of RP at which the mobile device is located via

$$\theta = [0, \ldots, 0, 1, 0, \ldots, 0]^T. \tag{4.21}$$

The online measurement is expressed as

$$y = \psi\Psi`\theta + \epsilon. \tag{4.22}$$

where $\Psi` \in \mathbb{R}^{N`*L`}$ is a matrix that is generated in Coarse localization, ψ is $M * L$ matrix which denotes the measurement matrix that indicates the selection AP as seen later, ϵ is the noise measurement.

$$y = \varphi\psi r, \qquad y \in \mathbb{R}^M \tag{4.23}$$

The sparse vector ϕ represents the AP selection matrix. The authors in [FAVT10] describe two approaches to select ϕ, namely the strongest APs and the Fish criterion method. The strongest APs approach is used in this thesis. So, the ϕ is set to $\phi(n) = 1$ where n is the index of selection AP such that

$$\phi m = [0, \ldots, 0, 1, 0, \ldots, 0], \quad m = \{1, 2, \cdots, M\}. \tag{4.24}$$

To apply the compressive sensing theory, ϕ and Ψ have to be incoherence and hold the Restricted Isometry Property [FAVT10]. Assuming the sensing matrix $R = \phi\Psi$, and evaluating $z = Ty$ where the measurement vector y is casted by using the orthogonal basis operator T. The notation $T = QR^{+}$ and T represent the orthogonal basis for R such that $Q = orth(R^T)^T$, $R`$ is the pseudo-inverse of matrix $R$, accordingly $TR`\{y\} = TR`R\theta + TR`$. The θ can be recovered totally from the observation vector z by solving the following problem using L1-minimization program such that: $\theta` = argmin||\theta||1, \theta \in \mathbb{R}^{N`}, z = Q\theta + \epsilon$, where ϵ is the white Gaussian noise. If the recovery vector in $\theta`$ contains only one, it means the estimated location is exactly that RP. Otherwise, the estimated position can be calculate by centroid the position among the selected RP in θ‘.

5 Implementation and Evaluation

In this chapter, we discuss the implementation details of the occupancy monitoring algorithms. First, the implementation of our proposed detection and data collection algorithm is described. Subsequently, the implementation of the position estimation algorithm, including both localization methods: the KNN method and the CS method is described. For evaluating the performance of applying KNN and CS, we built a testbed and performed a real experiment as discussed in Section 5.4.1.

5.1 Detection & Data Collection

In this section, we explain the implementation details of the detection and data collection algorithm. At the outset, we present a description of the adopted hardware. Subsequently, the implementation of the proposed algorithm is elaborated. All implementations, in this chapter, are written in C/C++.

5.1.1 Hardware Description

Most modern wireless network are managed by the wireless network controllers. In other words, all the APs in the wireless networks are connected together via a LAN network for management and administrative purposes, as illustrated in Figure 5.1. Thus, we assume that the APs are connected together through a LAN network. The employed APs is a commercial off-the-shelf hardware that can execute C/C++ code and support monitoring mode. Nowadays, a cheap AP can support open source firmwares (e.g. Tomato firmware, DD-WRT firmware and OpenWRT firmware). These firmwares are a lightweight embedded Linux distribution such as `LinksysWRT`. Beside, the main functionality of an AP is being an intermediary between the wireless network nodes. An AP is used as a monitoring sensor since it is already installed and covers all the area. The network traffic monitoring algorithm – which runs on each AP to detects and collect RSSI in a surrounding area – is explained in the next section.

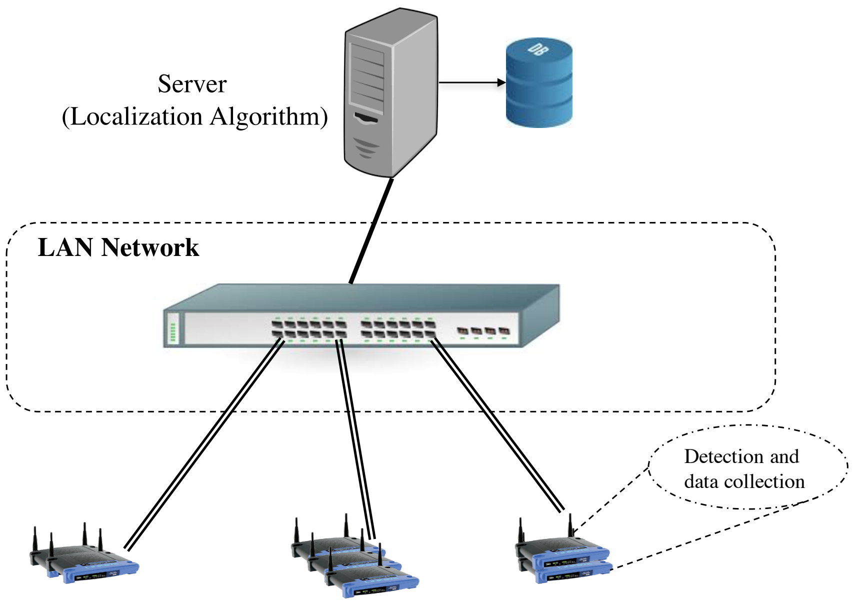


Figure 5.1: Detection and data collection hardware

5.1.2 Software Description

In chapter 4, the core idea behind our detection and data collection algorithm is discussed. In this section, we describe the implementation details of this aforementioned algorithm. The algorithm has been implemented to detect two types of mobile devices and then collect the data for each detected mobile devices. Figure 5.2 describes the data flow of the detection and data collection algorithm. (1) In the main method,the memory is initialed for creating the measurement pool.Next, tow new threads are thrown which are associatedList handler and un-associatedlist handler.(2) In the associatedList handler,the associatedList is read and the extracted RSSI and MAC are send to the measurements update pool as describe in Algorithm 5.2. (3)In the unassocaitedList handler, the *Packet Capture (*PCAP*)* analysis handler is used for capturing and analyzing the dumped traffic to extract the MAC and RSSI of each dumped packet and send them to measurements update pool as described in Algorithm 5.3.(4) The last process is measurement pool update process which receives the MAC and RSSI form both handlers for updating the measurement pool .It also filter the collected data using Kalman filter, average the RSSI and format the update as *JavaScript Object Notation (*JSON*)* for sending them to the server as described in Algorithm 5.4.

Algorithm 5.1 describes the main procedure of the detection and data collection algorithm where the initialization of algorithm is done and both handlers are started. The pseudo code starts by creating a measurement pool in which the set of space memories are allocated to hold the extract RSSI and MAC for both detected associated mobile

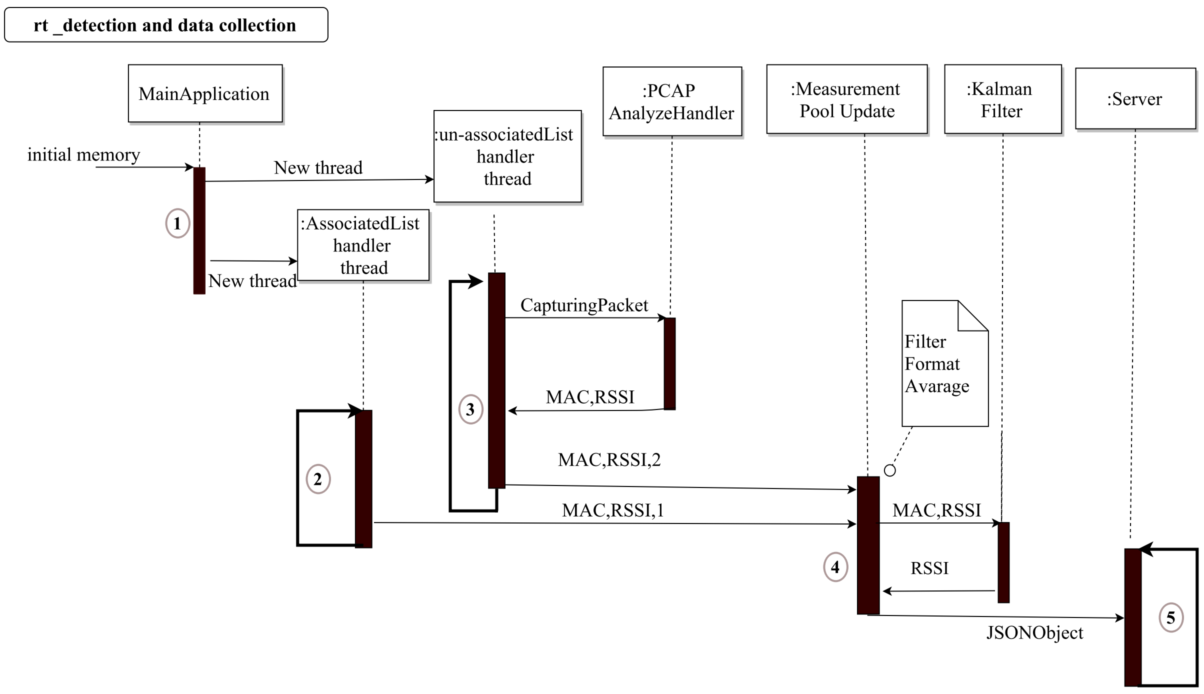


Figure 5.2: Data flow diagram of detection and data collection

devices and unassociated mobile devices. Then, the UDP port is created to connect to the server (l. 3). The two handlers of detecting associated mobile devices and unassociated mobile devices are run in parallel by throwing two new threads (l. 4-5).

Algorithmus 5.1 Detection and collection data algorithm

```
1: procedure MAINPROCEDUR($serverAddr$)
2:     $MeasurementPool \leftarrow allocateMeasurmentPool$
3:     $P \leftarrow createUDPport(serverAddr)$
4:     $startassociationlisthandler()$
5:     $startun - associationlisthandler()$
6: end procedure
```

To detect and collect the data for each associated mobile device, the algorithm reads the associated mobile devices list because most IEEE802.11 APs support accessing the associated devices list. The devices list contains all needed information about the associated mobile devices such as the MAC address, the corresponding RSSI as well as other information (e.g. data rate, and channel). For example, The `Broadcom` wireless card support `wi` tool that we used for dumping the associated devices list.

In a similar manner, Algorithm 5.2 describes the pseudo code of the association list handler. The associated mobile device list on AP is periodically read (l. 5) . Afterward, the RSSI and the MAC address are extracted from the dumped list (l. 6–7). Next, the extracted information (i.e. RSSI and MAC address) are passed to the measurement pool by calling *updatePool* procedure (l. 6) which manages the updates of the measurement pool, as discussed later.

Algorithmus 5.2 Handling *associated* mobile devices

```
1:  procedure ASSOCAITIONLISTHANDLER
2:      $assocList \leftarrow null$
3:      while (true ) do
4:          $List \leftarrow readassociatedlist()$
5:          for (line in list) do
6:              if (line contain MAC) then
7:                  $MAC \leftarrow extractMAC(line)$
8:              end if
9:              if (line contain RSSI) then
10:                 $\text{RSSI} \leftarrow extract\text{RSSI}(line)$
11:                 if $(MAC! = nulland\text{RSSI}! = null)$ then
12:                     $updatePool(MAC, \text{RSSI}, ass_type = 1)($
13:                 end if)
14:             end if
15:         end for
16:     end while
17: end procedure
```

The second handler is implemented to handle the unassociated mobile devices. By default, the existing hardware does not support the direct detection of and reading the RSSI of the unassociated mobile devices. Thus, the network traffic is dumped in order to analyze and extract the interested information of each mobile device. The RSSI of the detected value is placed in the *Radio Tap* header which exists in the low level layer of the MAC layer. In order to capture this header, a new wireless network interface is created and is configured to work in the monitoring mode and using `PCAP` which is an open source project that provides an API for low-level network monitoring. The `pcap` is implemented in the `libpcap` library in the Unix system which is an open source project [JLM16]. Similarly, Windows OS implements a port of `libpcap` which is known as `Winpacap`. The `libpcap` provides a framework for capturing, filtering and analyzing. The `libpcap` supports two ways of capturing and analyzing. First, the network traffic is captured and is saved into a file in `pcap` format for later filtering and analysis. Second, the `libpcap` supports an online capturing and analyzing which provides a packet handler for each captured packet.

Algorithm 5.3 describes the pseudo code that implements the unassociated mobile devices handler. The algorithm starts by creating a new wireless interface and monitoring mode configuration (l. 3-4). Next, an interface of `libpcap` is created (l. 5). Afterward, a network filter is applied on the created `libpcap` interface to avoid some unwanted traffic such as the outbound traffic and the Beacon frames (l. 6). After compiling and applying the setting of the `libpcap` interface, the capturing operation is started. Furthermore, the `libpcpa` library supports a packet handler that is invoked for each captured packet (l. 7-12). In the packet handler, the RSSI and MAC address of the sender are extracted by analyzing the *radio tap* header using radio tap API analysis library implemented by [Ber16] and the *updatpool* method is called to update the measurement pool with the extracted RSSI and the MAC address.

Algorithmus 5.3 Handling *unassociated* mobile devices

```
1: procedure UNASSOCAITIONLISTHANDLER
2:     $unassocList \leftarrow null$
3:     $monitoringInterface \leftarrow createwirelessnetworkinterface$
4:     $configure(monitoringInterface)$
5:     $PCAP \leftarrow createPCAPInterface()$
6:     $Settingfilter()$
7:     while ( true ) do
8:         $packet \leftarrow waitingforpacket$
9:         if (packet not empty) then P(C)APHandler($packet$)
10:        end if
11:    end while
12: end procedure
```

The last part is the *updatepool* procedure which is responsible for managing the measurement pool, as described in Algorithm 5.4. The pseudo code shows that the procedure checks whether the received MAC address already exists and the time since last seen of this MAC does not exceed the threshold time D_{th} (last seen). A Kalman filter is applied to filter the noise in the RSSI value based on the previous measurement value. The threshold D_{th} (last seen) should be a small interval. In addition, in order to reduce the redundancy update to the server, the algorithm averages the RSSI for a threshold number of measurement value that is specified by D_{th} before sending to the server. The measurement data is reconstruct as `JSON` object expressed as

$Data : [MACofAP :< mac_{AP} >, MACofmobile :< mac_{mu} >, rssi :<>]$.

For evaluating the performance of the KNN-based and CS-based localization methods, the proposed occupancy monitoring algorithm is implemented, as illustrated in Figure 4. First, the implementation of offline phase consists of two stages: (1) construction of a

Algorithmus 5.4 Measurement Pool Update

```
1: procedure UPDATEPOOL($MAC$, RSSI, $assType$)
2:     if ($MAC \in assListorMAC \in assList$) then
3:         if ($LastSeentime - currentTime < D_{thr_{time}}$) then
4:             RSSI $\leftarrow KalmanFilter(MAC,$ RSSI$)$
5:         end if
6:     end if
7:     $measurmentpool \leftarrow$ RSSI, $MAC$ u(p)dateLastseen(currenttime,MAC)
8:     if ($Lastupdate - currentTime > D_{thr^{update}}$ and $assType == 1$) then
9:         RSSI $\leftarrow getaverageof$RSSI$(MAC)$
10:        Else
11:        $return$
12:    end if
13:    $JsonObj \leftarrow convertToJSON(MAC, avr_rssi, assType)$
14:    $ServerPort \leftarrow sendUpdate(JsonObj)$
15: end procedure
```

radio map – which is discussed in Section 5.4.1.1 and the affinity propagation algorithm which is described in Algorithm 5.5. In the second part, implementation of the online phase is described. Below, the monitoring and initialization algorithm is introduced. subsequently, both of CS and KNN algorithms are explained in more detail.

5.2 Occupancy Monitoring: Offline Phase

In this section, the implementation of the affinity propagation algorithm is described which is used for clustering the RPs in the radio map. This clustering improves the accuracy by using CS and also reduces the number of the required APs during the online phase [FAVT10]. The second part of this section describes both KNN and CS localization methods.

5.2.1 Affinity Propagation

The implementation of affinity propagation is described in Algorithm 5.7 which is based on the approaches in [FAVT09; TTZT13]. The algorithm starts by initiating the parameters such as the matrix A holds the availability message, matrix R holds the responsibility message and matrix S of size $N * N - N$ holds the similarity between the RPs in the radio map; where N is the number of RPs in the radio map. Next, the distance or similarity between the RPs is calculated by calling *calcSimilarity* method in which the Euclidean distance is used for calculating the similarities between the RPs (l. 9).

Algorithmus 5.5 Affinity propagation algorithm

```
1: procedure AffinityPropogation($radiomap, \lambda, iteration$)
2:     $\Psi \leftarrow ConverttoMatrix(radiomap)$
3:     $N \leftarrow sizeof(\Psi)$
4:     if ($\lambda > 0.5 and \lambda 1.0$) then
5:         $return$
6:     end if
7:     $A[N][N] \leftarrow 0$
8:     $R[N][N] \leftarrow 0$
9:     $S[N][N] \leftarrow Calc_similarity(\Psi)$
10:    $P \leftarrow media(S)$
11:    for ($inti = 0; i < N; i++$) do
12:        $R[i][i] \leftarrow P$
13:    end for
14:    while ($iteration > 0$) do
15:        $R \leftarrow responsibilityUpdate(S, A)$
16:        $A \leftarrow availibiltyupdate(S, R)$
17:    end while
18:    for ($inti = 0; i < N; i++$) do
19:        if ($A[i][i] + R[i][i] > 0$) then
20:            $Exemplars \leftarrow i$
21:        end if
22:    end for
23:    for (int i=0;i<sizeof(Exemplar);i++) do
24:        $Cluster_i \leftarrow getCluster(i)$
25:    end for
26:    return <Exemplar,Cluster>
27: end procedure
```

To define the number of exemplars, the algorithm is initiated by calculating the median of matrix S (l. 10) and also by using a predefined parameter λ (l. 4). Then, clustering is performed by exchanging both update messages R and A between the RPs in order to select the well-suited exemplars to be as a center point for the generated cluster (l. 14-17), where The number of iterations is given, as given in [**tian2013fingerprin**]. The *responsibilityUpdate* method (l. 15) uses the responsibility update rule and also *availabilityUpdate* method uses the availability update rule, as discussed in chapter 4. After finishing the iteration, the well-suited exemplar is determined in (l. 18-17) by using $A[i][i] + R[i][i] > 0$. Finally, to select the RP for each generated cluster, *getCluster* method is used which selects the RPs with a minimum distance to the exemplar taking into account the distance to the other exemplars. The discussed algorithms in this chapter has been implemented in java.

5.3 Occupancy Monitoring: Online Phase

Algorithm 5.6 shows the pseudo code of the server listening on the UDP port and waiting for an update that is coming from the APs (l. 7). Then, a new thread is thrown for each incoming update(l. 9). The thread runs a handler that will be described by Algorithm 5.7.

Algorithmus 5.6 Initialisation of the localization process

```
1:  procedure MAINMONITORING
2:      initial prameters()
3:      $P \leftarrow CreateUDPport(serverport)$
4:      $db \leftarrow ConnectToDataBase()$
5:      while ( true ) do
6:          $JsonObject \leftarrow WaitForPacketFromAcessPoints$
7:          if (JsonObject is valid) then
8:              throw new $monitoringOccupancyHandler(jsoneObject, db)$
9:          end if
10:     end while
11: end procedure
```

Algorithm 5.7 shows that each received update is stored in the database after adding the time stamp. Then, it checks if there is enough measurement update. For example, the algorithm retrieves the last received update during the last one. Second, if the number of RSSI measurements – that is reported by the unassociated AP – is above a threshold value D_n, then the D_n is calibrated during the offline phase to guarantee enough RSSI and consequently accurate localization. If so, the average of the RSSI values is taken and the localization methods are called.

5.3.1 KNN Implementation

In chapter 4, the calculation of the user position by using KNN algorithm is discussed. In this section, Algorithm 5.8 gives the pseudo code which describes the implementation of the KNN method. The algorithm starts by retrieving the run-time RSSI for the MAC address to ψ_r and radio map Ψ from the database (l. 3-4). The Euclidean distance

Algorithmus 5.7 Monitoring occupancy

```
1: procedure MONITORINGHANDLER(jsoneObject,db)
2:     $R \leftarrow createmeasurementRecord(JSONObject, timestamp)$
3:     $db \leftarrow storeIntoDatabase(R)$
4:     if (time to update) then
5:         $M \leftarrow getlastupdatinlastsecond(MAC)$
6:         if ($numberOf$RSSI $> D_n$) then S(t)artLocalizationProcess(M)
7:         end if
8:     end if
9: end procedure
```

between the ψ_r and each RP_i in the radio map is calculated by using calling the *calcDistance* function (l. 8-10). By sorting the calculated distance and get the K first minimum RP (l. 6) and then calculate the centroid among them to find P (l. 9).

Algorithmus 5.8 KNN algorithm

```
1: procedure KNN($Psi, \psi_r$)
2:     if ($\psi_r isempty$) then
3:         $return$
4:     end if
5:     for each($RP_i \in \Psi$) do
6:         $S_i \leftarrow CalcSimiliarity(RP_i, \Psi_r)$
7:     end for
8:     $sort(S_i, \Psi)$
9:     $RPs \leftarrow selectFirstKminimumDistance(K, S_i, \Psi)$
10:    $P = CalcCentrinoposition(RPs)$
11:    return $P$
12: end procedure
```

5.3.2 CS Implementation

To improve the localization accuracy, the CS method is implemented. The implementation of this method has three separately stages. First, offline phase in which the radio map has to be clustered by using Affinity propagation algorithm. The second part is the online phase which consists of two stages, namely *coarse* localization and *fine* localization.

5.3.2.1 Coarse Localization

The aim of using coarse localization is to reduce the RPs that will be used in the fine localization to calculate the user position. Algorithm 5.9 gives a simple pseudo code that describes how we implement the coarse localization based on the approach in [FAVT10]. First, the similarly between the run-time measurement ψ_r and all exemplars is calculated by using the Euclidean distance (l. 2). The number of clusters significantly impacts the accuracy of the estimated position in the fine localization, as investigated later. Thus, to select the number of clusters that has the minimum distance, λ is calculated (l. 3), based on the calculated similarity in (l. 2). In order to select the well-suited number of clusters, λ is calculated (l. 3). Finally, the selected clusters are passed to fine localization stage to calculate the user position, as described in the next subsection.

Algorithmus 5.9 Coarse localization algorithm

```
 1: procedure COARSELOCALIZATION($(Exemplars, Clusters, \psi_r)$)
 2:     $S \leftarrow Calc_similarity(Exemplars, \Psi_r)$
 3:     $\lambda \leftarrow calcLamda(S)$
 4:     for (int i=0;i<sizeof(Exemplar);i++) do
 5:         if (S[Exampler[i]]>$\lambda$) then
 6:             $R \leftarrow Exampler[i]$
 7:         end if
 8:     end for
 9:     for each(i in R) do
10:         $RPs \leftarrow getRPInCluster(i)$
11:     end for
12:     $\Psi_{N,M} \leftarrow ConvertRP_iTomatrixMXN(RPs)$ F(i)neLocalization($\Psi_{N,M}, \psi_r$)
13:     return $\Psi_{N,M}$
14: end procedure
```

5.3.2.2 Fine Localization

As explained in chapter 4, this stage is implemented to recover the sparse location by applying the CS theory. During the performance evaluation, Algorithm 5.10 has been implemented in Matlab, based on the approach presented in [FAVT10]. The algorithm starts by initiating the parameters θ which holds the recovery vector. The sensing matrix Φ is determined by selecting M APs with strongest signal to perform the sparse recovery

(l. 5). An orthogonalization step – expressed in Equation 5.1 is required to provide the incoherency condition stated by the CS theory.

$$\theta^{'} = argmin\|\theta\|_1, Z = Q\theta + \varepsilon \tag{5.1}$$

Algorithmus 5.10 Fine localization algorithm for calculate the user localization

```
1: procedure FineLocalization(($L, M, \Psi^{N,L}, \psi_r$))
2:     $\Psi \leftarrow \Psi^T$
3:     $N \leftarrow Sizeof(\Psi)$
4:     $\theta[N] \leftarrow 0$
5:     $\Phi^{MxL} \leftarrow selectionAPs(\psi_r, M, L)$
6:     $y^{Mx1} \leftarrow \Phi\psi_r$
           // %comment: Apply orthognalization for recover the sparse localization%
7:     $R^{MxN} \leftarrow \Phi\Psi$
8:     $Q \leftarrow orth(R^T)^T$
9:     $R^{\dagger} \leftarrow psudoInverse(R)$
10:    $T \leftarrow QR^{\dagger}$
11:    $Z \leftarrow Ty$
12:    $\varepsilon = QZ$
13:    $\theta^{'} \leftarrow argmin\|\theta\|_1, Z = Q\theta + \varepsilon$
14:    if ($\theta^{'} hasonlyoneitemequalto1$) then
15:        $P \leftarrow \theta^{'}(n), nistheindexwhere\theta^{'} equalto1$
16:        return $P$
17:    end if
18:    $\lambda \leftarrow calclamda(\theta^{'})$
19:    $P \leftarrow calcCentrinoPosition(\theta^{'}, lambda)$
20:    return $P$
21: end procedure
```

We used L1-minimization code "$l1eq_pd$" (l. 5) which is proposed in [Pey13]. The recovered vector may hold more than a single one. In this case, the user location is calculated by determining the centroid among the λ selected RP. In our implementation, the λ has been selected as the minimum positive value in the θ

5.4 Performance Evaluation

To evaluate the performance of our approach, a testbed is built for deploying our implementation and carrying out an experimental study where the real radio map of the environment is obtained. Then set of test points are collected. Afterward, the performance of both KNN and CS algorithms are evaluated in terms of the localization error.

5.4.1 Localization Testbed

The goal of the experiment is to obtain the real radio map from a certain geographic area that reflects a typical setting of indoor environment and then collect set of test points in order to study the performance of KNN and CS. In the following section, the hardware and the software of the testbed are presented before the construction of radio map is discussed. Figure 5.3 shows that the testbed consists of Five APs (Linksys WRT 54GL) were connected together with a laptop as a server; via Ethernet LAN switch 8 ports. the mobile device was used is IPhone 4. On the AP side, the implemented detection and data collection algorithm is run on each APs. On the server side, we implemented a tool for constructing the radio map.

Each AP is configured for running the software. For example, we use the `OpenWRT` as a 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`. Be caution that the version of those packages have to be compatible with that are used to compile the code. As illustrated in Figure 5.3, the testbed has been constructed in 13m x 10m area of two student rooms and the cross hallway (second floor, Stuttgart University, Building 38).

5.4.1.1 Radio Map Construction

To construct the radio map for the monitoring area where the experiments have been performed, First, the area has been divided and a set of reference points (RP) were placed as shown in Figure 5.3. The distance between the RPs were two meters. Then, the software on both the APs and the server were run. The mobile device was carried by a volunteer where he was moved from a RP to another RP. A *fingerprint* is the average of RSSI of the mobile device on each surrounding AP that are collected by the server while the mobile device was associated with each AP. The radio map are generated over 35 RPs and cross the different orientations represented by the main four angles (0, 90, 180, 270).

Now, we explain how to construct the fingerprint for each RP. First, the *radio map constructor* tool asks to type the coordinates location (X_i, Y_j) and then the RSSI values

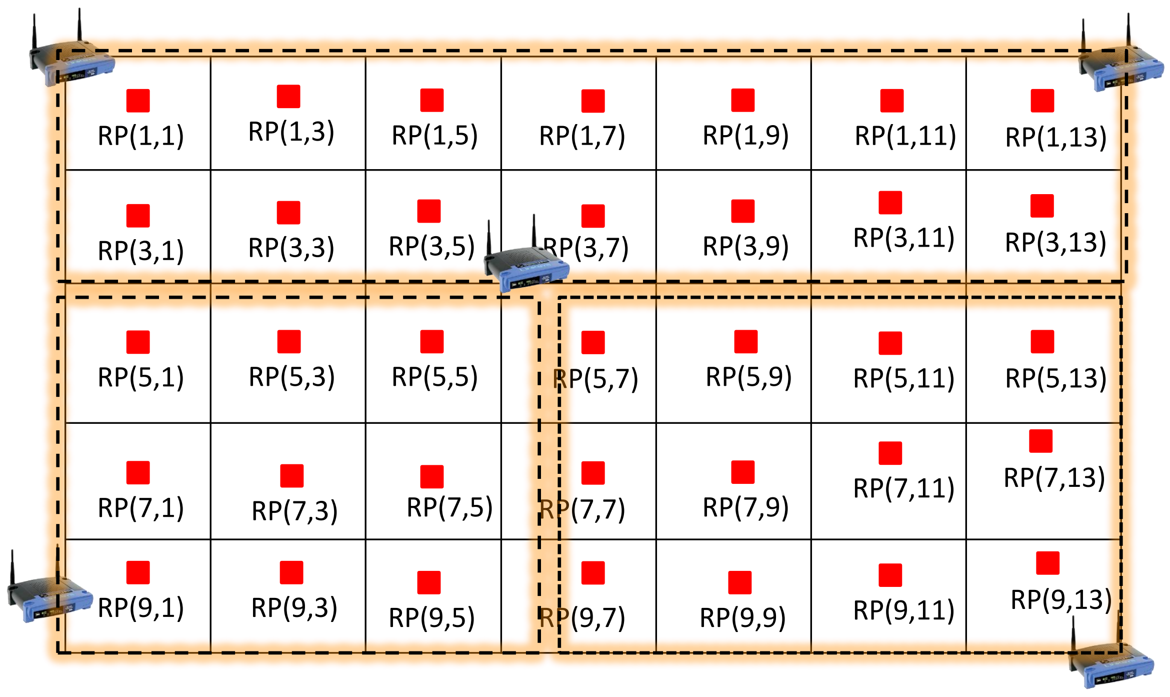
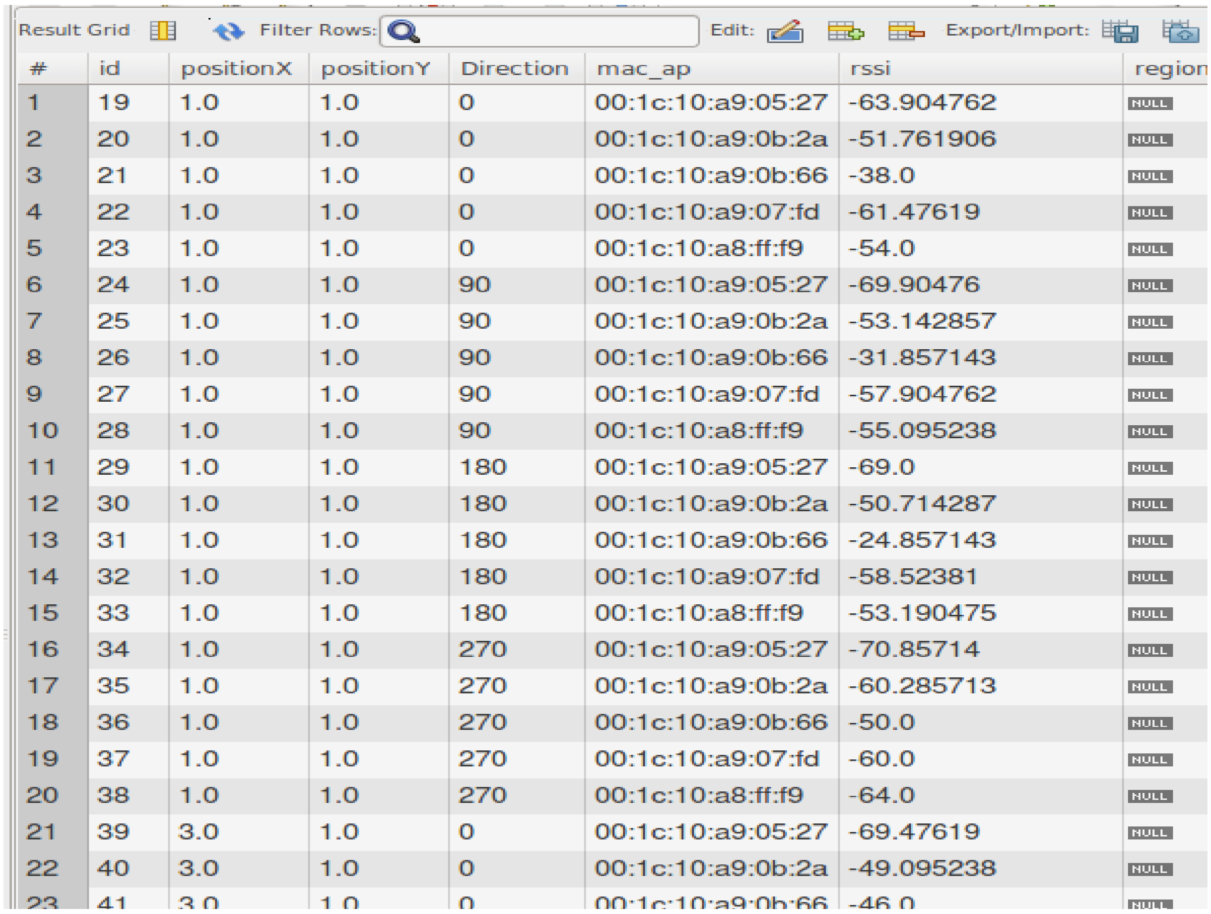


Figure 5.3: Testbed environment and hardware

from all APs are collected until the user type a next orientation. Afterward, the average of the collected RSSI values are used to create a new fingerprint recode which holds

$$\psi = \{X_i, Y_j, direction, \{RSSI_1, RSSI_k, \ldots, RSSI_L\}\}k, i, j \in R^N \tag{5.2}$$

Where $RSSI_k$ is the average of RSSI on AP_k and L is the number of APs. Then, the created fingerprint record ψ is stored into the database. The procedures are repeated four times for each RP in the radio map until all RPs in the interesting area are visited. Figure 5.4 shows a screenshot of the fingerprint table in the database that is known as the radio map.



#	id	positionX	positionY	Direction	mac_ap	rssi	region
1	19	1.0	1.0	0	00:1c:10:a9:05:27	-63.904762	NULL
2	20	1.0	1.0	0	00:1c:10:a9:0b:2a	-51.761906	NULL
3	21	1.0	1.0	0	00:1c:10:a9:0b:66	-38.0	NULL
4	22	1.0	1.0	0	00:1c:10:a9:07:fd	-61.47619	NULL
5	23	1.0	1.0	0	00:1c:10:a8:ff:f9	-54.0	NULL
6	24	1.0	1.0	90	00:1c:10:a9:05:27	-69.90476	NULL
7	25	1.0	1.0	90	00:1c:10:a9:0b:2a	-53.142857	NULL
8	26	1.0	1.0	90	00:1c:10:a9:0b:66	-31.857143	NULL
9	27	1.0	1.0	90	00:1c:10:a9:07:fd	-57.904762	NULL
10	28	1.0	1.0	90	00:1c:10:a8:ff:f9	-55.095238	NULL
11	29	1.0	1.0	180	00:1c:10:a9:05:27	-69.0	NULL
12	30	1.0	1.0	180	00:1c:10:a9:0b:2a	-50.714287	NULL
13	31	1.0	1.0	180	00:1c:10:a9:0b:66	-24.857143	NULL
14	32	1.0	1.0	180	00:1c:10:a9:07:fd	-58.52381	NULL
15	33	1.0	1.0	180	00:1c:10:a8:ff:f9	-53.190475	NULL
16	34	1.0	1.0	270	00:1c:10:a9:05:27	-70.85714	NULL
17	35	1.0	1.0	270	00:1c:10:a9:0b:2a	-60.285713	NULL
18	36	1.0	1.0	270	00:1c:10:a9:0b:66	-50.0	NULL
19	37	1.0	1.0	270	00:1c:10:a9:07:fd	-60.0	NULL
20	38	1.0	1.0	270	00:1c:10:a8:ff:f9	-64.0	NULL
21	39	3.0	1.0	0	00:1c:10:a9:05:27	-69.47619	NULL
22	40	3.0	1.0	0	00:1c:10:a9:0b:2a	-49.095238	NULL
23	41	3.0	1.0	0	00:1c:10:a9:0b:66	-46.0	NULL

Figure 5.4: Fingerprint database

5.4.1.2 Test Point Collection

After constructing the radio map, a set of 17 test points were collected selected at random locations, in order to use it later for evaluating the performance of both KNN and CS methods. The test points were collected in different ways. Some of them were collected by associating mobile device with APs and other were collected by capturing the probe request packets that are broadcasted from the mobile device. In the next section, the generated radio map and the collected test point are used to studies the performance of both KNN and CS localization methods.

5.4.2 KNN Evaluation

In this section, the KNN performance is evaluated in terms of the localization error which is calculated by taking the average of the Euclidean distance between the actual location of the volunteer and the estimated position over 17 test points using KNN and the constructed radio map. The calculation is repeated for studying different cases. First, we studied the impact of number of RPs on the average of localization error. The number of RP has been changed between 1 to 12. The average of localization error for each selected k is depicted in Figure 5.5.

Figure 5.5 shows the average localization error, with respect to the number of RPs using KNN for both average and integrated radio maps. We studied the benefits of integrating the orientation data with the radio map across the four orientations. Obviously, selecting K equal to 7 is the well-suited for our environment where it achieves the minimum error. The result also shows that the average of localization error can vary between 4m and 2.86m. The performance improvement is decreased as k increase above 7. The variable K should not set to a high value which involves many RPs in calculating the user position in order to avoid calculating the center point of area instead of estimating the exact user position. Thus, the optimal k value was selected during the calibrating phase.

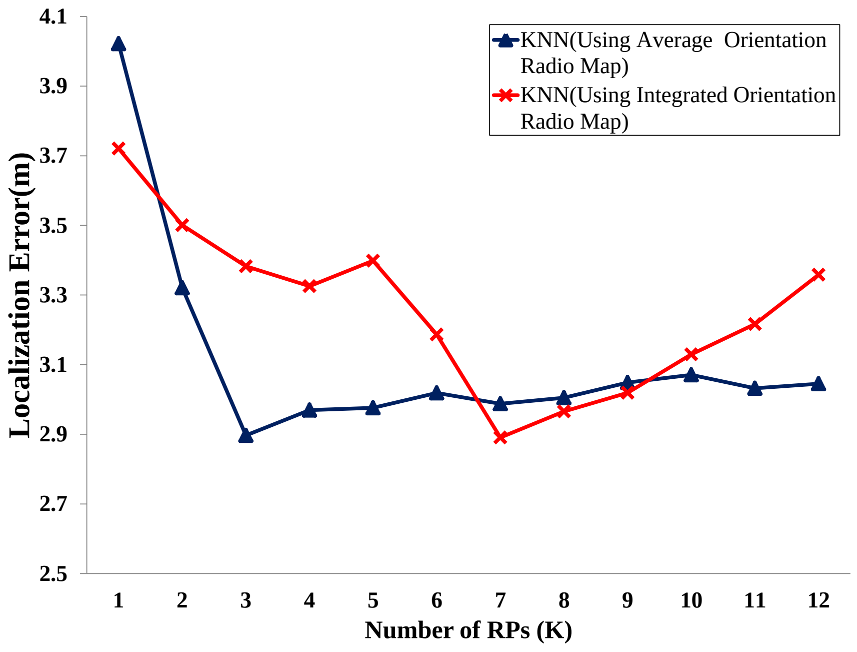


Figure 5.5: Average error using KNN for both the average and the integrated radio maps

The figure also shows the average of localization error for both calculation using averaged radio map and using integrated radio map with respect to the selected K from 1 to 12. The figure demonstrates that the average radio map is more stable than integrated radio map and the improvement of performance keep stable above $k = 3$. The minimum average localization error is almost the same. For studying the effect of number of AP on the average localization error. The KNN method has been run again but this time versus the number of APs which were changed from one to five APs. Figure 5.6 shows the average localization error with respect to the number of used APs while k was set to four.

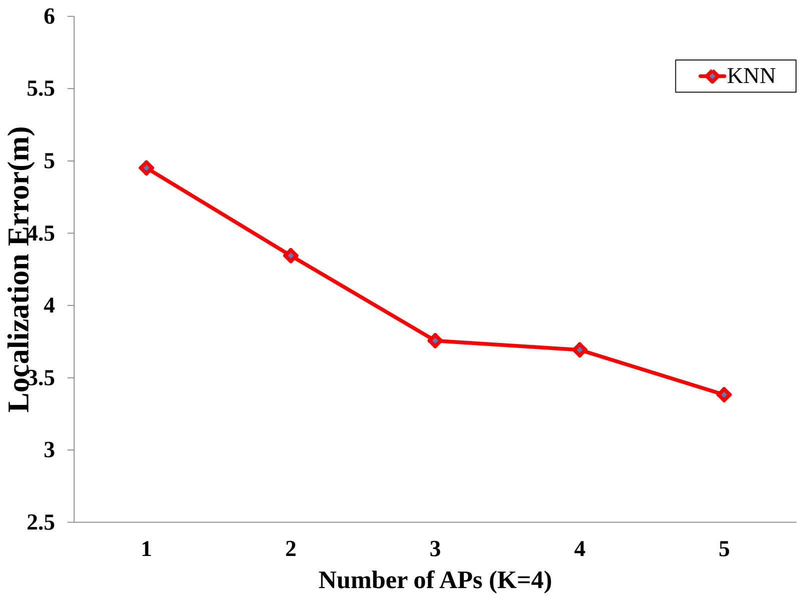


Figure 5.6: Average error versus the number of APs using KNN

The result shows that performance of the KNN methods improves as the number of APs increases. Thus, the number of APs significantly affects the performance of localization estimation. The last evaluation is testing the improvement of using the WKNN method. Again, the average localization error is calculated by using WKNN and using both average radio map and integrated radio map as shown in Figure 5.6. Both KNN and WKNN methods have similar performance because the number of APs is not relatively high and the weighted value does not affect the accuracy.

5.4.3 CS Evaluation

In this section, the improvement of CS on the the performance of localization estimation is investigated. The CS method was implemented to estimate the position of the collected test points. The estimated position is determined by estimating the centroid of the minimum positive values of the recovered output vector θ. Then, the Euclidean

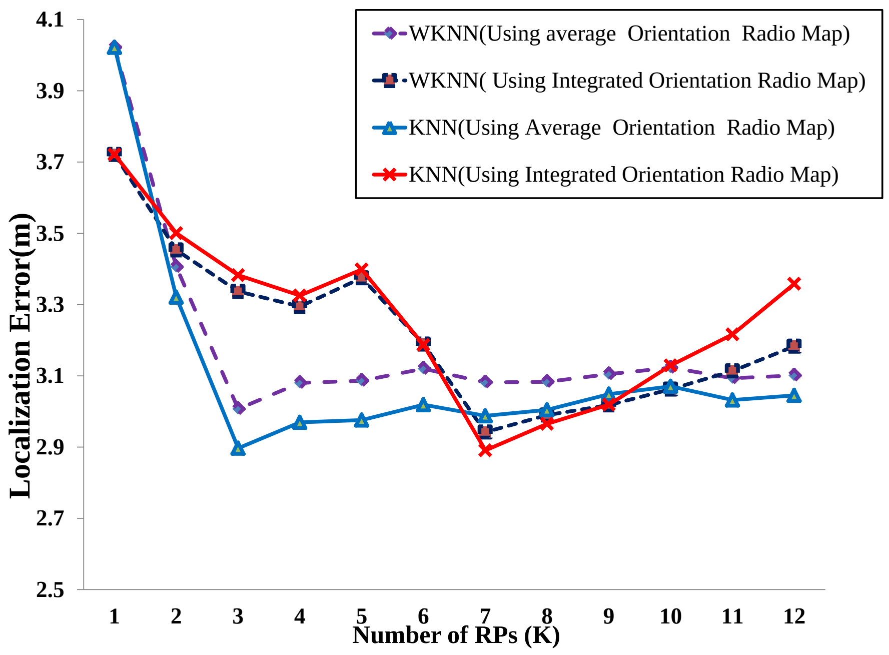


Figure 5.7: Average error versus the number of RPs using WKNN and KNN

distance between the actual user's location and the estimated position over 17 tests points and using 5 APs are calculated. First, the impact of number of cluster on the average localization error, we run the CS method several times for different number of clusters and then we calculated the average localization error, as shown in Figure 5.8.

The figure shows that the number of clusters significantly affects the localization performance. It also shows that CS method minimizes the average error form 2.9m with KNN to approximately 2m. Moreover, the figure shows that the average error varies in the range between 2m as the minimum distance and 2.5 m. Compared to the KNN method, we found that CS achieves better localization for the various number of APs.

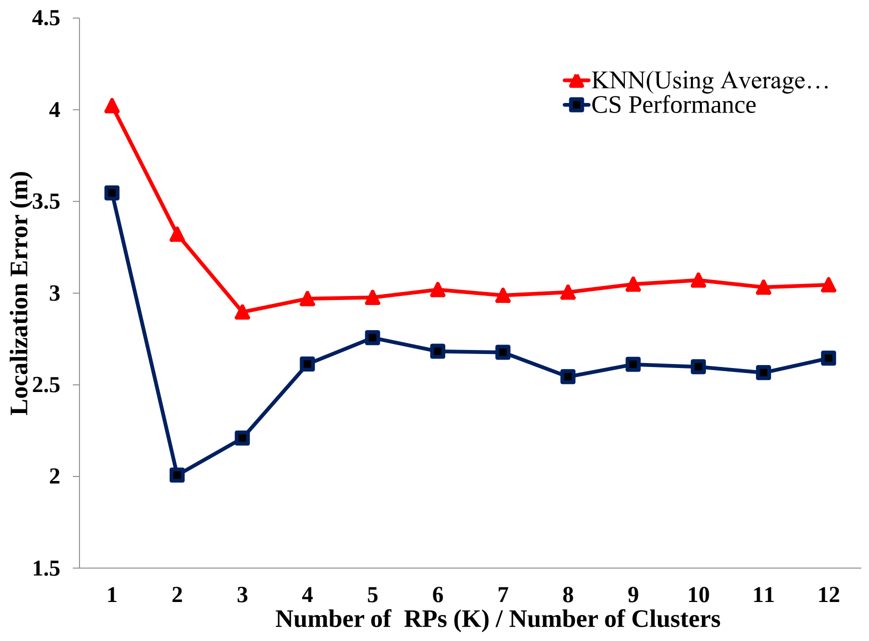


Figure 5.8: Average error versus the number of RPs using CS and KNN

6 Conclusions and Future Work

6.1 Summary

In this thesis, we presented the design and the implementation of an occupancy monitoring method for indoor public sensing application that leverages the existing APs and exploiting the already-existent wifi traffic to localize the mobile device. The approach includes two algorithms which are (1) detection and data collection algorithm runs on the AP level and (2) positioning estimation algorithm which run on the server. We demonstrated the possibility of developing occupancy monitoring method through several traffic measurement experiments to check the applicability of exploiting the already-existent WiFi traffic. The traffic analysis indicated that there is enough traffic. Based on the traffic analysis, we attempted to improve the accuracy of localization estimation by applying a machine learning algorithm where the radio map is clustered by using affinity propagation clustering algorithm and compressive sensing has been applied to accurately find the mobile device's position.

In contrast to the existent occupancy monitoring algorithms in [BXN+13; MRNC11] which can detect the existing of the mobile device within the space of AP and predict the existing of mobile device in the space of the occupant office. Our proposal approach improves the accuracy of monitoring occupancy by detecting both associated and unassociated mobile devices even if the mobile devices are inactive. We also attempted to improve the accuracy of monitoring occupancy by applying a machine learning algorithm where the radio map is clustered by using affinity propagation algorithm. The CS method improved the accuracy from 2.9m in case of KNN to 2m. In addition, our proposed approach reduce the cost of monitoring the occupancy as the algorithm can be deployed on cheap commercial AP which already exist in most indoor environments.

6.2 Future Work

Our proposal can be improved from different point of views. To improve the accuracy, time-based such as *Differential Time Difference of Arrival (*DTDOA*)* can be used instead of the RSS-based methods. Actually, we attempted to employ a time-based method for monitoring the occupancy through modifying an open-source firmware to exploit the RTC/CTS and DATA/ACK traffic. Unfortunately, we faced many problems due to hardware limitations. Indeed, it requires special hardware to accurately estimate a position. Furthermore, the mobility of mobile devices can be detected precisely by monitoring the *Channel State Information (*CSI*)*. The performance also can be improved by increasing the traffic, either by concurrently monitoring all channels or via generating a new traffic by sending RTS frames to the targeted mobile devices. Then, the server receives CTS frames which can be processed and replied by the *Network interface controller (*NIC*)* card. The last proposal may add energy burden due to activating the NIC card for sending the CTS frames.

List of Abbreviations

Abbreviation	Meaning	First occurrence
ACK	Acknowledgment	34
AP	Access Point	19
API	Application Programming Interface	14
BLE	Bluetooth Low Energy	11
CS	Compressive Sensing	8
CSI	Channel State Information	77
CTS	clear To Send	34
DR	Dead Reckoning	12
DSSS	Direct Sequence Spread Spectrum	33
DTDOA	Differential Time Difference of Arrival	77
GPS	Global Positioning System	11
IPS	Indoor Positioning System	23
JSON	JavaScript Object Notation	59
KNN	K-Nearest Neighbor	7
LOS	Line Of Sight	28
LS	Localization Sever	23
MAC	Media Access Control	13
NIC	Network interface controller	77
NLOS	Non-Line Of Sight	28
PCAP	Packet Capture	59
PS	Public Sensing	7
QOS	Quality of Service	13
RP	Reference Point	28
RSS	Received Signal Strength	25
RSSI	Received Signal Strength Indicator	17
RTS	Request To Send	34
WEP	Wired Equivalent Privacy	33

Bibliography

[AGC13] J. Alvarez-Lozano, J. A. Garcia-Macias, E. Chávez. “Learning and User Adaptation in Location Forecasting.” In: *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*. UbiComp ’13 Adjunct. Zurich, Switzerland: ACM, 2013, pp. 461–470. ISBN: 978-1-4503-2215-7. DOI: 10.1145/2494091.2495978. URL: http://doi.acm.org/10.1145/2494091.2495978 (cit. on p. 29).

[Bai15] P. Baier. “Efficient query distribution and positioning in public sensing systems.” PhD thesis. Stuttgart, Universitaet Stuttgart, Diss., 2015, 2015 (cit. on p. 15).

[Ban13] S. Banerji. “On IEEE 802.11: Wireless LAN Technology.” In: *CoRR* abs/1307.2661 (2013). URL: http://arxiv.org/abs/1307.2661 (cit. on pp. 33, 34).

[BDR13] P. Baier, F. Duerr, K. Rothermel. “Opportunistic Position Update Protocols for Mobile Devices.” In: *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp ’13. Zurich, Switzerland: ACM, 2013, pp. 787–796. ISBN: 978-1-4503-1770-2. DOI: 10.1145/2493432.2493439. URL: http://doi.acm.org/10.1145/2493432.2493439 (cit. on pp. 12, 22).

[Ber16] J. Berg. *radiotap.org*. http://www.radiotap.org. 2016 (cit. on pp. 35, 63).

[BXN+13] B. Balaji, J. Xu, A. Nwokafor, R. Gupta, Y. Agarwal. “Sentinel: occupancy based HVAC actuation using existing WiFi infrastructure within commercial buildings.” In: *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*. ACM. 2013, p. 17 (cit. on pp. 30, 77).

[CCR10] I. Constandache, R. R. Choudhury, I. Rhee. “Towards mobile phone localization without war-driving.” In: *Infocom, 2010 proceedings ieee*. IEEE. 2010, pp. 1–9 (cit. on p. 30).

[Far11] S. Farahani. *ZigBee wireless networks and transceivers*. newnes, 2011 (cit. on pp. 28, 53).

[FAVT09] C. Feng, W. S. A. Au, S. Valaee, Z. Tan. “Orientation-aware indoor localization using affinity propagation and compressive sensing.” In: *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2009 3rd IEEE International Workshop on*. IEEE. 2009, pp. 261–264 (cit. on p. 64).

[FAVT10] C. Feng, W. S. A. Au, S. Valaee, Z. Tan. “Compressive sensing based positioning using RSS of WLAN access points.” In: *INFOCOM, 2010 Proceedings IEEE*. IEEE. 2010, pp. 1–9 (cit. on pp. 55–57, 64, 68).

[FD07] B. J. Frey, D. Dueck. “Clustering by passing messages between data points.” In: *science* 315.5814 (2007), pp. 972–976 (cit. on p. 50).

[FNI13] Z. Farid, R. Nordin, M. Ismail. “Recent advances in wireless indoor localization techniques and system.” In: *Journal of Computer Networks and Communications* 2013 (2013) (cit. on p. 12).

[HW08] C. Hoene, J. Willmann. “Four-way TOA and software-based trilateration of IEEE 802.11 devices.” In: *2008 IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE. 2008, pp. 1–6 (cit. on p. 35).

[JLM16] V. Jacobson, C. Leres, S. McCanne. *pcap: Packet Capture library*. URL: http://www.tcpdump.org/. accessed: 27.11.2016 (cit. on p. 62).

[KBBN11] M. B. Kjærgaard, S. Bhattacharya, H. Blunck, P. Nurmi. “Energy-efficient trajectory tracking for mobile devices.” In: *Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM. 2011, pp. 307–320 (cit. on p. 22).

[KLGT09] M. B. Kjærgaard, J. Langdal, T. Godsk, T. Toftkjær. “Entracked: energy-efficient robust position tracking for mobile devices.” In: *Proceedings of the 7th international conference on Mobile systems, applications, and services*. ACM. 2009, pp. 221–234 (cit. on pp. 12, 30).

[KMD13] I. Koenig, A. Q. Memon, K. David. “Energy consumption of the sensors of Smartphones.” In: *Proceedings of the 10th International Symposium on Wireless Communication Systems (ISWCS)*. Ilmenau, Germany: IEEE, Aug. 2013, pp. 1–5. ISBN: 978-3-8007-3529-7. URL: http://dblp.uni-trier.de/db/conf/iswcs/iswcs2013.html#KonigMD13 (cit. on p. 21).

[KPV07] A. Kushki, K. N. Plataniotis, A. N. Venetsanopoulos. “Kernel-based positioning in wireless local area networks.” In: *IEEE transactions on mobile computing* 6.6 (2007), pp. 689–705 (cit. on p. 53).

[Kur15] F. D. Kurt Rothermel, ed. *Public Sensing – Using Your Mobile Phone for Crowd Sourcing*. Vol. 55th. Stuttgart University - Photogrammetric Week, 2015 (cit. on pp. 15, 16).

[LNR01] A. Leonhardi, C. Nicu, K. Rothermel. “A map-based dead reckoning protocol for updating location information.” In: (2001) (cit. on p. 29).

[LR01] A. Leonhardi, K. Rothermel. “A comparison of protocols for updating location information.” In: *Cluster Computing* 4.4 (2001), pp. 355–367 (cit. on pp. 12, 22, 25).

[LSDR] B. Li, J. Salter, A. G. Dempster, C. Rizos. “Indoor positioning techniques based on wireless LAN.” In: *LAN, FIRST IEEE INTERNATIONAL CONFERENCE ON WIRELESS BROADBAND AND ULTRA WIDEBAND COMMUNICATIONS*, pp. 13–16 (cit. on p. 54).

[MRNC11] R. Melfi, B. Rosenblum, B. Nordman, K. Christensen. “Measuring building occupancy using existing network infrastructure.” In: *Green Computing Conference and Workshops (IGCC), 2011 International*. IEEE. 2011, pp. 1–8 (cit. on pp. 30, 77).

[Pey13] G. Peyre. *Project Title*. https://github.com/gpeyre/matlab-toolboxes. 2013 (cit. on p. 69).

[Pri13] Primer. *Wi-Fi: Overview of the 802.11 Physical Layer and Transmitter Measurements*. 2013. URL: http://info.tek.com/www-wi-fi-overview-of-the-physical-layer-and-transmitter-measurements-primer.html (cit. on p. 33).

[RCK+09] R. K. Rana, C. T. Chou, S. Kanhere, N. Bulusu, W. Hu. “Ear-Phone assessment of noise pollution with mobile phones.” In: *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. ACM. 2009, pp. 395–396 (cit. on p. 16).

[SMRP] R. Singh, L. Macchi, C. S. Regazzoni, K. N. Plataniotis. “A statistical modelling based location determination method using fusion technique in wlan.” In: () (cit. on p. 53).

[TTZT13] Z. Tian, X. Tang, M. Zhou, Z. Tan. “Fingerprint indoor positioning algorithm based on affinity propagation clustering.” In: *EURASIP Journal on Wireless Communications and Networking* 2013.1 (2013), pp. 1–8 (cit. on pp. 51–53, 64).

[WAK15] D. Wang, T. Abdelzaher, L. Kaplan. *Social Sensing: Building Reliable Systems on Unreliable Data*. Elsevier Science, 2015. ISBN: 9780128011317. URL: https://books.google.de/books?id=o-OcBAAAQBAJ (cit. on p. 13).

All links were last followed on November 30, 2016.

Declaration