THE UNIVERSITY *of* EDINBURGH

DESIGN OF AN ADAPTIVE RF FINGERPRINT INDOOR POSITIONING SYSTEM

Roslee Mohd Sabri

THE UNIVERSITY *of* EDINBURGH

Thesis submitted for the degree of

Doctor of Philosophy

University of Edinburgh

School of Engineering

2017

Abstract

RF fingerprinting can solve the indoor positioning problem with satisfactory accuracy, but the methodology depends on the so-called radio map calibrated in the offline phase via manual site-survey, which is costly, time-consuming and somewhat error-prone. It also assumes the RF fingerprint's signal-spatial correlations to remain static throughout the online positioning phase, which generally does not hold in practice. This is because indoor environments constantly experience dynamic changes, causing the radio signal strengths to fluctuate over time, which weakens the signal-spatial correlations of the RF fingerprints. State-of-the-arts have proposed adaptive RF fingerprint methodology capable of calibrating the radio map in real-time and on-demand to address these drawbacks. However, existing implementations are highly server-centric, which is less robust, does not scale well, and not privacy-friendly. This thesis aims to address these drawbacks by exploring the feasibility of implementing an adaptive RF fingerprint indoor positioning system in a distributed and client-centric architecture using only commodity Wi-Fi hardware, so it can seamlessly integrate with existing Wi-Fi network and allow it to offer both networking and positioning services. Such approach has not been explored in previous works, which forms the basis of this thesis' main contribution.

The proposed methodology utilizes a network of distributed location beacons as its reference infrastructure; hence the system is more robust since it does not have any single point-of-failure. Each location beacon periodically broadcasts its coordinate to announce its presence in the area, plus coefficients that model its real-time RSS distribution around the transmitting antenna. These coefficients are constantly self-calibrated by the location beacon using empirical RSS measurements obtained from neighbouring location beacons in a collaborative fashion, and fitting the values using path loss with log-normal shadowing model as a function of inter-beacon distances while minimizing the error in a least-squared sense. By self-modelling its RSS distribution in real-time, the location

beacon becomes aware of its dynamically fluctuating signal levels caused by physical, environmental and temporal characteristics of the indoor environment. The implementation of this self-modelling feature on commodity Wi-Fi hardware is another original contribution of this thesis.

Location discovery is managed locally by the clients, which means the proposed system can support unlimited number of client devices simultaneously while also protect user's privacy because no information is shared with external parties. It starts by listening for beacon frames broadcasted by nearby location beacons and measuring their RSS values to establish the RF fingerprint of the unknown point. Next, it simulates the reference RF fingerprints of predetermined points inside the target area, effectively calibrating the site's radio map, by computing the RSS values of all detected location beacons using their respective coordinates and path loss coefficients embedded inside the received beacon frames. Note that the coefficients model the real-time RSS distribution of each location beacon around its transmitting antenna; hence, the radio map is able to adapt itself to the dynamic fluctuations of the radio signal to maintain its signal-spatial correlations. The final step is to search the radio map to find the reference RF fingerprint that most closely resembles the unknown sample, where its coordinate is returned as the location result.

One positioning approach would be to first construct a full radio map by computing the RSS of all detected location beacons at all predetermined calibration points, then followed by an exhaustive search over all reference RF fingerprints to find the best match. Generally, RF fingerprint algorithm performs better with higher number of calibration points per unit area since more locations can be classified, while extra RSS components can help to better distinguish between nearby calibration points. However, to calibrate and search many RF fingerprints will incur substantial computing costs, which is unsuitable for power and resource limited client devices. To address this challenge, this thesis introduces a novel algorithm suitable for client-centric positioning as another contribution. Given an unknown RF fingerprint to solve for location, the proposed algorithm first sorts the RSS in descending order. It then iterates over

this list, first selecting the location beacon with the strongest RSS because this implies the unknown location is closest to the said location beacon. Next, it computes the beacon's RSS using its path loss coefficients and coordinate information one calibration point at a time while simultaneously compares the result with the measured value. If they are similar, the algorithm keeps this location for subsequent processing; else it is removed because distant points relative to the unknown location would exhibit vastly different RSS values due to the different site-specific obstructions encountered by the radio signal propagation. The algorithm repeats the process by selecting the next strongest location beacon, but this time it only computes its RSS for those points identified in the previous iteration. After the last iteration completes, the average coordinate of remaining calibration points is returned as the location result. Matlab simulation shows the proposed algorithm only takes about half of the time to produce a location estimate with similar positioning accuracy compared to conventional algorithm that does a full radio map calibration and exhaustive RF fingerprint search.

As part of the thesis' contribution, a prototype of the proposed indoor positioning system is developed using only commodity Wi-Fi hardware and open-source software to evaluate its usability in real-world settings and to demonstrate possible implementation on existing Wi-Fi installations. Experimental results verify the proposed system yields consistent positioning accuracy, even in highly dynamic indoor environments and changing location beacon topologies.

Lay Summary

RF fingerprinting is a popular indoor positioning algorithm that uses signal-of-opportunity of Wi-Fi networks. It requires a so-called radio map, which is a list of known locations and their respective RF fingerprints i.e. a collection of RSS of Wi-Fi networks heard on site. Previously, radio map is calibrated via site-survey by physically visiting each location and recording its RF fingerprint. Although simple, this method is laborious, time-consuming and somewhat error-prone. The calibrated radio map also does not account for RSS fluctuations caused by various factors, such as temperature, humidity, opening/closing of doors, space reorganization, crowd movements, etc. Hence, the previously calibrated radio map might become outdated later.

To address these drawbacks, several works have proposed an on-demand radio map calibration using arrays of RF fingerprint sensors deployed at known indoor locations. These sensors constantly measure RF fingerprint of their locations in real-time and upload them to a central location server for post-processing into final radio map. Any mobile client wanting to locate itself can either download the radio map and compute its position locally, or upload the unknown RF fingerprint and request the server to compute its position and relay the result back. The disadvantage of present solution is its complex sensor and networking infrastructure that is costly to implement. The location server is also a single point of failure, and if it is down, the whole system is out-of-service. In addition, mobile clients must exchange identifiable information with the location server to access its service. Possible breach of privacy may occur if the server is hacked and all personal information stolen.

This thesis proposes a low cost alternative to present solution by exploiting existing networks of Wi-Fi APs to collect the RF fingerprints instead of using dedicated sensor arrays. These fingerprints are then exchanged between neighbouring APs through the wireless interface, so that each AP ends up with its own RSS distribution as a function of propagation distance that it can model

to reflect its real-time radio propagation characteristics. The model's coefficients are then broadcasted together with the AP's location coordinate. Using these two pieces of information, a mobile client could simulate the RSS of all detected APs for each predetermined location within the site, effectively calibrating the radio map "on-the-fly" that it can use to solve its location. The distributed architecture of the proposed solution is similar to how GPS satellites transmit timing and orbital information used by ground receivers to compute their own positions.

The proposed methodology requires that mobile client simulates a new radio map and searches for fingerprint match each time it wants to locate itself, which is compute-intensive that might affect the system's response. As system improvement, a novel algorithm is proposed to minimize the client's computing time. The idea is to incrementally compute RSS of one AP at a time for each calibration point and simultaneously compare the estimate with the actual value. If they are significantly different, the algorithm immediately knows that this particular point is unlikely to be the correct location so it can be removed. In the next iteration, the algorithm only needs to compute RSS of subsequent AP for the remaining calibration points identified in previous iteration. This cycle is repeated, and for subsequent iteration, the number of calibration points will reduce in a systematic manner, thus reducing the computing burden of the client. When all APs have been surveyed, the average coordinate of the remaining calibration points is returned as the location result. It is also possible that only one point remains before the last iteration completes. In this case, the algorithm no longer needs to continue and it can just return this point as the location result.

A system prototype was developed to demonstrate the feasibility and usability of the proposed solution in real-world scenarios using commodity off-the-shelf Wi-Fi hardware and open-source software. From experimental results, the proposed system prototype exhibits consistent positioning performance even when subjected to varying radio propagation environment and changing wireless network topologies.

Publications

R. Mohd Sabri, T. Arslan. (2011, Nov. 29 – Dec. 1, 2011). *A Real-Time Wi-Fi Indoor Positioning System*. Paper presented at the European Navigation Conference, London, UK.

R. Mohd Sabri, T. Arslan. (2011, Sept. 19-23, 2011). *Inferring Wi-Fi Angle-of-Arrival from Received Signal Strength Distribution*. Paper presented at the 24th International Technical Meeting of the Satellite Division of Institute of Navigation, Portland, Oregon, USA.

R. Mohd Sabri, T. Arslan. (2010, Sept. 21-24, 2010). *Accurate Mapping of Wi-Fi Access Point Location for Indoor Positioning.* Paper presented at the 23rd International Technical Meeting of the Satellite Division of Institute of Navigation, Portland, Oregon, USA

F. Alsehly, R. Mohd Sabri, Z. Sevak, T. Arslan. (2011, Sept. 19-23, 2011). *Customizing Indoor Positioning for Shopping Centres Environments.* Paper presented at the 24th International Technical Meeting of the Satellite Division of Institute of Navigation, Portland, Oregon, USA.

F. Alsehly, R. Mohd Sabri, Z. Sevak, T. Arslan. (2010, Sept. 21-24, 2010). *Dynamic Indoor Positioning with the Handover Algorithm*. Paper presented at the 23rd International Technical Meeting of the Satellite Division of Institute of Navigation, Portland, Oregon, USA.

Declarations of Originality

I confirm that this thesis presented for the degree of Doctor of Philosophy, has

i. been composed entirely by myself
ii. been solely the result of my own work
iii. not been submitted for any other degree or professional qualification

Roslee Mohd Sabri

19th March 2018

Acknowledgements

First and foremost, I would like to extend my deepest gratitude to my supervisor, Professor Tughrul Arslan, for his constant encouragement, guidance and expertise throughout my PhD research.

My sincerest appreciation goes out to all my colleagues and others who have contributed directly and indirectly to the completion of this research and thesis. Their views and tips are useful indeed. At the same time, the constant encouragement and camaraderie shared between all my friends during my PhD studies has been an enriching experience.

I would also like to express my deepest gratitude and thanks to my sponsor, Majlis Amanah Rakyat (MARA) and the Government of Malaysia for funding my studies.

Finally, I would like to express my love and appreciation to my wife who has shown unrelenting care and support throughout this challenging endeavour. To my daughter and son, thank you for making my life so wonderful.

Table of Contents

List of Figures

List of Tables

Acronyms and Abbreviations

ABI	Application Binary Interface
AMD	Advanced Micro Devices
ANN	Artificial Neural Network
AP	Access Point
API	Application Programming Interface
BSS	Basic Service Set
BSSID	BSS Identifier
CDF	Cumulative Distribution Function
dBm	DeciBels below 1 Milliwatt
ESS	Extended Service Set
FAF	Floor Attenuation Factor
GB	Giga Bytes
GHz	Giga Hertz
GNSS	Global Navigation Satellite System
IBSS	Infrastructure BSS
IE	Information Element
IEEE	Institute of Electrical and Electronics Engineers
KNN	K Nearest Neighbour
LAN	Local Area Network
LBS	Location Based Services
LEASE	Location Estimation Assisted by Stationary Emitters
LoS	Line of Sight

LTS	Long Term Support
MAC	Medium Access Control
MB	Mega Byte
MS	Mobile Station
NN	Nearest Neighbour
OS	Operating System
OUI	Organization Unique Identifier
QoS	Quality of Service
RAM	Random Access Memory
RF	Radio Frequency
RSS	Received Signal Strength
TFTP	Tiny File Transfer Protocol
WAF	Wall Attenuation Factor
WKNN	Weighted K Nearest Neighbour
WLAN	Wireless LAN

Nomenclature

IEEE 802.11	IEEE specification for wireless communication in the unregulated radio spectrum
Access Point(s)	Provide wireless access to the network
Bootloader	A program that loads an operating system when a computer is turned on
Channels	Frequencies within the unregulated radio spectrum that are available for use by wireless devices. Channel usage for wireless devices is regulated by each individual country and varies worldwide
Coverage	The amount of area in which the wireless network is available
Daemon	A background process that handles requests for services and is dormant when not required
Line of Sight	The ability to see one network component from another without obstacles in between
Mobile station	Any 802.11 wireless client device located within the network
RF Interference	Noise within the radio band that causes communication and connectivity issues between components on the network
RF Fingerprint	A collection of radio signal information, typically received signal strengths, to identify specific location in target area
Roaming	The ability of a mobile station to seamlessly switch between Access Points on a network and maintain connectivity

Chapter 1 : Introduction

1.1. Background

Advances in wireless communication technologies and mobile computing devices have allowed users to be highly mobile yet connected to the network at the same time. Given this newfound freedom in mobility, being able to determine user's location in real-time would enable the network to provide context-specific information that can improve his/her computing experience. For examples, providing turn-by-turn direction to the nearest pharmacy to pick up a prescription, or receiving shopping reminder on the mobile phone as user walks past a favourite grocery store. Delivering this location information requires a positioning system that is both pervasive and low-cost to ensure successful large-scale adoption (LaMarca et al., 2005).

GNSS is an example of a highly successful outdoor location technology. Its coverage is available anywhere in the world and the service is free-of-charge. Various outdoor location-aware applications and services use GNSS to obtain location information, with car navigation being one of the most successful use cases. However, satellite signals are either non-existent or too weak to be received indoors, which limits its usability in these environments. This is unfortunate because there are various indoor applications and services that could benefit from location-awareness, such as asset tracking, concierge services, monitoring of minor and the elderly, etc. Thus, an indoor location system is required to fulfil this need.

The RADAR project by Microsoft Research introduced an innovative indoor positioning system that exploits radio signals of IEEE802.11 wireless local area networks, popularly known as Wi-Fi, to infer locations (Bahl & Padmanabhan, 2000). The RADAR system uses the so-called radio map, which is a lookup table comprising of RSS values of multiple Wi-Fi networks observed at known locations inside the building. These RSS-tuples, known as RF fingerprint in the literature, are

generally unique to the locations where they are measured. This is because radio signals from multiple APs propagate along independent paths and arrive at a particular spot with varying RSS values due to the diverse and random obstructions encountered relative to one another. Locating any point within the building simply reduces to obtaining its RF fingerprint by measuring the RSS of Wi-Fi signals heard onsite, looking up the radio map to find the best match and returning the coordinate of this closest-match RF fingerprint as the location result. RF fingerprinting can yield median positioning error of 10ft (Elnahrawy, Xiaoyan, & Martin, 2004). Such positioning performance is sufficient to satisfy the requirements of many indoor location-aware applications.

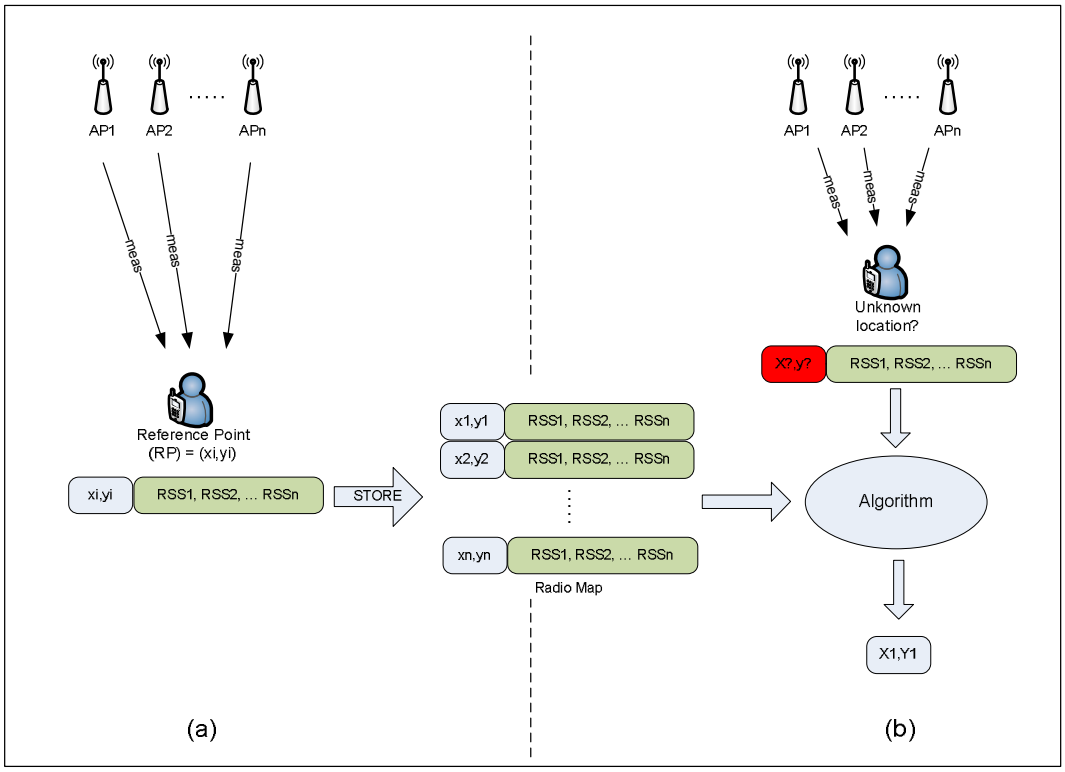


Fig. 1.1: RF fingerprinting algorithm (a) offline phase, (b) online phase

The basic setup of RF fingerprint positioning system involves a two-step process. The first step is to calibrate the radio map of the target area by partitioning the site into grids of identifiable locations, and surveying each location by an expert surveyor to record the RSS of multiple Wi-Fi signals heard onsite into a lookup

table. Typically, the site-survey process is performed several rounds over different periods while the collected RSS data are post-processed to minimize error, remove outliers and extract useful information. The second step is where the actual positioning takes place. First, obtain the RF fingerprint of the unknown location by measuring RSS of multiple Wi-Fi signals heard onsite. Next, compare it against the pre-calibrated reference RF fingerprints in the radio map to find the closest-matched record, usually by minimizing some distance function (Kaemarungsi, 2005). Finally, return the coordinate of this most similar RF fingerprint as the location result. Fig. 1.1 illustrates the RF fingerprinting algorithm.

1.2. Motivation

Conventional RF fingerprint algorithm depends on manual site-survey to calibrate the radio map, where an expert surveyor must physically visit each pre-determined location at the site in order to record its RF fingerprint. Such process is labour-intensive, time-consuming and subjected to human-error. Moreover, to resolve the location of an unknown RF fingerprint, the algorithm does an exhaustive search over all reference RF fingerprints inside the radio map to find the best match. This search contributes to the processing time of the system and can be significant if the search space is large or if the radio map has a high RF fingerprint density per unit area. This delay can affect the system's responsiveness and in some cases might even be detrimental to its overall performance.

Another challenge is maintaining the validity of radio map over time. An implicit assumption of RF fingerprinting is that radio map is static; that is once calibrated it can be used to estimate locations in later periods without adaptation (Jie, Qiang, & Lionel, 2005). Unfortunately, such assumption generally does not hold in practice because indoor environment constantly experiences dynamic changes, such as fluctuating humidity levels, opening/closing of doors, space reorganizations and remodelling, crowd movements, etc. that affect the RSS of radio signals in stochastic manner. This effectively alters the RF fingerprint model of the site temporally,

which renders the previously calibrated radio map unusable. Frequent recalibration of radio map is often necessary but static techniques that extensively profile the site involve steep upfront costs and effort to deployment, and add significantly to the complexity of managing the radio map (Krishnan, Krishnakumar, Ju, Mallow, Gamt, 2004).

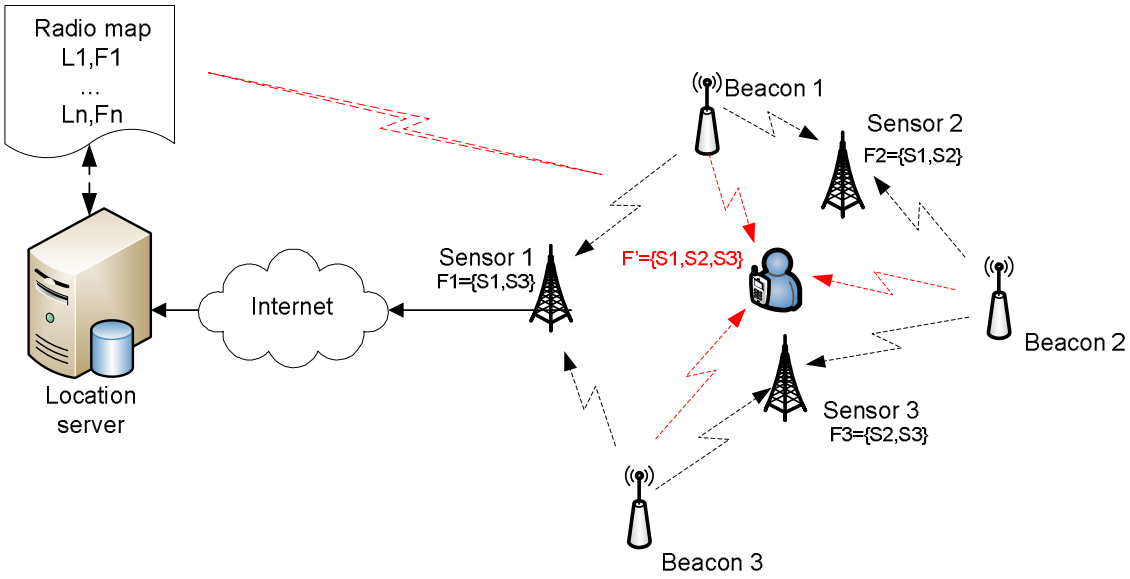


Fig. 1.2: Adaptive RF fingerprinting methodology

An innovative RF fingerprinting methodology, where radio maps are auto-calibrated on-demand to adapt to the dynamicity of indoor radio propagation environments, has been proposed by state-of-the-arts to address these drawbacks (Krishnan et al., 2004; Gwon & Jain, 2004; Jie et al., 2005; Moraes & Nunes, 2006; Hyuk, Lu-Chuan, Hou, & Haiyun, 2006; Ivanov, Nett, & Schemmer, 2008; Atia, Noureldin, & Korenberg, 2013). A representative implementation of this methodology makes use of a network of RF sensors deployed at known locations within the site and connected to a centralized location server via a communication network. These sensors continuously collect RF fingerprints at their respective locations and forward them to the server for post-processing to construct the radio map on-demand. Any client device wanting to locate itself either downloads the radio map and computes its position locally, or uploads its unknown RF fingerprint

and requests the server to resolve its position and relay the result back. Fig. 1.2 illustrates the adaptive RF fingerprinting methodology.

A straightforward implementation may deploy the RF sensors across the site following a dense spatial grid, so the server just consolidates the received RF fingerprints into a lookup table to construct the radio map. However, such strategy is not feasible in terms of hardware and communications costs. Practical implementation normally utilizes limited number of RF sensors uniformly deployed in sparse locations, and then applies suitable mathematical algorithms on these sparse RF fingerprints to construct the full radio map that meets the desired spatial granularity.

With this methodology, the process of collecting RF fingerprints and transforming them into radio maps happen automatically in real-time, thus eliminating the manual site-survey. In addition, the radio map can be constructed according to arbitrary spatial granularity easily and cheaply; for example if user wants to trade-off between positioning accuracy and computing efficiency. Moreover, since the server collects RF fingerprints over time, it knows if the signal-spatial integrity of current radio map has deteriorated, thus it can construct a new one. One approach is to compare the current RF fingerprints with previous samples to assess their disparities, and if the differences exceed certain thresholds, this can trigger the server to construct a new radio map. Alternatively, the server can repeatedly construct a new radio map every T period using the latest RF fingerprints it receives.

With location server managing the radio maps, naturally it can provide positioning service to client devices. The device would first acquire the RF fingerprint of the unknown location, and then uploads the readings and requests the server to compute its position based on the current radio map. The server then relays the location result back to the client device. By offloading the location computations to take advantage of the server's more powerful computing resources, more advanced

and computationally intensive positioning algorithms can be used for improved accuracy, which otherwise is inefficient or not possible on resource-limited client devices. Besides, by not having to do these expensive computations, client devices can preserve their battery lives.

1.3. Problem Statement

The main drawback of adaptive RF fingerprint methodology proposed by state-of-the-arts is their server-centric system architecture. While contributing to additional hardware cost, the location server is a single point of failure. If it is down due to hardware or power failures, or compromised by security breach, this will negatively affect the location service availability. Moreover, server access incurs additional communication costs and assumes network connectivity is available, which is not always the case because networks could be down due to maintenance or disasters. This makes the overall positioning system less robust. Consequently, system administrators must put in place strategies to ensure service availability and security, such as backup server, uninterrupted power supplies, encryption and authentication protocols, etc. Invariably, this will drive the deployment and operational costs of the positioning system up.

Limited scalability is another common drawback of server-centric system architecture. With the increase in coverage area, the number of RF sensors deployed will also increase proportionally, resulting in multiple-fold increase in the amounts of RF fingerprints the server must manage. This could be problematic if the server is undersized during deployment to handle such increase in storage and computational load. Furthermore, all servers have limited computing and hardware resources. This means the server can only service a finite number of users at the same time before they consume all the server's resources. Beyond this point, no additional users can request location service until the server has completed the earlier requests and reclaimed its resources.

An important aspect of any positioning system is privacy because location information is inherently sensitive since it reveals users' physical presence and, to some extent, their personal behaviours and activities. Users may feel threatened or uncomfortable if their locations and movements can be recorded and analysed, or misused (Dao, Rizos, & Wang, 2002). Thus, it is important that users are in full control of their location data, and only disclose them to trusted parties (Dao et al., 2002). Unfortunately, the server-centric architecture is not privacy-friendly because users must establish connection with the server and expose some identifiable information in order to access its service.

1.4. Research Overview

The main objective of this research is to advance the works in adaptive RF fingerprinting methodology to address the robustness, scalability and privacy drawbacks of state-of-the-art implementations. It explores the feasibility of implementing such systems in a distributed and client-centric architecture, which is a novel approach not explored in previous works. The research also aims to realize the proposed system using just commodity Wi-Fi hardware so it can seamlessly integrate with regular Wi-Fi networks. This has the advantage of using the same infrastructure to offer both networking and positioning services, thus adding value to existing Wi-Fi networks while helping to lower the overall costs of the indoor positioning system.

Fig. 1.3 illustrates a high-level architecture of the proposed adaptive RF fingerprint indoor positioning system. A network of software-enhanced commodity Wi-Fi APs that double as location beacons makes up its reference infrastructure. Each location beacon constantly broadcasts its coordinate and path loss coefficients that model its RSS distributions around the transmitting antenna by embedding these parameters into vendor-specific IE of Wi-Fi's beacon and probe response frames. Furthermore, the proposed system employs client-centric positioning model, whereby the client devices estimate their own locations, to improve system

scalability and privacy. The process starts by passively listening for broadcasted beacon frames or actively probing for probe response frames from nearby location beacons and measuring their RSS, which forms the RF fingerprint of the unknown point. Next, the client device decodes and parses the vendor-specific IE of the received frames to obtain the coordinates and path loss coefficients of all detected location beacons, and uses them to compute the location beacons' RSS values at the chosen grid-points inside the target area, effectively calibrating the site's radio map on-demand. Finally, the client device searches the newly calibrated radio map to find the closest-matched RF fingerprint and uses its coordinate as the location result.

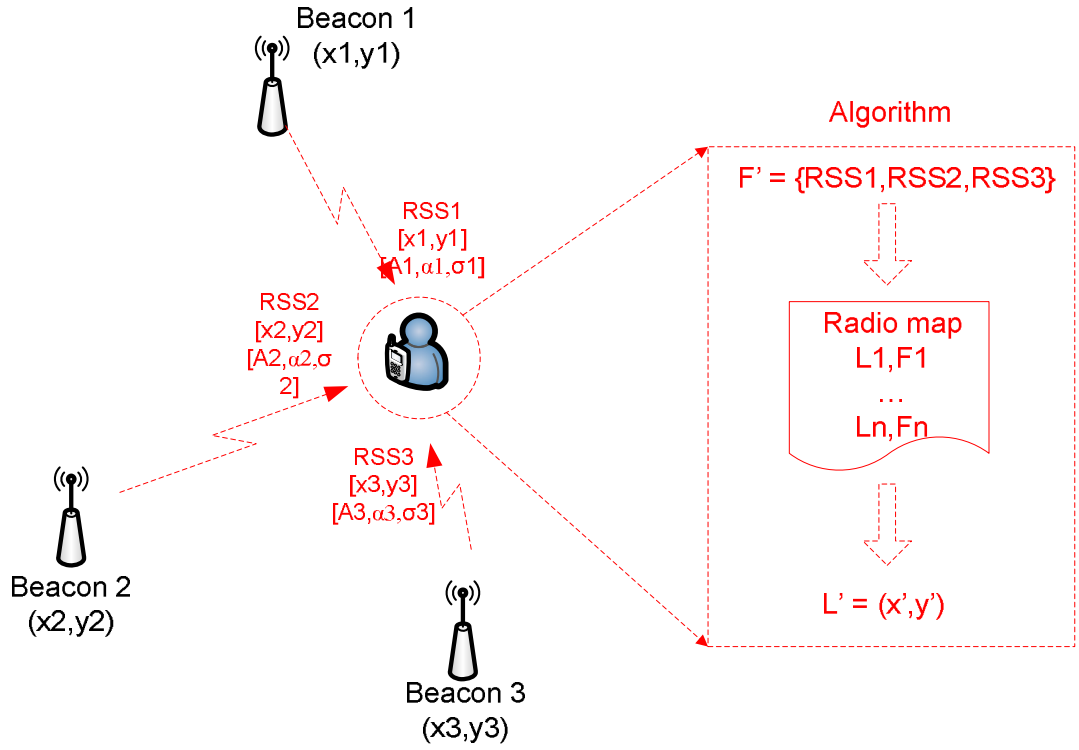


Fig. 1.3: High-level system architecture of the proposed adaptive RF fingerprinting indoor positioning system

Modelling the location beacon's path loss coefficients consists of acquiring its empirical RSS measurements from several locations around its transmitting antenna, and fitting the path loss model as a function of propagation distance using these empirical measurements while minimizing the error in a least squared sense.

An innovative feature of the proposed location beacon is its ability to self-model its path loss coefficients in real-time, which eliminates the needs for offline manual site-survey to collect the empirical RSS measurements, hence lowering the operational costs of the system. By constantly modelling its path loss characteristics, the location beacon implicitly learns the dynamic changes affecting its indoor radio propagation and transfers this knowledge to the client devices by updating the beacon and probe response frames. Consequently, the RSS computed using the updated path loss information would reflect the actual values more accurately, resulting in an improved radio map. This would yield more accurate location estimates since the unknown and reference RF fingerprints share similar signal-spatial correlations.

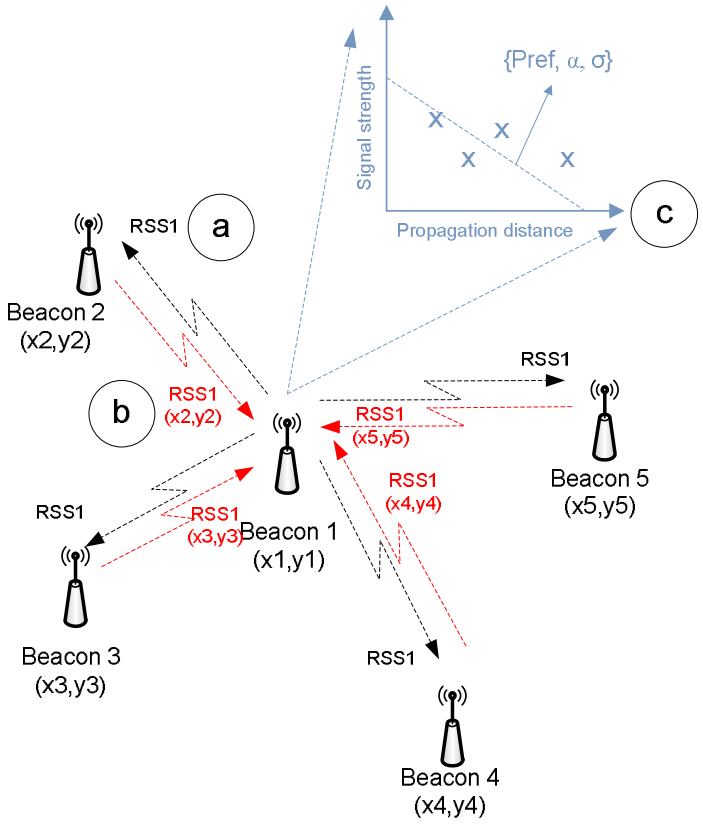


Fig. 1.4: Location beacon self-modelling process (a) neighbouring beacons (Beacon 2 to 5) measure RSS of target beacon (Beacon 1), (b) exchange RSS and coordinates between beacons, (c) target beacon calibrates its path loss model parameters via linear regression

The path loss self-modelling process happens in a distributed fashion via collaborative assistance among neighbouring location beacons, which is a novel methodology introduced in this thesis. Specifically, individual location beacon constantly measures RSS of its neighbours and exchanges these measurements among them in a collaborative fashion via the wireless interface, such that each location beacon ends up with its own RSS as measured by the neighbours. Since all location beacons also broadcast their coordinates, each location beacon knows exactly where the empirical RSS measurements originate from, so it can compute how far its radio signal has propagated. Equipped with the distance and RSS information, the location beacon has enough information to calibrate its path loss coefficients. This work uses a linear regression algorithm as the calibration function. Fig. 1.4 illustrates the path loss self-modelling process.

In the proposed adaptive RF fingerprint methodology, client device must calibrate a full radio map and then exhaustively search it to find the closest-matched RF fingerprint every time it needs to solve for locations. These tasks are computationally expensive and can incur substantial computing effort. Such approach is not suitable for client-centric positioning since client devices are generally small, embedded systems with limited computing power and battery life.

As system improvement, this thesis introduces a novel location-estimation algorithm that calibrates and searches the radio map in an incremental manner, in order to minimize the computing burden of client devices. Given an unknown RF fingerprint to solve for location, the algorithm first sorts the RSS in descending order. It then iterates over this list, first selecting the strongest RSS and its associated location beacon since this implies that the unknown location is closest to the said location beacon. Next, the algorithm iteratively computes the location beacon's RSS using the coordinate and path loss coefficients broadcasted by the said location beacon for each chosen location in the target area. If the computed RSS is within the specified range of the measured RSS value, the algorithm keeps

this location for subsequent processing; else it is removed because distant locations would exhibit vastly different RSS. The algorithm then repeats this process by selecting the next strongest location beacon, but this time it only computes the RSS for those locations identified in the previous iteration, hence the overall number of locations to calibrate and search is significantly reduced. After the last iteration completes, the algorithm returns the average of remaining locations as the location result. Compared with having to calibrate the full radio map and perform an exhaustive search to find the closest-matched RF fingerprint, the proposed algorithm yields similar positioning accuracy but it achieves this in half of the time.

1.5. Thesis Contributions

The contributions of this thesis are as follows:

1. Development of novel software that implements the self-modelling location beacon features using commodity Wi-Fi AP. The chosen AP is WRT54GL wireless router manufactured by Linksys running a Linux-based OpenWRT ver10.03 operating system. The software is a C program cross-compiled to run on the wireless router as a userspace Linux application.
2. Development of novel location-estimation algorithm, where radio map calibration and RF fingerprint search are performed simultaneously in a successive elimination manner to minimize computing burden of client devices.
3. Implementation of the proposed adaptive RF fingerprint indoor positioning system prototype to evaluate its performance in a real-world setting and to demonstrate possible deployment on existing Wi-Fi network.

1.6. Thesis Outline

The remainder of this thesis is as follows:

Chapter 2 presents an overview of the literature survey, classified based on certain taxonomy related to RF fingerprinting. The chapter concludes with reviews of several related works on adaptive RF fingerprint indoor positioning systems.

Chapter 3 presents the design of the proposed location beacon and its path loss radio propagation self-modelling feature to characterise its RSS distribution around the transmitting antenna. Client devices acquire these path loss coefficients and coordinates of all detected location beacons to construct an adaptive radio map prior to solving for each location. The chapter concludes with comparative discussions on positioning accuracy and computing time between the adaptive and static RF fingerprint algorithms in terms of varying radio propagation characteristics and changing location beacon topology.

Chapter 4 describes the iterative radio map calibration and location estimation algorithm as system improvement and discusses its positioning and computing performance against calibrating a full radio map and doing an exhaustive RF fingerprint search.

Chapter 5 details the implementation of the self-modelling location beacon using commodity Wi-Fi router and open-source software, as well as the implementation of location sensor using Wi-Fi capable laptop. The chapter concludes with discussions on the performance of the system prototype under varying wireless channel conditions and changing Wi-Fi network topology.

Chapter 6 summarizes and concludes this thesis. It re-highlights the thesis contributions and suggests directions for future works based on the techniques developed in this thesis.

1.7. Summary

Positioning system based on Wi-Fi networks and RF fingerprinting can solve the indoor positioning problems but its radio map calibration via manual site-survey is highly inefficient. Several state-of-the-arts propose an adaptive approach to radio

map calibration as system improvements. The methodology uses a network of RF sensors connected to a central server to continuously measure RF fingerprints at their respective locations and forward them to the server for post-processing to construct the radio map on-demand. However, the adaptive RF fingerprinting methodology proposed by the state-of-the-arts suffers from several drawbacks, such as less robust, limited scalability and not privacy-friendly.

This thesis proposes a novel methodology that implements the adaptive RF fingerprint positioning system following a distributed and client-centric architecture to address these drawbacks. The proposed system introduces a location beacon capable of self-modelling its path loss radio propagation to characterise its real-time RSS distributions around the transmitting antenna. Client devices use these path loss coefficients received from multiple location beacons detected onsite to compute their RSS at chosen grid-points in the target area, effectively calibrating the radio map. Since the path loss coefficients model the real-time RSS distributions of the location beacons, the just calibrated radio map reflects the current signal-spatial correlations of the site. This is more accurate compared to static radio map calibrated in the offline phase, whose signal-spatial integrity might have deteriorated over time. Equipped with the adaptive radio map, client devices solve for locations using a novel iterative radio map calibration and location estimation algorithm, which is another contribution of this thesis. This thesis also develops a prototype of the proposed system using commodity Wi-Fi hardware and open-source software to evaluate its feasibility in the context of real-world deployment.

Chapter 2 : Literature Review

2.1. Introduction to RF Fingerprinting

The main assumption of RF fingerprint algorithm is each point inside a building has a unique radio signature. The basis of this assumption resulted from the observation that indoor environment has many physical obstructions, such as walls, doors, floors, furniture, etc. that reflect, diffract and scatter radio waves in stochastic manner. The effect of these physical transformations results in radio signal that exhibits highly site-specific characteristics.

RF fingerprint algorithm consists of two main components. The first component is a model that describes the spatial distribution of RF fingerprints inside the target area, commonly known as the radio map. Each entry in the radio map represents one RF fingerprint, comprising of radio signature F labelled with location information L and denoted as a $\{L, F\}$-tuple.

The second component is the location estimation algorithm that takes as inputs the RF fingerprint sampled at an unknown location and outputs the estimated location where the unknown fingerprint is likely to originate. Essentially, RF fingerprint algorithm treats location estimation problem as classification problem. Each entry in the radio map represents a class of RF fingerprint that describes a particular location. The objective is to classify the unknown RF fingerprint into one of the predefined classes based on some optimization criteria (Kaemarungsi, 2005). Usually, the type of classifier and optimization scheme used is dependent on the type of information the RF fingerprints represent (Kaemarungsi, 2005).

2.2. Radio Map Calibration

Calibrating the radio map of the site is a pre-requisite of RF fingerprint algorithm. There are two main calibration approaches; those using empirical RSS measurements and those based on radio propagation modelling techniques.

2.2.1. Empirical Calibration

Calibrating an empirical radio map involves manual site-survey to collect RSS measurements of multiple radio sources at predefined locations. Suppose there are N radio sources/transmitters providing wireless network coverage. At each predefined location within the site, an expert surveyor collects several RSS samples from these N transmitters over a window of time t. For transmitter i, its average signal strength value p_i is computed and recorded as an element of the RF fingerprint. A vector of N average RSS values forms the RF fingerprint for each location i.e.

$$F = \{p_i\}, 1 \leq i \leq N$$

Additional information, such as standard deviation σ of each element in vector F, can also be included to provide extra information about the RF fingerprints (Saha, Chaudhuri, Sanghi, & Bhagwat, 2003)

$$D = \{\sigma_i\}, 1 \leq i \leq N$$

RF fingerprints represented in this manner are termed deterministic due to the constant RSS values used.

Another alternative describes the RF fingerprints in terms of conditional probability distribution of the form $P(F|L)$ where F denotes the observation vector of RSS and L denotes the location information (Kaemarungsi, 2005). The conditional probability $P(F|L)$ is called the *likelihood function* because it provides the probability of the occurrence of the RSS vector given the location information (Kaemarungsi, 2005).

Calibrating an empirical radio map is laborious and time-consuming, and represents a significant upfront cost to practical system implementation, especially for large-scale deployment. In view of this, techniques that minimize the amount of site profiling are attractive. One way is to reduce the number of calibration points and the number of RSS samples collected at each point. In RADAR, the authors

observed that reducing the number of calibration points affects positioning accuracy but the impact is not as adverse as initially thought (Bahl et al., 2000). Using slightly more than half of the original radio map size, the accuracy only degrades about 10% compared to using the full radio map (Bahl et al., 2000). In terms of number of samples required, Bahl et al (2000) found that using just 3 RSS samples per calibration point achieved comparable positioning performance as using 20 RSS samples per point. Note that RADAR employed deterministic location estimation algorithm and the use of small number of RSS samples to represent the RF fingerprints might be appropriate. However, such approach might not be applicable for probabilistic techniques, which generally require sufficient RSS samples to model the RF fingerprint's probability distribution.

Interpolation has been proposed as an effective technique to increase the radio map density by filling the "missing" calibration points with interpolated values (Gami, Krishnakumar, & Krishnan, 2004; Krishnan et al., 2004; Li, Salter, Dempster, & Rizos, 2006). For example, Li et al. (2006) proposed the use of interpolation to increase the RF fingerprint density and reported comparable positioning performance to using denser empirical radio maps. The study also observed that beyond certain density, positioning performance does not improve much, suggesting there is a diminishing point to having highly dense radio maps (Li et al., 2006). Other works made similar observations (Krumm & Platt, 2003; Bahl & Padmanabhan, 2000; Xiaoyong & Qiang, 2007 and Gwon & Jain, 2004).

2.2.2. Model Based Calibration

Another calibration approach employs radio propagation models to compute RSS values at predefined locations to construct the RF fingerprints. Both deterministic and probabilistic radio maps can be calibrated using this approach (Brunato & Battiti, 2005; Roos, Myllymaki, Tirri, Misikangas, & Sievanen, 2002). An advantage of model-based calibration is it eliminates the expensive empirical site-survey, which is one of the main drawbacks of RF fingerprint methodology. Furthermore,

one can easily recalibrate the radio map at arbitrary granularity whenever the indoor environment and/or wireless network topology experience any changes.

Model-based radio map calibration assumes a suitable model exists that accurately describes RSS distributions of radio sources within the site. Most works in the literature employ empirical path loss models to calibrate the radio map (Alonso, Rodriguez, & Barbolla, 2009; Capulli, Monti, Vari, & Mazzenga, 2006). These models characterize RSS of radio signals as a function of distance from the transmitting antenna plus other environmental factors. An example is the log-distance path loss with lognormal shadowing model:

$$P_R(d) = P_R(d_0) - 10\alpha \ log\left({}^{d}/_{d_0}\right) + X_\sigma \qquad \text{(Eq. 2.1)}$$

Eq. 2.1 statistically models the strength of radio signal, as a function of propagation distance d. $P_R(d_0)$ is the reference received power at distance d_0 and usually determined via empirical measurements. Parameter α is the path loss exponent and is dependent on the specific propagation environment. In free space, α = 2, but in most indoor environments, α typically takes on larger values. X_σ is a Gaussian random variable with zero mean and standard deviation σ that characterizes shadowing effect around the transmitter.

Another variant of the indoor path loss model takes into account the effects of walls and floors

$$P_R(d) = P_R(d_0) - 10\alpha \ log\left({}^{d}/_{d_0}\right) - WAF - FAF \qquad \text{(Eq. 2.2)}$$

The *WAF* and *FAF* coefficients model the signal attenuation introduced by walls and floors penetrated by direct signal path (Barsocchi, Lenzi, Chessa, & Giunta 2009) and expressed as

$$WAF, FAF = \sum_{i=1}^{N} k_i l_i \qquad \text{(Eq. 2.3)}$$

where k_i is the number of penetrated wall or floor of type i, while l_i is the attenuation due to the wall or floor of type i e.g. brick, plaster, wood or concrete (Barsocchi et al., 2009).

It is clear from the equations that radio propagation models require detailed knowledge of the wireless channel in order to arrive at an accurate RSS estimate. They usually rely on extensive empirical RSS measurements and regression analysis to characterize the radio propagation environment. This represents additional cost to radio map calibration, although the degree of measurement effort could be lesser compared to empirical radio map calibration.

Another alternative is to use site-specific models that are based on the theory of electromagnetic-wave propagation, such as ray tracing (Akl, Tummala, & Li, 2006; Hatami, 2006). Ray tracing approach approximates the scattering of electromagnetic waves by simple reflection and refraction, where the degree of transmission and reflection of a signal through and off an obstacle is dependent on the complex permittivity of the obstacle (Akl et al, 2006). Hatami (2006) uses ray tracing simulation tool to model RSS distribution inside two buildings. The results were comparable with empirical RSS measurements and the author concluded that ray tracing could closely approximate actual data (Hatami, 2006). However, to achieve such accurate estimations, detailed knowledge of the environments, such as the building's floor plan and environmental parameters, must be known (Sarkar, Zhong, Kyungjung, Medouri, & Salazar-Palma, 2003). In practice, these environmental data are difficult or impossible to obtain. Moreover, site-specific models are generally more complex and computationally expensive (Sarkar et al., 2003).

Several works in the literatures investigate the feasibility of model-based radio map calibration. In RADAR, Bahl & Padmanabhan (2000) compared positioning performance between empirically derived radio maps and those based on radio propagation modelling technique. The latter produced median positioning error of

4.3m, which is 46% worse than the results obtained from empirical radio map (Bahl & Padmanabhan, 2000). Kwon, Dundar, & Varaiya (2004) observed similar results where the authors reported median accuracy of 3.2m using radio map calibrated using radio propagation modelling technique, compared to 1.7m using empirically derived radio map for their test site. The reduction in positioning accuracy is mainly due to the characteristics of the radio propagation model. Both works utilized the statistical path loss models due to their simplicity and computational efficiency. However, these models implicitly characterize all environmental influences into a single parameter, the so-called propagation exponent, regardless whether they can be separately recognized (Sarkar et al., 2003). As a result, RF fingerprints predicted by these models are less accurate, propagating the errors to location results.

2.3. Location Estimation Algorithm

Location estimation algorithm depends on the approach used to model the relationship between location and RF fingerprint when calibrating the radio map, which is either deterministic or probabilistic. This section gives an overview of the common location estimation algorithms used in the literatures.

2.3.1. Deterministic Algorithm

One of the popular deterministic algorithms is the Nearest Neighbour (NN). This algorithm solves the location estimation problem by comparing the unknown RF fingerprint with every reference RF fingerprint in the radio map to find the closest-matched record. This "closeness" is determined by computing the "difference" between unknown and reference RF fingerprints using some form of distance function i.e. *Dist(·)*.

Suppose the radio map contains a set of l reference RF fingerprints $\{F_1, F_2, \ldots, F_l\}$ that correspond to l discrete locations $\{L_1, L_2, \ldots, L_l\}$ calibrated during the offline

phase. Moreover, suppose the unknown RF fingerprint measured during the online phase is denoted as *S*. Assuming the system considers RSS from *N* transmitters, each reference RF fingerprint *i* in the database is expressed as $F_i = (p_1^i, p_2^i, \ldots, p_N^i)$, $i=1, 2, \ldots, l$, while the unknown RF fingerprint is expressed as $S = (s_1, s_2, \ldots, s_N)$. The NN algorithm selects reference RF fingerprint *j* that corresponds to the shortest distance in signal space:

$$\boldsymbol{Dist}(\boldsymbol{S}, \boldsymbol{F_j}) \leq \boldsymbol{Dist}(\boldsymbol{S}, \boldsymbol{F_i}), \forall \boldsymbol{i} \neq \boldsymbol{j} \qquad \text{(Eq. 2.4)}$$

The generalized distance formula between two vectors is as follows:

$$\boldsymbol{L_q} = \left(\sum_{\boldsymbol{k=1}}^{\boldsymbol{N}} |\boldsymbol{p_k} - \boldsymbol{s_k}|^{\boldsymbol{q}}\right)^{\boldsymbol{1/q}} \qquad \text{(Eq. 2.5)}$$

where $q = 1$ and $q = 2$ corresponds to Manhattan and Euclidean distance, respectively (Li et al., 2006). Consequently, coordinate of RF fingerprint *j* represents the location result.

Several variants exist for NN algorithm. Instead of choosing only the nearest neighbour, *K* nearest neighbours (*KNN*) are chosen whose locations are then averaged to yield the location result. The intuition is that locations that are nearby in space exhibit similar RF fingerprints and may have equal probability of being the correct location. In addition, RF fingerprints are inherently "noisy" due to fluctuations of RSS caused by dynamic changes within the radio propagation environment. Choosing *K* nearest fingerprints and averaging their coordinates effectively averages out this noise and can lead to better location estimates. Li et al. (2006) reported that using highly dense radio map and choosing $K = 3$ or 4 generally yield better location results. However, choosing too many neighbours can decrease the accuracy since some of them could be too far from the actual location (Bahl & Padmanabhan, 2000).

Another variant, known as Weighted *K* Nearest Neighbour (*WKNN*), computes the weighted average, rather than just the average, of *K* nearest neighbours. One approach is to use the inverse of signal distance L_q as the weights

$$\boldsymbol{w} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{l}} \frac{\mathbf{1}}{\boldsymbol{L_{qi}} + \boldsymbol{\varepsilon}} \cdot \boldsymbol{p_i} \qquad \text{(Eq. 2.6)}$$

where ε is a small constant to avoid dividing by zero (Ximei, Chao, & Jizhen, 2008). The idea is that closer RF fingerprints should have larger influence compared to farther ones on the location estimate.

Using the NN algorithm, Saha et al. (2003) observes that some test RF fingerprints are nowhere near the reference RF fingerprints but the algorithm provides a location estimate anyway. This is because NN algorithm blindly chooses the reference RF fingerprint whose centre is nearest in feature space to the unknown RF fingerprint without any consideration of the distribution of the data (Saha et al., 2003). To solve this problem, the authors proposed a slight modification to the NN classifier such that it reports such data as unclassifiable (Saha et al., 2003). This occurs when the sample RF fingerprint lies outside a region defined by two standard deviations around the mean RSS of the reference RF fingerprint (Saha et al., 2003).

Suppose reference RF fingerprint *i* in the radio map is represented by $M_i = \{p^i_1, p^i_2, \ldots, p^i_N\}$ and $D_i = \{\sigma^i_1, \sigma^i_2, \ldots, \sigma^i_N\}$, where *M* and *D* correspond to the mean and standard deviation of RSS samples, respectively. Given an unknown RF fingerprint $S = (s_1, s_2, \ldots, s_N)$, the modified NN classifier can be expressed mathematically as (Saha et al., 2003):

$$\begin{array}{lll} p^i_1 - 2\sigma^i_1 \leq & s_1 & \leq p^i_1 + 2\sigma^i_1 \\ p^i_2 - 2\sigma^i_2 \leq & s_2 & \leq p^i_2 + 2\sigma^i_2 \\ \vdots & \vdots & \vdots \\ p^i_N - 2\sigma^i_N \leq & s_N & \leq p^i_N + 2\sigma^i_N \end{array} \qquad \text{(Eq. 2.7)}$$

If the unknown RF fingerprint *S* lies outside this region for all reference RF fingerprints in the radio map, it is unclassifiable.

2.3.2. Probabilistic Algorithm

RF fingerprints can be modelled probabilistically. In this case, it is described in terms of conditional probability *P(F|L)* i.e. the probability that RF fingerprint *F* is observed at location *L*. By application of the Bayes theorem, the *posterior probability distribution P(L|F)*, can be computed as:

$$P(L|F) = \frac{P(F|L)P(L)}{P(F)} \quad \text{(Eq. 2.8)}$$

P(L|F) describes the probability of being at location *L* given the observed RF fingerprint is *F*. Thus, a maximum posterior probability yields the most probable location based on the observed RF fingerprint. For example, given two RF fingerprints *A* and *B*, the probabilistic classifier will choose location *A* over *B* if $P(L_A|F) > P(L_B|F)$ (Kaemarungsi, 2005). *P(L)* is the prior probability of being at location *L*. This term provides a principled way of incorporating additional information regarding the user mobility pattern into location estimate. For example, if a user was at location *L*, it is very likely that user will be at adjacent location in the future. Such information is useful to implement tracking. If the user's mobility profile is not available, then all locations can have equal probability. Meanwhile, *P(F)* does not depend on location variable and can be treated as a normalizing constant (Roos et al., 2002).

The term *P(F|L)* is known as the likelihood function, determined based on RSS distributions at each location during the radio map calibration phase, either via empirical RSS measurements or using radio propagation modelling technique and knowledge of the environment (Kaemarungsi, 2005). Roos et al. (2002) describe two methods to estimate the likelihood function, which are kernel method and histogram method.

2.3.2.1. Kernel Method

Consider a one-dimensional vector of m RSS samples measured from a radio source at location L i.e. $\{p_1, p_2, \ldots, p_m\}$. In the kernel method, a probability mass such as Gaussian distribution is assigned to a "kernel" around each element in p (Roos et al., 2002). Given a signal strength sample s in location L, the resulting likelihood function is an equally weighted sum of all probability mass kernel functions

$$P(s|L) = \frac{1}{m} \sum_{i=1}^{m} \left[\frac{1}{\sqrt{2\pi}\sigma} \, exp\left(-\frac{(s-p)^2}{2\sigma^2} \right) \right] \quad \text{(Eq. 2.9)}$$

where σ is an adjustable parameter that determines the width of the kernel (Roos et al., 2002). Extending the one-dimensional formula to multiple signal strength observations from N transmitters, the individual conditional probabilities are multiplied together, assuming the observations are independent of one another:

$$P(F|L) = P(s_1|L).P(s_2|L) \ldots P(s_N|L) \quad \text{(Eq. 2.10)}$$

2.3.2.2. Histogram Method

In the histogram method, the probability distribution is described in terms of discrete density functions. The method first defines several bins, which is a set of non-overlapping intervals. The sample data are grouped into one of the bins if their values fall between the bin's intervals. The number of samples in each bin is then normalized to yield the bin's relative frequency, which is used to describe the probability distribution. The number of bins, denoted k, and its size are important parameters that greatly affect the resulting density estimates. The larger the number of bins, the better the histogram can approximate the probability density function (Kaemarungsi, 2005).

2.4. Accuracy of RF Fingerprinting

Several works in the literature provide analytical frameworks to characterize positioning accuracy of RF fingerprint algorithm. This section summarizes the findings of these works.

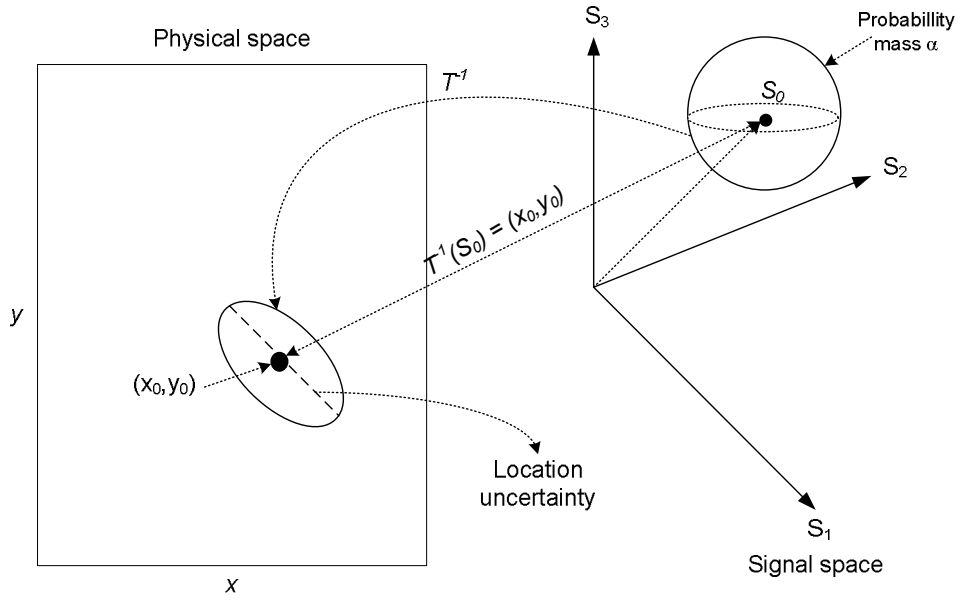


Fig. 2.1: Mapping uncertainty in signal strength space to uncertainty in location from (Krishnakumar & Krishnan, 2005)

Krishnakumar & Krishnan (2005) present a theoretical background for analysing fundamental limits of positioning accuracy using RSS measurements. The main intuition is that variation in measured RSS due to a change in location is indistinguishable from variation due to shadowing (Krishnakumar & Krishnan, 2005). Hence, any decision rule will map several locations in the neighbourhood of point (*x,y*) to (*x,y*), effectively resulting in uncertainty in location estimates (Krishnakumar & Krishnan, 2005). The authors derived a mathematical expression to quantify this uncertainty by mapping the uncertainty in signal strength space to location uncertainty in the (*x,y*) plane, as summarized in Fig. 2.1 (Krishnakumar & Krishnan, 2005).

An underlying assumption of RF fingerprint methodology is the array of RSS has direct mapping to location. Based on this assumption and on the continuity of signal strength measurement in a small region around the point of interest, Krishnakumar & Krishnan (2005) derive the structure of uncertainty in location space given the probability mass α in the signal strength space. Assume an array of N RSS is used to locate a terminal, where each signal strength element is a random variable that can be modelled as a normal distribution around a mean with variance σ_i, $i=1,2,\ldots,N$ (Krishnakumar & Krishnan, 2005). The idea is to define an α-region such that the total probability that the observed signal strength array due to a terminal located at some point in the region is α. It was shown that this uncertainty region is an ellipse with semi-axes given by (Krishnakumar & Krishnan, 2005):

$$r_{major,minor} = \sqrt{\frac{-2d}{(a+b) \pm \sqrt{(a-b)^2 + c^2}}} \qquad \text{(Eq. 2. 11)}$$

$$a = \sum_{i=1}^{N} \frac{t^2_{i1}}{\sigma^2_i}, b = \sum_{i=1}^{N} \frac{t^2_{i2}}{\sigma^2_i}, \; c = \sum_{i=1}^{N} \frac{2t_{i1}t_{i2}}{\sigma^2_i}, d = -R^2{}_N \qquad \text{(Eq. 2.12)}$$

$T = \{t_{ij}\}$ is the Jacobian of the mapping from location to signal strength measurements and is an n x 2 matrix, where

$$t_{i1} = \frac{\partial \bar{s_i}}{\partial x}, t_{i2} = \frac{\partial \bar{s_i}}{\partial y} \qquad \text{(Eq. 2.13)}$$

and R_N is a scaling factor that is related to α and given by

$$\alpha = \frac{\Gamma(\frac{N}{2}, \frac{R^2}{2})}{\Gamma(\frac{N}{2})} \qquad \text{(Eq. 2.14)}$$

where $\Gamma(\cdot,\cdot)$ is the incomplete gamma function (Krishnakumar & Krishnan, 2005).

Due to the assumption of signal strength continuity in the local neighbourhood, the partial derivatives t_{ij} are constant and T becomes a locally linear transformation from the n-dimensional signal strength space into (x,y)-plane (Krishnakumar & Krishnan, 2005). Based on the structure of the uncertainty region, the semi-major

axis, semi-minor axis and their geometric mean are defined as upper, lower and mean uncertainty, respectively, and these quantities were shown to be bounded (Krishnakumar & Krishnan, 2005).

Formulating the specific form of Jacobian matrix T is possible by considering the log-distance radio propagation model. The log-distance radio propagation model describes the mean signal strength at distance d from a transmitting antenna as:

$$\bar{S} = \overline{S_0 - K\,log(d)} \qquad \text{(Eq. 2.15)}$$

where S_0 is the reference signal strength (usually taken at distance of 1m from the transmitter for indoor radio propagation) and K is the propagation constant of the environment. Given N transmitters located at (x_i,y_i), $i = 1,2,\ldots,N$ and assuming different K for different transmitter, the mean signal strengths at location (x_i,y_i) are $S_i = S_0 - K_i\ log\ d_i$, $i=1,2,\ldots,N$ with $d_1 = \sqrt{(x - x_i)^2 + (y - y_i)^2}$. The resulting Jacobian matrix T is as follows (Krishnakumar & Krishnan, 2005):

$$t_{i1} = \frac{\partial \bar{s}_i}{\partial x} = \frac{-K_i(x - x_i)}{\{(x - x_i)^2 + (y - y_i)^2\}} \qquad \text{(Eq. 2. 16)}$$

$$t_{i2} = \frac{\partial \bar{s}_i}{\partial y} = \frac{-K_i(y - y_i)}{\{(x - x_i)^2 + (y - y_i)^2\}} \qquad \text{(Eq. 2.17)}$$

Based on their analytical framework, the authors made several interesting observations about the uncertainty and its dependence on various parameters of the location estimation problem (Krishnakumar & Krishnan, 2005):

a. The confidence level α indirectly affects the uncertainty measure through its effect on the factor R_N. Increasing the value of α increases R_N through their relationship shown in Eq. 2.14, and consequently increases the uncertainty region described by Eq. 2.11. As $\alpha \to 1$, $R_N \to \infty$; thus increasing the confidence level to a high value incurs disproportionate increase in the location uncertainty.

b. The uncertainty in location estimate is proportional to signal strength variance (Kaemurangsi, 2005). This dependence is explicitly described by

the term σ_i, $i=1,2,\ldots,N$ in the definitions of a, b and c in Eq. 2.12 and their proportional relationship to the axes of uncertainty ellipse described by Eq. 2.11.

c. From Eq. 2.16 and 2.17, the terms t_{i1} and t_{i2} are proportional to the propagation constant of the environment K. Using these terms in the formula for the axes of uncertainty ellipse in Eq. 2.11, the uncertainty is inversely proportional to the propagation constant. This is because large propagation constant implies rapid change in signal strength over distance; thus, a given variation in signal strength corresponds to smaller distance than with smaller propagation constant.

d. The authors noted that based on the propagation model used, the relationship between uncertainty and distance between transmitters is linear. This can be verified by expressing the distance expressions of t_{i1} and t_{i2} in Eq. 2.16 and 2.17 in terms of reference distance L, and use this in the definitions of a, b, and c of Eq. 2.12 to compute the uncertainty axes of Eq. 2.11. It can be seen that the uncertainty is proportional to L.

e. All the quantities that determine the uncertainty depend on the number of radio sources/transmitters i.e. N. From their simulations, the authors observed that the uncertainty decreases with additional transmitters while keeping the same sized area. However, if the coverage area increases together with additional transmitters, the uncertainty increases or remains stable in some cases.

Youssef & Agrawala (2003) proposed an analytical framework that quantifies the average distance error as a function of its probability of error i.e. the probability that the location determination technique will give an incorrect estimate. They formally proved that probabilistic techniques are more accurate than deterministic techniques (Youssef & Agrawala, 2003). This is because probabilistic techniques

implicitly reduce the effect of RSS variance, which is the source of error in location estimates. The variance causes asymmetrical distribution of RSS but not taken into account by deterministic algorithms.

Elnahrawy et al. (2004) explored the fundamental limits of using RSS for location determination in indoor environment by comparing several location determination techniques on the same experimental test bed. In particular, they investigated several *area-based* algorithms that can trade accuracy (the likelihood an object is within an area) for precision (the size of the returned area) and showed that these algorithms have similar fundamental performance (Elnahrawy et al., 2004). They then generalized their results by comparing against several deterministic and probabilistic algorithms (or *point based* algorithms) and observed striking similarity in performance graphs, apart from several Bayesian approaches that had higher location errors (Elnahrawy et al., 2004). Over a range of algorithms, approaches and sample sizes tested, the authors observed a median localization error of 10ft (Elnahrawy et al., 2004). A corollary to this result is that computationally simple algorithms that do not require many training samples are preferable because the performance of more complex algorithms is unlikely to be justified (Elnahrawy et al., 2004).

The results of these analytical works suggest there is a fundamental limit to positioning accuracy based on RSS using RF fingerprint algorithm. The main cause of error is the random signal strength fluctuations due to the effect of multipath propagation and shadowing. Shadowing distribution is lognormal with variance related to the signal strength (Kaemarungsi, 2005). This signal strength variance places some uncertainties on location estimate and reducing its influence is vital for improved positioning performance. Algorithms that take into account the signal strength variation, such as probabilistic technique can yield better results. However, such algorithms typically incur additional computational complexity and this trade-off must be understood (Krishnakumar & Krishnan, 2005). In some

cases, the marginal increase in positioning performance might not justify the added computational complexity.

2.5. Related Works

Early indoor positioning systems that employ RF fingerprint algorithm have to deal with two main challenges. One is the laborious and time-consuming radio map calibration, and another is the instability of RF fingerprints caused by environmental dynamics, which renders previously calibrated radio map invalid, thus affecting positioning accuracy of the system. To address these drawbacks, several state-of-the-arts proposed implementation methodologies capable of adapting the radio map to dynamic changes of the indoor environment (Atia et al., 2013; Gwon & Jain, 2004; Hyuk et al., 2006; Ivanov et al., 2008; Jie et al., 2005; Krishnan et al., 2004; Moraes & Nunes, 2006).

Jie et al (2005) proposed an implementation methodology that can automatically offset temporal variations in RF fingerprints to maintain signal-spatial integrity of the radio maps. Their system utilizes a network of sensors deployed at pre-determined reference locations within the site and programmed to collect RF fingerprints during operational phase of the system. However, prior to system deployment (or at time t_0), an initial radio map had to be calibrated which was done via site survey. Predictive functions that model the relationships between the initial radio map and RF fingerprints of the reference locations were developed. At later time period t_i $i \geq 1$, the system first obtains the real-time reference RF fingerprints provided by the sensors. Next, the predictive functions are applied using these reference RF fingerprints as inputs to yield a new radio map that already contains all necessary signal corrections. The authors proposed two approaches to build the predictive functions. One uses multiple linear regressions and another a general non-linear approximation algorithm based on model tree (Jie et al., 2005). Wang, Ma, Xu, & Deng (2011) adapted similar methodology in their work. Their

contribution is the use of artificial neural network to model the predictive functions. Fig. 2.2 illustrates the main idea of this technique.

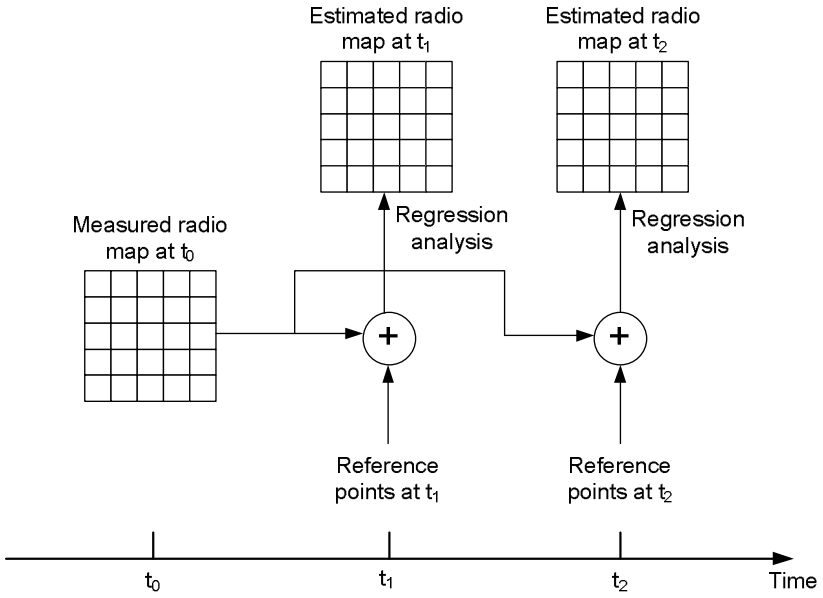


Fig. 2.2: Illustration of the temporally adaptive radio map methodology from (Jie et al., 2005)

One drawback of this approach is it still depends on an initial static radio map calibrated via manual site-survey as the basis for modelling the predictive functions. Another drawback of the approach is its inability to handle spatial variations caused by infrastructure changes, for example due to addition, removal or relocation of the radio sources. In this scenario, the initial static radio map which form the basis of the predictive functions is no longer valid, thus any corrections applied to it do not yield the desired results. The solution is to calibrate a new radio map to model the new predictive functions, which add to the system costs.

The LEASE system addresses the radio map auto-calibration and adaptability problem via an infrastructure-centric system implementation, using a combination of dedicated RF receivers and emitters (Krishnan et al., 2004). The system deploys multiple emitters, or location beacons, at known locations that broadcast wireless packets announcing their presence in the area. Meanwhile, a network of RF

receivers captures these beacon packets from multiple emitters, records their RSS values and unique identifiers, and forwards the readings to a location server. Using these measurements, the server computes the RSS of multiple emitters at several predefined locations via a non-parametric interpolation algorithm (Krishnan et al., 2004). For *k* emitters, the server computes *k* signal strength values for each location, effectively calibrating the radio map on-demand. The location server performs this calibration each time it detects significant variation in RSS reported by any RF receiver, or when any stationary emitter or receiver is added, removed or relocated. Thus, the radio map is constantly adapted to offset both temporal and spatial variations in the RF fingerprints. Fig 2.3 describes the overall architecture of the LEASE system.

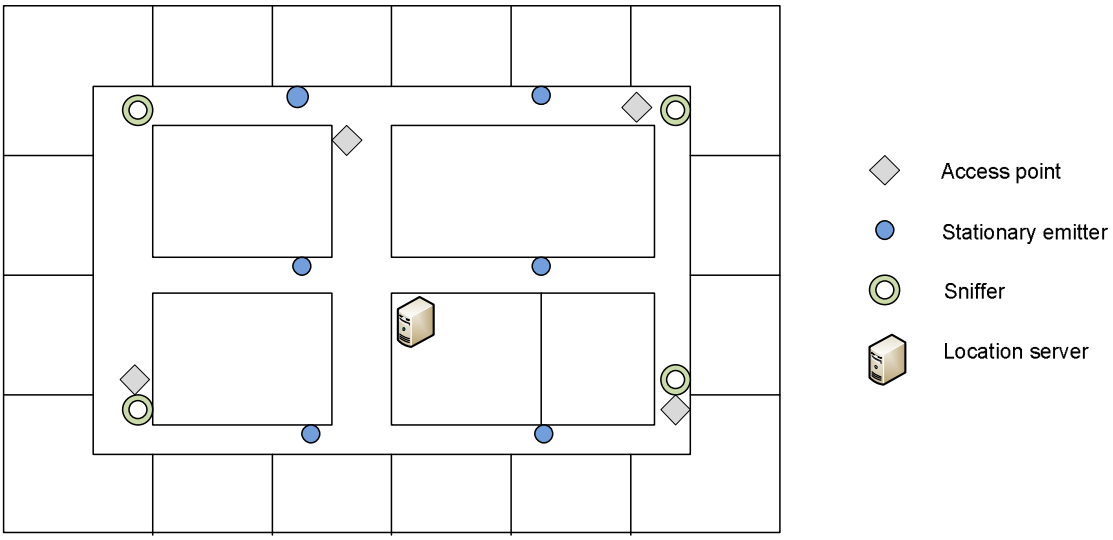


Fig. 2.3: System architecture of LEASE indoor positioning system from (Krishnan et al., 2004)

Moraes & Nunes (2006) proposed similar infrastructure-centric implementation using RF receivers distributed in the target area. Unlike LEASE, their system does not utilize dedicated RF emitters; instead, it uses existing networks of Wi-Fi APs as the location beacons (Moraes & Nunes, 2006). Moreover, the system knows the locations of both RF receivers and APs, so if any receiver or AP is added, removed or relocated, the system becomes aware of the spatial changes and can adapt the radio

map accordingly. Each receiver captures wireless packets from multiple APs, records their RSS and forwards the measurements to a location server. Based on these readings, the server calibrates the radio map using parametric path loss modelling technique. Each RF fingerprint is modelled using probability distribution to match its probabilistic location estimation algorithm. If the system detects significant statistical deviations in RSS measurements forwarded by the receivers, it recalibrates the radio map to offset for the temporal variations (Moraes & Nunes, 2006). An alternative scheme is to recalibrate the map periodically every T period (Moraes & Nunes, 2006). This ensures the radio map remains valid at most for T period. Fig. 2.4 describes their overall system architecture.

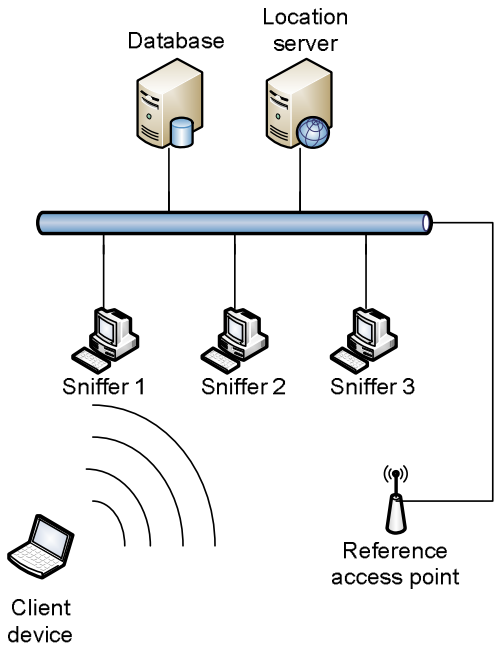


Fig. 2.4: Architecture of indoor positioning system described in (Moraes & Nunes, 2006)

A common theme in both works is the use of dedicated RF receiver infrastructure. These are typically realized using commodity wireless interface cards running suitable packet sniffing software, for example Linux-based Kismet (Dujovne, Turletti, & Filali, 2010). Although the prices of wireless interface cards have become very competitive in recent years, deploying the RF receivers incurs additional hardware costs.

Several state-of-the-arts recognized that commodity, off-the-shelf Wi-Fi APs are nothing more than wireless interface card with specialized management functions and it is possible to software-enhance the AP itself to implement the receiver functionality (Gwon & Jain, 2004; Hyuk et al., 2006; Barsocchi et al., 2009; Kao, Liao, & Lyu, 2010; Lo, Hsu, & Tseng, 2012; Atia et al., 2013). The AP could then passively measure RSS of neighbouring APs, and since they already connected to the network, forward these inter-AP RSS measurements to the location server for further processing. Such implementation methodology is very cost-effective because it eliminates the use of dedicated sensor infrastructure and the need to provide network access to these sensors.

All prior works reviewed so far adopted a server-centric implementation. Although this has the benefit of easing computing burden of client devices, such centralized architecture is not robust. Lorincz & Welsh (2007) proposed a decentralized approach to RF fingerprint positioning system called Motetrack. The authors deployed a network of positioning beacons; each one programmed to broadcast a subset of the radio map in a manner that minimizes per-node storage overhead while ensuring the information can still be recovered in the event of high beacon failures (Lorincz & Welsh, 2007). Client device wanting to locate itself just need to listen and decode the broadcast packets from several beacons and pieces together the information to make up a complete radio map, before applying the RF fingerprint matching algorithm to compute its position. By distributing the radio map information among several beacon nodes, the system is robust against any single node failure. In fact, the authors claimed that their system could tolerate up to 60% of beacon failures without severely degrading positioning accuracy (Lorincz & Welsh, 2007).

Another notable advantage of Motetrack system is it facilitates direct access of positioning information between client devices and beacons, unlike server-centric implementations that depend on communication networks for server-access. This

further improves fault-tolerant of the system by eliminating another potential failure point i.e. network breakdown. Moreover, by removing both backend server and networking infrastructure, the Motetrack system can be easily scaled to service larger areas and/or bigger user base. Its decentralized architecture also lends itself well to privacy-friendly positioning because client devices have full control over their positioning information. Whereas in server-centric systems, client devices are forced to exchange certain identifiable data with the backend server in order to access the positioning service, while in some server-centric implementations, the system can even passively locate and track client devices without their knowledge (Krishnan et al., 2004; Moraes & Nunes, 2006).

The Motetrack system uses specialized embedded wireless nodes called Mica motes that are battery-operated for easy placements at desired locations (Lorincz & Welsh, 2007). Although they simplify system deployment, these Mica motes represent additional costs to the system. Gschwandtner & Schindhelm (2011) proposed a similar but more cost-effective solution that uses Wi-Fi APs to realize the location beacon infrastructure. This is accomplished by embedding the positioning data into vendor specific information element (IE) fields of Wi-Fi's beacon management frames, so when the AP transmits these frames, the positioning data are broadcasted as well (Gschwandtner & Schindhelm, 2011). The IEEE802.11 standard allocates this special field to allow devices to carry non-standard information within a predefined format to maintain interoperability between devices. Typical usage is to embed special signalling message that describes extended hardware options to enhance operation between devices made by the same vendor (Gschwandtner & Schindhelm, 2011). Additionally, it provides a convenient means for MS to receive arbitrary data without having to associate and authenticate with the APs, effectively offering free communication channels.

Both Motetrack and the system in (Gschwandtner & Schindhelm, 2011) require the

radio map to be calibrated prior to partitioning it into smaller subsets and programming them for broadcast by the beacons. This means the radio map remains static and unable to adapt to the dynamicity of the indoor radio propagation environments, leading to the degradation of the system's positioning performance. To update the positioning information requires calibration of a new radio map and reprogramming of the location beacons. These tasks are costly and to carry them out frequently is impractical.

A characteristic of decentralized positioning systems is the client-centric positioning model whereby client devices compute their own locations. A conventional algorithm would require a full search of the radio map to find the most probable location where the unknown RF fingerprint might originate. Doing so is very computationally expensive, especially if the radio map has high RF fingerprint density. This would place large computing burden on the already resource-limited client device and quickly drain its battery power. One way to minimize computing burden of client devices is to reduce the density of reference RF fingerprints, so with less fingerprints in the radio map to search, the less processing the device needs to perform. However, this generally reduces positioning accuracy of the system.

Youssef, Agrawala & Udaya Shankar (2003) proposed a clustering-based algorithm to reduce the fingerprint search space during location computations. The authors defined a cluster being locations that contain a subset of the APs, called the cluster key, and grouped RF fingerprints that match the cluster key into the same cluster. Given an unknown RF fingerprint, the algorithm first finds the appropriate cluster based on the cluster key, and then searches only those RF fingerprints within the cluster to find the closest-match, hence reducing the search space considerably. The authors reported an order of magnitude saving in computing costs compared to conventional RF fingerprint algorithm (Youssef et al., 2003).

An alternative positioning algorithm uses artificial neural network (ANN) (Battiti, Nhat, & Villani, 2002). Through supervised learning using reference RF fingerprints as training samples, a neural network learns the relationship between radio fingerprints and their associated locations, and generalizes this knowledge in an appropriate manner when presented with new radio fingerprints not present in the training set (Kaemarungsi, 2005). ANN enables instantaneous location estimations because it does not require any explicit radio map search. For example, a multi-layer perceptron (MLP) neural network with 3 input nodes, 8 hidden nodes and 2 output nodes with training strategy using second-order derivative was used (Battiti et al., 2002). The authors reported best average error of 1.52m using 300 samples and about 4000 iterations to train the network (Battiti et al., 2002).

Both the works in (Youssef et al., 2003) and (Battiti et al., 2002) achieved the reduction in positioning computation by first extracting the useful information from an initial radio map and then using this reduced format to solve for locations in later periods. This implies the methodologies work best for static radio map. For radio maps that are dynamically calibrated on-demand, these methodologies are less effective since the clustering and neural learning must be repeatedly applied. Due to this, there may not be any net improvement in overall computing costs.

Recent works in the literature advocated an innovative radio map calibration via collaborative efforts from many casual surveyors, a technique known as crowdsourcing. The idea is each surveyor only maps small part of the area, so individually the task is less tedious. In addition, many surveyors map the area at different points of times to ensure the radio maps stay current. Popularity of crowdsourcing is mainly due to the pervasive availability of 802.11-capable consumer device, such as smart phones and tablet computers. By running special client software, each device can function as RF fingerprint collection tool.

A typical process is to first identify the location by entering its coordinate (Phillip, 2008) or pointing to a pre-supplied map (Bhasker, Brown, & Griswold, 2004). The software then captures the RF fingerprints at that point of time, and forwards them to a server to combine with RF fingerprints from other surveyors. Kim, Chon, & Cha (2012) proposed an autonomous methodology to calibrate the radio map using low-cost inertial sensors embedded in smart phones and dead reckoning technique to compute the user's current location. This allows the system to auto-calibrate RF fingerprints of the target area without explicit participation from users.

Crowdsourced radio map is both cost-effective and practical because it does not involve any changes to existing Wi-Fi installations. However, such organic approach introduces other sets of challenges, namely conveying uncertainties, determining when user input is required and discounting erroneous and stale data (Park et al., 2010).

2.6. Summary

RF fingerprint algorithm comprises of two main components. One is a model that describes the spatial distribution of the site's RF fingerprints, or the so-called radio map. Another is a positioning algorithm that searches the radio map for locations that exhibit similar characteristics as the unknown RF fingerprints. The search algorithms can be either deterministic or probabilistic, depending on how the radio map is set up.

The RF fingerprint methodology can yield an average positioning accuracy of 10ft, which is generally sufficient for many indoor location based applications, although it is unlikely that the algorithm can produce highly accurate location results. Its main source of error stems from shadowing effect, which is inherent in typical indoor environments. Shadowing causes the RSS to fluctuate randomly, leading to RF fingerprint variations. Unfortunately, fingerprint variations caused by shadowing are indistinguishable from location-related variations. Techniques or

algorithms that can minimize or take into account the shadowing effects stand better chance of yielding better location results.

In terms of system implementation, RF fingerprinting suffers from several challenges, such as the tedious and time-consuming radio map calibration and susceptibility of the RF fingerprints to environmental dynamics of the indoor environment, causing its signal-spatial integrity to deteriorate over time. The literature has extensively explored these challenges. Of particular interest, several state-of-the-arts proposed innovative system implementations capable of calibrating the radio map on-demand to tackle these challenges.

Chapter 3 : A Self-Modelling Location Beacon

3.1. Introduction

A fundamental attribute of adaptive RF fingerprint methodology is the ability to maintain the radio map's signal-spatial integrity at all times. This implies the system must be able to detect if the RF fingerprints have degraded, and if so, calibrate a new one automatically. State-of-the-arts implementations detect radio map degradation by comparing the real-time RF fingerprints with previous records and observing for significant variations (Moraes & Nunes, 2006), and mathematically post-process these real-time RF fingerprints to calibrate new radio maps (Jie et al., 2005, Krishnan et al., 2004, Moraes & Nunes, 2006). In contrast, this work models the RSS distributions of radio signals around their transmitting antennas in real-time to characterise the dynamicity of indoor wireless channel, and uses the adaptive channel models to simulate RF fingerprints at various grid-points inside the target area to calibrate the radio map.

This thesis introduces a novel location beacon capable of self-modelling its path loss radio propagation in real-time (Mohd Sabri & Arslan, 2011). A network of these location beacons, realized using commodity Wi-Fi APs, forms the underlying reference infrastructure of the proposed indoor positioning system. Each location beacon constantly broadcasts its coordinate and coefficients that model its signal strength distribution; this work employs the path loss model with lognormal shadowing to characterise this distribution (Mohd Sabri & Arslan, 2011). Client devices listen for these broadcast packets from multiple location beacons, parse their coordinates and path loss coefficients, and use these parameters to compute the RSS of the detected location beacons at various grid-points inside the indoor area to construct new radio maps prior to solving for locations.

In order to self-calibrate its path loss radio propagation coefficients, the location

beacon must first acquire empirical RSS measurements from several points around its transmitting antenna. It achieves this through collaborative assistance from neighbouring location beacons by leveraging the network discovery feature of Wi-Fi protocol; Chapter 5 describes the implementation details of this feature. The next step is to fit the path loss model using the empirical RSS measurements as a function of propagation distances, and minimize the error in a least squared sense to calibrate the model's coefficients. By constantly self-modelling its path loss radio propagation channel, the location beacon implicitly learns the dynamic changes affecting its signal strength distribution. Consequently, the simulated RF fingerprints based on these adaptive channel models reflect the actual signal-spatial correlations of the site, resulting in more accurate radio maps.

3.2. Modelling the Indoor Radio Propagation Channel

As radio signal propagates away from the transmitting antenna, its strength decays as a function of propagation distance and properties of the propagation environment. However, for reliable wireless communications between two antennas, the transmitted radio signal must arrives at the receiver with sufficient strength in order for the receiver to distinguish it from noise. Therefore, the ability to predict signal strengths of wireless channels is crucial for system and network design (Sarkar et al., 2003). Since actual site measurements are expensive, radio propagation models provide a more practical means for analysing the performance of wireless channels and estimating the strengths of radio signals at arbitrary distance from the transmitting antenna.

Several mathematical models have been proposed in the literature to characterise radio propagation channels. For indoor radio propagation, the path loss model is a popular choice (de Souza & Lins, 2008):

$$PL(d) = PL(d_0) + 10\alpha \log_{10}(\frac{d}{d_0}) \qquad \text{(Eq. 3.1)}$$

This model characterises radio signal attenuation as a function of propagation distance d relative to the attenuation at a reference distance d_0. For indoor environment, d_0 is typically set at 1m. Parameter α is the path loss exponent that describes the rate of signal attenuation. In free space, $\alpha = 2$. In hallways with line-of-sight between transmitter and receiver, α takes a value between 1.5 – 1.8, and when the receiver is located in another room, α generally takes on higher value (Sarkar et al., 2003). Additionally, α varies with frequency and is dependent on building materials of the particular environment (Sarkar et al., 2003).

Path loss is the difference between transmit and receive power. With simple algebraic manipulation:

$$PL(d) = P_T - P_R(d) \quad \text{(Eq. 3.2)}$$

$$P_R(d) = P_T - PL(d) \quad \text{(Eq. 3.3)}$$

$$P_R(d) = P_T - PL(d_0) - 10\alpha \log_{10}(\frac{d}{d_0}) \quad \text{(Eq. 3.4)}$$

$$P_R(d) = P_{REF} - 10\alpha \log_{10}(\frac{d}{d_0}) \quad \text{(Eq. 3.5)}$$

Eq. 3.5 denotes the strength of received signal as a function of transmitter-receiver distance d. P_{REF} is the RSS at reference distance d_0 from the transmitting antenna and is typically determined through empirical measurements, thus known *a priori*. Alternatively, P_{REF} can be estimated by subtracting the path loss at d_0 from transmit power:

$$P_{REF} = P_T - PL(d_0) \quad \text{(Eq. 3.6)}$$

At distance $d_0 = 1m$, it is reasonable to assume free space condition exists between transmitter and receiver, thus the Free Space Path Loss model can be used (Debus, 2006):

$$PL(d) = 32.45 + 20\log(d) + 20\ \log(f) \quad \text{(Eq. 3. 7)}$$

In Eq. 3.7, distance d is in km while transmit frequency f is in MHz. For Wi-Fi AP operating at 2.4GHz frequency and 0dBm (or 1mW) transmit power, P_{REF} is approximately -40 dBm.

Eq. 3.5 describes an isotropic RSS distribution around the transmitting antenna. In reality, propagation paths at two different points with the same transmitter-receiver distance may be subjected to vastly different propagation environments with each one altering the strength of radio signals in stochastic manner, resulting in signal strength distribution that appears anisotropic. Such phenomenon is known as shadowing (Kaemarungsi, 2005). To reflect this shadowing contribution on the RSS characteristics, an additional $X(0,\sigma)$ term, which is a Gaussian random variable with zero mean and σ standard deviation, is added to Eq. 3.5 to yield:

$$P_R(d) = P_{REF} - 10\alpha \log_{10}\left(\frac{d}{d_0}\right) + X(0,\sigma) \qquad \text{(Eq. 3.8)}$$

In Eq. 3.8, coefficients P_{REF}, α and $X(0,\sigma)$ statistically describe RSS of radio signals at arbitrary distance d around the transmitting antenna. To calibrate these coefficients, obtain the empirical RSS measurements at various distances d around the transmitter and apply linear regression to minimize the squared errors between measured and estimated values.

Consider the setup where the location beacon is placed at location (x_p, y_p) in the target area and RSS_i denotes its empirical RSS measurement taken at location (x_i, y_i), where $i \neq p$. Assume there are L of such empirical RSS measurements i.e. $\{(x_i, y_i), RSS_i\}; i \in L, L > 2$. The propagation distance d_i between the location beacon and the ith measurement point is deterministic and can be calculated from:

$$d_i = \sqrt{\left(x_p - x_i\right)^2 + \left(y_p - y_i\right)^2}\ ;\ i \in L; i \neq p \qquad \text{(Eq. 3.9)}$$

Meanwhile, the errors between actual RSS as measured by the ith receiver and the estimated value predicted by the path loss model are given by:

$$err_i = RSS_i - (P_{REF} - 10\alpha \log_{10}(d_i));\ i \in L; i \neq p \quad \text{(Eq. 3.10)}$$

Eq. 3.10 is an over-determined linear system with L equations and two unknowns, and can be solved using linear regression by minimizing the sum of squared errors (Alonso et al., 2009). Rewriting Eq. 3.10 into matrix form yields:

$$\begin{pmatrix} err_1 \\ \vdots \\ err_L \end{pmatrix} = \begin{pmatrix} RSS_1 \\ \vdots \\ RSS_L \end{pmatrix} - \begin{pmatrix} 1 & -10\,log_{10}(d_1) \\ \vdots & \vdots \\ 1 & -10\,log_{10}(d_L) \end{pmatrix} \begin{pmatrix} P_{REF} \\ \alpha \end{pmatrix} \quad \text{(Eq. 3.11)}$$

Let's denote Eq. 3.11 using the following vector notation

$$\boldsymbol{e} = \boldsymbol{f} - \boldsymbol{X}\boldsymbol{c} \quad \text{(Eq. 3.12)}$$

The least squared estimator computes coefficient $\boldsymbol{c}$ as (Alonso et al., 2009):

$$c = (X^T X)^{-1} X^T f = \begin{pmatrix} P_{REF} \\ \alpha \end{pmatrix} \quad \text{(Eq. 3.13)}$$

By substituting P_{REF} and α into Eq. 3.10, the standard deviation of residual errors, or σ, can be computed as follows:

$$\sigma = \sqrt{\frac{1}{L} \sum_{i=1}^{L} (err_i - \overline{err})^2} \quad \text{(Eq. 3.14)}$$

Once the parameters P_{REF}, α and σ have been calibrated, the RSS at arbitrary locations around the location beacon can be computed. First, distance d between the location beacon and target location is computed using Eq. 3.9. Substituting d into path loss model of Eq. 3.5 yields the average RSS i.e. P_{AVG} of the target location. To account for the shadowing effect, the instantaneous received power $P_R(d)$ is computed using Gaussian probability distribution function with P_{AVG} mean and standard deviation σ:

$$P_R(d) = X(P_{AVG}(d), \sigma) = f(x; 0, \sigma) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{P_{AVG}}{\sigma}\right)^2} \quad \text{(Eq. 3.15)}$$

3.3. Location Estimation Using Adaptive Radio Map

In conventional RF fingerprint algorithm, one frequently encountered issue is the radio signals mismatch between the unknown and reference RF fingerprints. For example, when calibrating the static radio map during the offline phase, the reference RF fingerprints may comprise of *x* number of RSS components. Since then, radio transmitters might be added, removed or relocated, or radio signals from particular transmitters might be attenuated due to changes in the radio propagation environment not present during the offline phase. As a result, the unknown RF fingerprint can contain lesser or greater number of RSS components. Usually, the algorithm filters out the "mismatched" RSS components from either the reference or unknown RF fingerprints, or both, prior to searching for the best match, depriving the search algorithm of additional information that might be useful.

The adaptive RF fingerprint algorithm proposed in this thesis addresses this drawback. Prior to solving for location, client device first scans for nearby location beacons and measures their RSS at the unknown point to establish its RF fingerprint. Next, it calibrates a new radio map using the radio propagation models broadcasted by the detected location beacons. Therefore, the reference RF fingerprints in the radio map always comprises of the same RSS components as the unknown RF fingerprint, which means the search algorithm always has complete information for finding the best match. Finally, client device searches the newly calibrated radio map to find the closest-matched RF fingerprint, whose coordinate is returned as the location result.

Consider the setup illustrated in Fig. 3.1, where a two-dimensional space R containing C discrete points identifiable by $c_i = (x_i, y_i); i \in C$ and M location beacons opportunistically placed within R at locations $B_j = (x_j, y_j); j \in M$ where $c(x,y) \neq B(x,y)$. Each location beacon periodically broadcasts data packets containing

its location coordinate and path loss coefficients that model its RSS distribution.

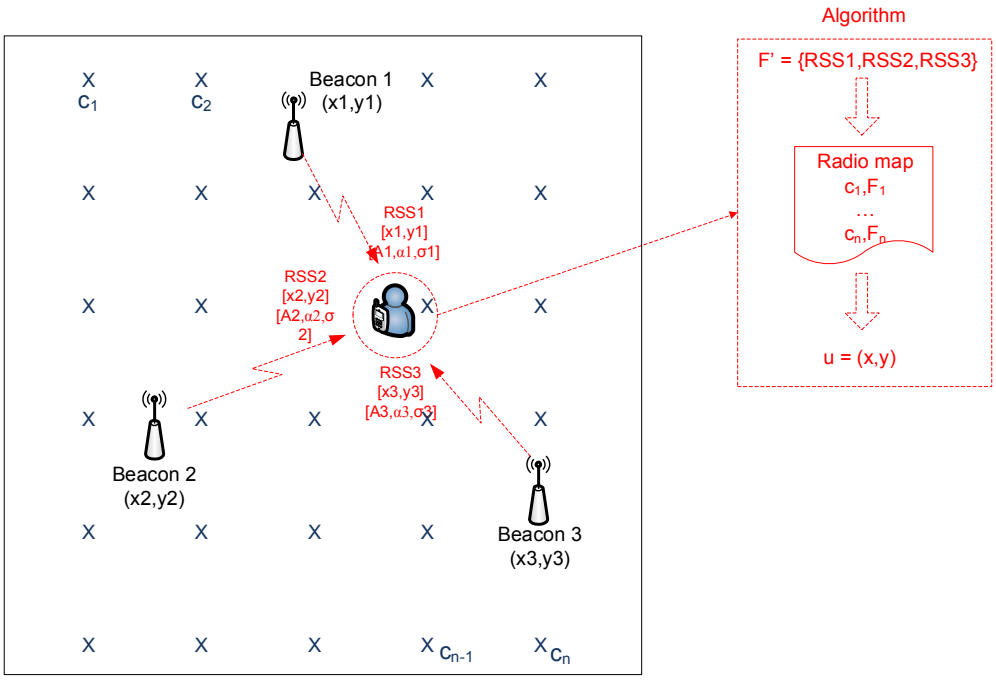


Fig. 3.1: Illustration of the proposed RF fingerprint indoor positioning system setup

At an unknown location *u(x,y)* in *R*, a location sensor acquires the RSS of nearby location beacons by passively listening and measuring their broadcasted data packets, thus forming the unknown RF fingerprint *F' = {RSS'1, RSS'2, ...}*. Note that the unknown RF fingerprint need not comprise of RSS from all *M* beacons because radio signals of some beacons might be attenuated well below the sensor's receive sensitivity threshold at *u(x,y)*. For each detected radio beacon in *F'*, its RSS is computed for every location in *C* using Eq. 3.5, Eq. 3.9 and Eq. 3.15 with the help of the location beacon's coordinate and path loss coefficients obtained from the data packets, effectively calibrating the radio map. Since physical radio receivers have finite sensitivity RX_{sens}, the receiver cannot detect radio signal with strength below this threshold; hence, the computed RSS values lower than RX_{sens} are replaced with RX_{sens}.

The next step is to estimate the location of *u(x, y)* by searching the simulated radio map for an entry that best resembles the unknown RF fingerprint *F'*. Chapter 2

reviews several methods of classifying the best match depending on the RF fingerprint characteristics of the radio map. This work uses the deterministic KNN algorithm, with $K = 4$ for best result (Li et al., 2006), since the calibrated radio map is also deterministic. The algorithm starts by computing the "difference" between unknown RF fingerprint F' and every entry of the radio map using a Euclidean distance metric. Next, K simulated RF fingerprints with the shortest distances (or nearest) to the unknown RF fingerprint are chosen, where their coordinates are averaged and returned as location result for $u(x, y)$.

3.4. System Simulation & Discussion

This work uses Matlab version R2103a to simulate various test scenarios, such as varying wireless channel conditions and changing location beacon topology in order to analyse the performance of the proposed adaptive RF fingerprint algorithm. The simulation is run on an Intel-based computer with 2.8GHz processor and 4GB RAM running 32-bit version of Microsoft Windows 7 OS. This section describes the simulation setup and test scenarios, and discusses the results in terms of positioning error and computing time relative to conventional RF fingerprint algorithm that uses static radio map calibrated in the offline phase.

3.4.1. Simulation Setup

Consider a 2-dimensional space R of dimensions x-by-y meters comprising of C distinct points with spatial granularity of k meter. Each point in C represents one calibration location with coordinate $(c_{i,x}, c_{i,y}); i \in C$. Within R, a total of M location beacons are deployed at locations $(B_{i,x}, B_{i,y}); i \in M$. Their radio signals propagate omni-directionally from the transmit antenna along the x-y plane and are modelled using the log-distance path loss with lognormal shadowing model i.e. $RSS_i(d) = P_{REF,i} - 10\alpha_i \log_{10}\left(\frac{d}{d_0}\right) + X(0, \sigma_i); i \in M.$ Meanwhile, the location sensor can exist at arbitrary point (u_x, u_y) in R. The simulation further assumes the wireless transceiver

design of both location beacon and sensor is equivalent; in particular, they have the same receive sensitivity threshold RX_{sens}.

$P_{REF,i}$ is the received power at reference distance d_0 = 1m from the *i*th location beacon. Its value depends on transmit power, as well as geometries and gain of the communicating antennas. Because the location beacon and sensor are assumed to have equivalent transceiver hardware, P_{REF} is a constant and its value can be determined using Eq. 3.6 and 3.7. Meanwhile, coefficients α_i and σ_i model the radio propagation characteristics of the *i*th location beacon. Their values depend on the site-specific dynamics; in an ideal world, $\alpha = 2$ and $\sigma = 0$. Several studies have empirically characterized α and σ for their respective test sites and the results suggest that α varies between 2.18 and 5.22, while σ varies between 3 and 16.3 dB (Andersen, Rappaport, & Yoshida, 1995; Seidel & Rappaport, 1992). To simulate the real-world radio propagation channels, path loss coefficients of the *i*th location beacon are assigned random values within these ranges.

A total of *T* positioning tests are simulated. For each test, the location sensor is assigned random coordinate (u_x,u_y) in *R* to denote its actual location. Next, the RSS of *M* location beacons are computed at this point to simulate the unknown RF fingerprint *F'* using the beacons' simulated path loss coefficients. The last step is to estimate the location sensor's coordinate (u'_x,u'_y) based on its unknown RF fingerprint *F'* using the proposed adaptive RF fingerprint algorithm, as well as the static RF fingerprint algorithm for comparative analysis. For both algorithms, the KNN algorithm with $K = 4$ is used as the search function. To access the performance of both algorithms, the positioning error being the difference between (u_x,u_y) and (u'_x,u'_y) in Euclidean distance sense is computed using Eq. 3.16 and plotted using Cumulative Distribution Function (CDF) over all *T* iterations, while their computing costs in terms of average processing times are recorded using Matlab's tic-toc function.

The static RF fingerprint algorithm requires a radio map calibrated in the offline

phase for use throughout all positioning tests in the online phase. To construct this static radio map, the *i*th location beacon is first assigned path loss coefficients to simulate its offline wireless channel conditions i.e. $\alpha_{offline,i}$, $\sigma_{offline,i}$ prior to computing its RSS values at all calibration points *C* in *R*.

A novel feature of the proposed location beacon is the ability to self-calibrate its path loss coefficients in real-time. To simulate this feature, RSS values of the *i*th location beacon are first computed using its assigned path loss coefficients α_i and σ_i at coordinates of the other *M-1* location beacons. This is followed by fitting the path loss model to the computed RSS values as a function of inter-beacon distances and minimizing the errors using a least square estimator to produce the estimated path loss coefficients α'_i and σ'_i. Finally, RSS values of the *i*th location beacon are computed using the estimated path loss coefficients for all calibration points *C* in *R* to construct the adaptive radio map prior to searching for the best RF fingerprint match to solve for location.

3.4.2. Results & Discussion

3.4.2.1. Positioning Performance as a Function of Varying Wireless Channel

The first simulation investigates the positioning system performance under fluctuating radio signal strengths caused by varying wireless channel dynamics. Two channel conditions are simulated. The first condition assumes a stationary indoor environment where the signal strengths of the location beacons remain relatively stable over time. To simulate this condition, path loss coefficients of all location beacons are assumed constant throughout all *T* online positioning tests. In addition, the offline path loss coefficients for calibrating the static radio map are assumed equal to the online path loss coefficients i.e. $\alpha_{offline,i} = \alpha_{online,i}$ and $\sigma_{offline,i} = \sigma_{online,i}$. The second condition assumes a non-stationary or "busy" indoor wireless environment where the location beacons constantly experience

dynamic changes that affect their RSS distributions in stochastic manner. To simulate this condition, path loss coefficients of all location beacons are randomly re-assigned with values based on the typical range described in previous section for each iteration of the online positioning tests.

A test site R of dimension 50m x 50m is defined and partitioned into C points with granularity of $k = 1m$. A total of $M = 5$ location beacons are opportunistically positioned in R as to achieved a well-balanced network topology as follows: $B_1 = (13,13)$, $B_2 = (37,13)$, $B_3 = (25,25)$, $B_4 = (13,37)$ and $B_5 = (37,37)$. A total of $T = 1000$ online positioning tests are performed. Fig. 3.2 illustrates the simulation setup and Fig. 3.3 plots the simulation results.

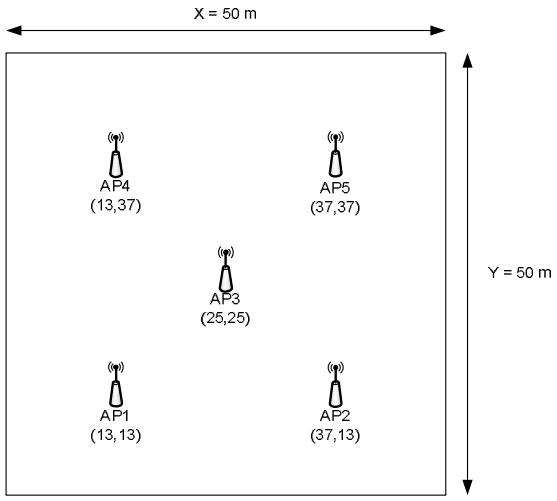
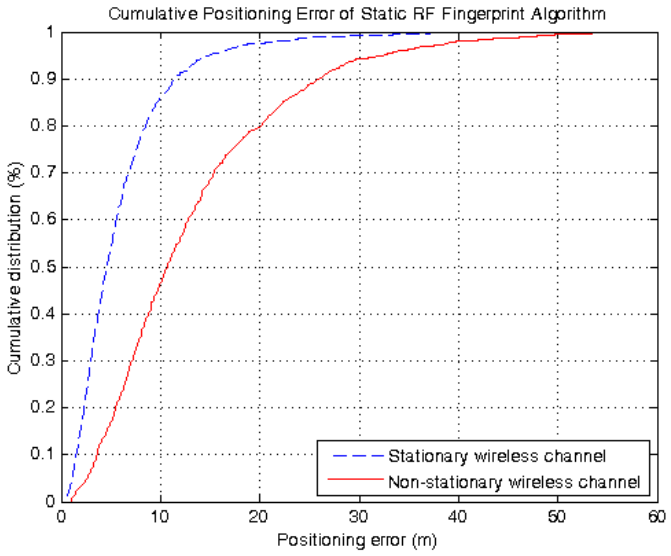


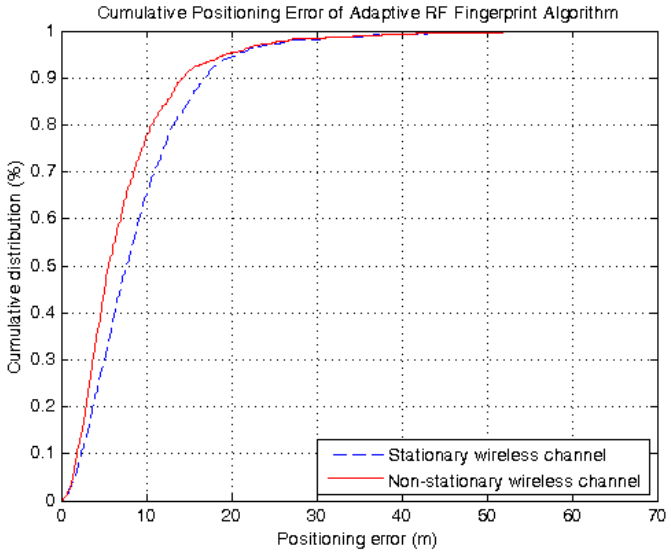
Fig. 3.2: Simulation setup to investigate system performance under varying wireless channel conditions

Fig. 3.3(a) plots the cumulative positioning error of the static RF fingerprint algorithm. It yields the best positioning performance under stationary wireless channel simulation scenario with median positioning error less than 5m. This is expected because the wireless channel of the test site continues to be relatively stable over time, so the signal-spatial integrity of the static radio map calibrated during the offline phase remains valid in the online phase. On the other hand, its positioning performance severely degrades as the wireless channel between the

offline and online phase differs. As shown by the non-stationary simulation curve in Fig. 3.3(a), its median positioning error doubles to about 10m.



(a)



(b)

Fig. 3.3: Cumulative positioning error of static and adaptive RF fingerprint algorithm as a function of varying wireless channel conditions

In comparison, positioning performance of the adaptive RF fingerprint algorithm is relatively consistent, with median positioning error of about 5m, for both wireless

channel simulation scenarios as shown in Fig. 3.3(b). This is due to the self-modelling location beacons introduced in this work. Collectively, they provide the system with real-time wireless channel information, so the algorithm becomes aware of dynamic changes happening within the site. Equipped with this information, the algorithm can simulate a more accurate radio map as the basis for its RF fingerprint search. As a result, it yields consistent location estimates since the signal-spatial variations between the unknown RF fingerprint and the reference RF fingerprints of the radio map are minimized.

3.4.2.2. System Performance as a Function of Changing Location Beacon Topology

The second simulation investigates the positioning system performance as a function of changing location beacon topologies. This is because the proposed solution is realized using Wi-Fi networks, and in practice such networks can be re-planned where new APs are commissioned to improve capacity, or existing ones removed or relocated to minimize co-channel interference and improve bandwidth. Changing the AP topologies would alter the signal-spatial correlations that model the RF fingerprints of the site. For conventional RF fingerprint positioning systems, this usually results in degradation of their positioning accuracy because the static radio map is no longer valid.

The same test site R of 50m x 50m in dimension with C calibration points of spatial granularity $k = 1m$ is assumed. This simulation evaluates three different topologies corresponding to addition, removal and relocation of location beacons. For each topology, T online positioning tests are performed and the cumulative positioning errors of static and adaptive RF fingerprint algorithms are plotted to compare their positioning performance. Furthermore, the wireless channel is assumed stationary in order to minimize the contributions from radio propagation-specific errors and to simplify analysis. This means the path loss coefficients stay the same from calibrating the static radio map in the offline phase until throughout the online

positioning tests.

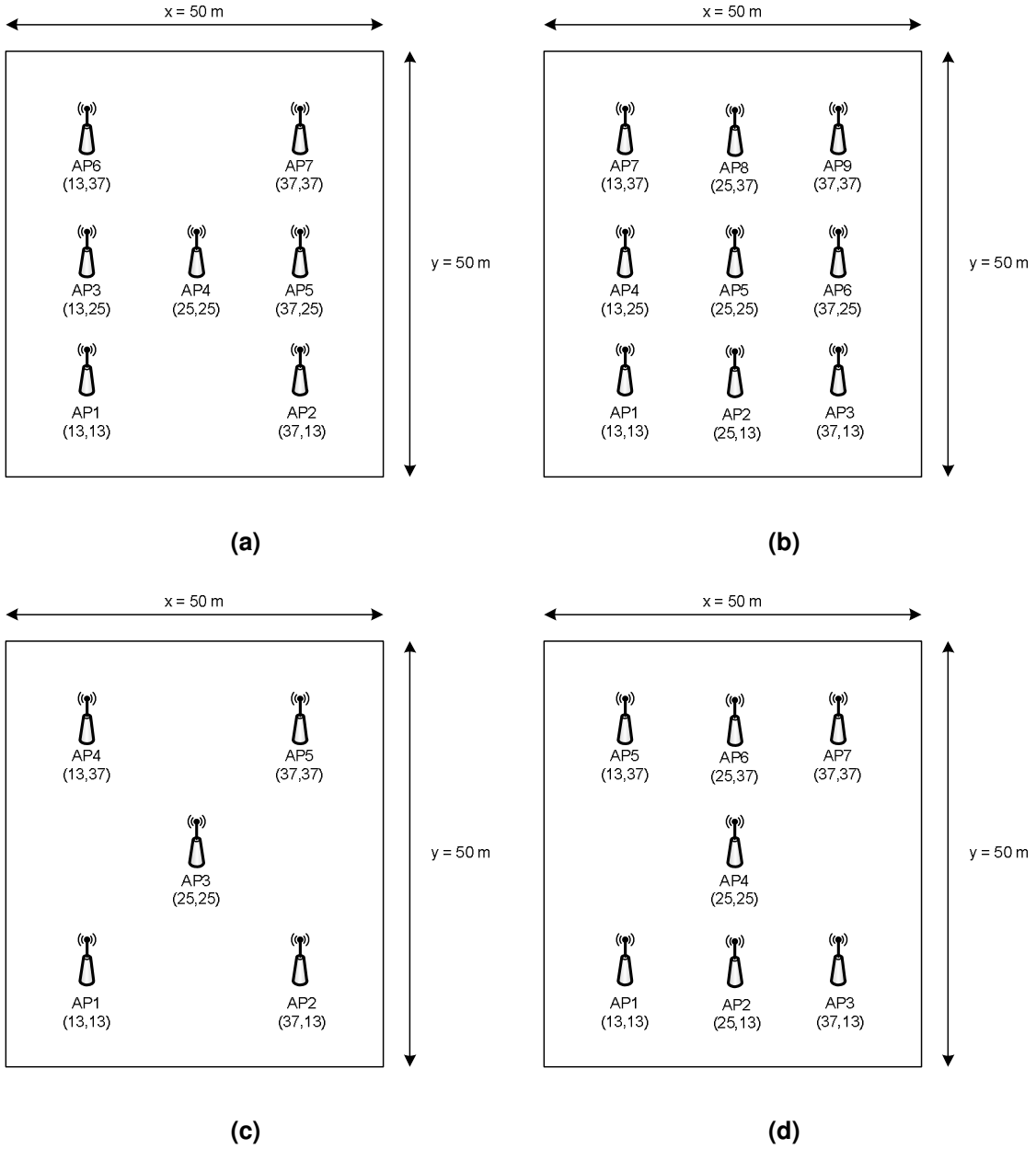
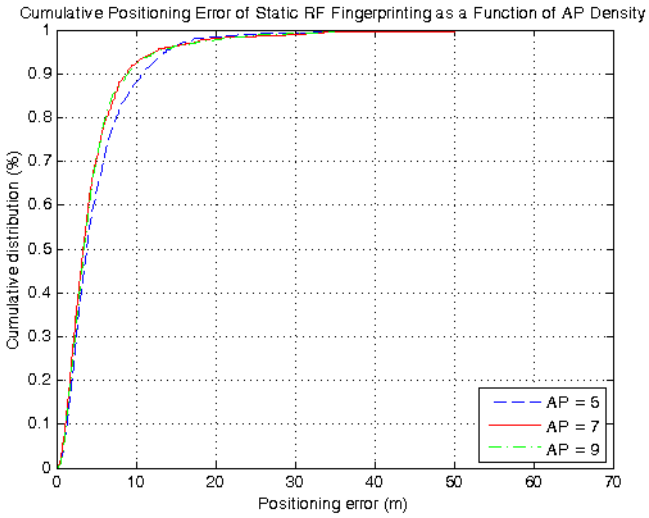


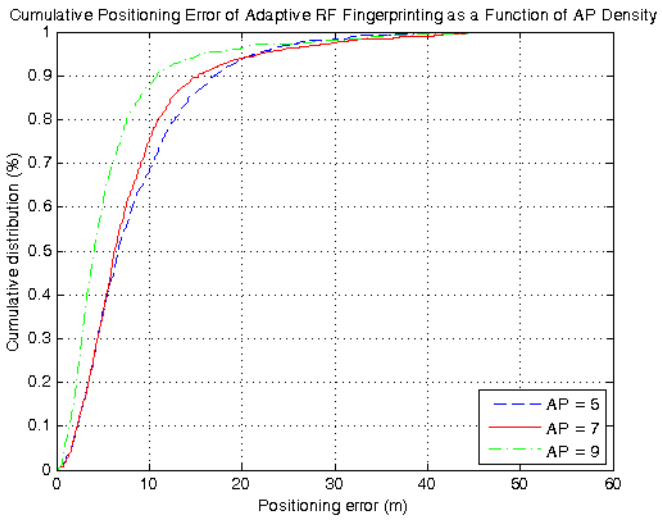
Fig. 3.4: Simulation setup to investigate system performance under changing location beacon topology

Initially, *M = 7* location beacons are deployed in the test site to denote the baseline topology as shown in Fig. 3.4(a). To simulate topology change due to addition and removal of location beacons, *P = 2* beacons are then added to and removed from the

baseline topology, as illustrated in Fig. 3.4(b) and (c), respectively. For all three topologies, *T = 1000* online positioning tests are performed and the results in terms of cumulative positioning errors between static and adaptive RF fingerprint algorithms are plotted in Fig. 3.5. Meanwhile, to simulate topology change due to relocation of location beacons, *P = 2* beacons are repositioned to yield a new location beacon topology as shown in Fig. 3.4(d). Another *T = 1000* online positioning tests are performed and the results are plotted in Fig. 3.6.



(a)

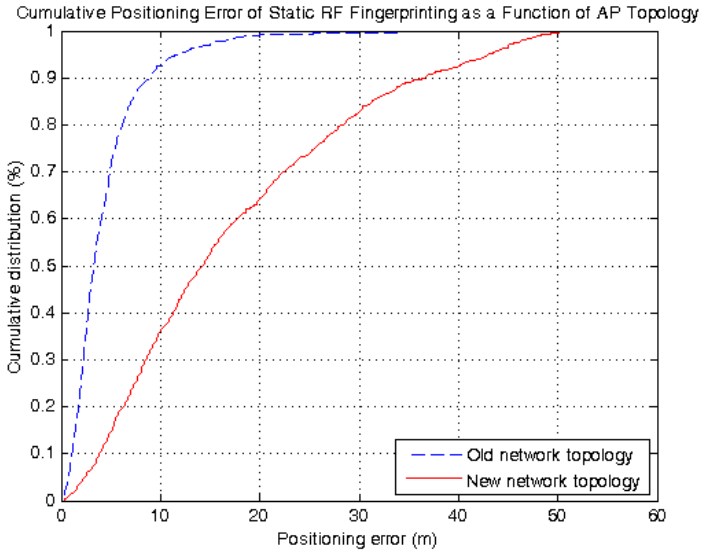


(b)

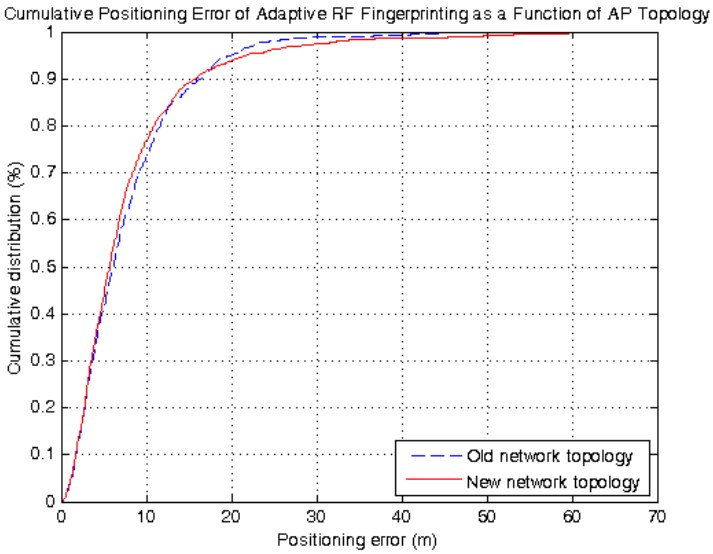
Fig. 3.5: Cumulative positioning error under changing location beacon topology due to addition and removal of location beacons

Fig. 3.5(a) plots the cumulative positioning error of the static RF fingerprint algorithm. Note that the curve corresponding to *M = 7* beacons denotes the baseline performance of the system since the static radio map was calibrated based on this topology. When two additional location beacons are introduced i.e. *M = 9*, the algorithm did not exhibit any noticeable positioning improvement compared to the baseline results. This is because the algorithm is unable to benefit from the extra information provided by the new location beacons since the static radio map does not contain their corresponding RSS components. These extra information are filtered out from the unknown RF fingerprint prior to searching the radio map, hence the positioning performance remains identical to the baseline topology. On the other hand, there is a small drop in positioning performance when two location beacons are removed from the baseline topology i.e. *M = 5*. In this case, the unknown RF fingerprint contains less number of signal strength components compared to the radio map. Hence, the "extra" signal strength components of the radio map have to be filtered out prior to searching for the closest RF fingerprint match, thus "diluting" the radio map's resolution. This means the algorithm has lost some information that would otherwise help to better distinguish between nearby locations. As a result, the accuracy of the system deteriorates.

Fig. 3.5(b) plots the cumulative positioning error of the adaptive RF fingerprint algorithm. From the results, it appears there is slight improvement in system performance with increasing number of location beacons. This is consistent with the analytical model introduced in (Krishnakumar & Krishnan, 2005) that concludes the uncertainty of RF fingerprint reduces if more radio sources are added while maintaining the same area size. In addition, the adaptive RF fingerprint algorithm constructs new radio map based on the signal strength information of the unknown RF fingerprint, thus it always has complete information when searching for fingerprint matches. As the number of location beacon increases, the radio map resolution improves and the algorithm can better distinguish between nearby calibration points to yield more accurate location estimates.



(a)



(b)

Fig. 3.6: Cumulative positioning error under changing network topology due to relocation of location beacons

Fig. 3.6(a) plots the cumulative positioning error of the static RF fingerprint algorithm when two location beacons are relocated to yield a new topology. From the result, there is substantial degradation in positioning performance of the system when the location beacon topology changes, even though the number of location beacons stays the same. This is because the signal-spatial information of

the static radio map no longer correlates with the unknown RF fingerprint. This error is then propagated to the location estimates because the search algorithm is unable to find the best fingerprint match.

On the other hand, positioning performance of the adaptive RF fingerprint algorithm remains consistent even though the location beacon's topology has changed, as shown in Fig. 3.6(b). This is due to the location beacon's self-modelling ability proposed in this work. When the radio map is calibrated based on the newly calibrated path loss model coefficients, it models the most recent RF fingerprint characteristics of the site. This implicitly enables the system to adapt to the changing location beacon topology, hence the algorithm is able to yield consistent location estimates.

3.4.2.3. Positioning Accuracy and Computing Costs Comparison

Fig. 3.7 compares the cumulative positioning errors between static and adaptive RF fingerprint algorithms for the case of stationary wireless channel. In terms of positioning performance, the adaptive RF fingerprint algorithm is less accurate compared to the static algorithm. This is typical of RF fingerprinting that uses simulated radio map compared to empirical ones, as concluded in (Bahl & Padmanabhan, 2000; Kwon et al., 2004). Moreover, in the proposed adaptive RF fingerprint algorithm, each location beacon has only small number of empirical RSS measurements of which to model its real-time wireless channel, hence the calibrated path loss coefficients are less accurate. One way to improve this is by deploying additional location beacons to increase the measurement density as shown by the results in Fig. 3.5(b). However, in actual system implementation using Wi-Fi networks, deploying additional APs in the target area might not be so straightforward due to radio interference. Another alternative is to use site-specific radio propagation models that can better characterize the signal strength distribution of the location beacon by taking into account specific geometries and building materials (Alonso et al., 2009), instead of the path loss model used in this

work that lumps together all site-specific contributions into a single parameter.

Meanwhile, Table 3.1 compares the average processing time between the static and adaptive RF fingerprint algorithms as measured by the Matlab's tic-toc function. The results show that the proposed adaptive RF fingerprint algorithm takes about 130 times longer on average to solve for location as compared to the static algorithm. The reason for the substantial processing time of the proposed algorithm is the need to calibrate a new radio map prior to solving for each location in order to adapt the radio map to the dynamic changes of the wireless channel. On the other hand, static algorithm calibrates its radio map just once during the offline phase and only needs to search the radio map to solve for location; hence, its processing time is much lower.

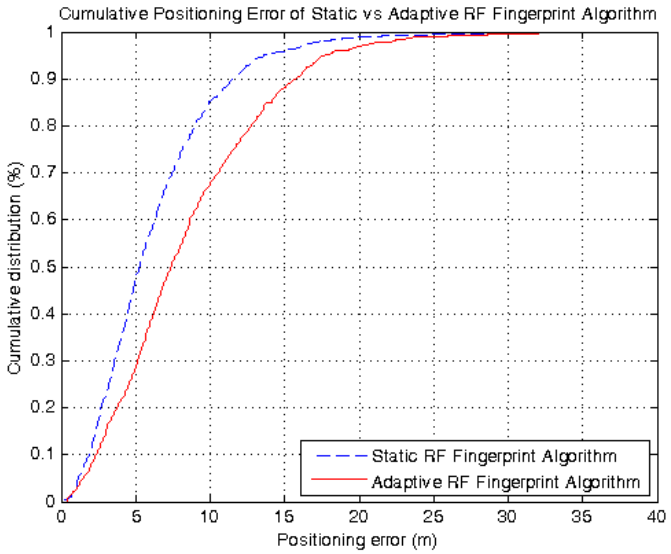


Fig. 3.7: Cumulative positioning error comparison between static and adaptive RF fingerprint algorithm

Table 3.1: Computing time comparison between static and adaptive RF fingerprint algorithm

RF Fingerprint Algorithm	Average Computing Time (ms)
Static	7.48
Adaptive	975.09

3.5. Summary

This chapter presents a novel self-modelling location beacon, capable of calibrating its path loss coefficients in real-time to adapt to the dynamic changes of the radio propagation environment. Client device then uses these coefficients to calibrate an adaptive radio map that is more representative of the site's actual RF fingerprints. A network of these self-modelling location beacons forms the reference infrastructure of the proposed indoor positioning system.

In order to self-calibrate its path loss model's coefficients, each location beacon must first acquire its real-time empirical RSS measurements from several known points. In the proposed system, the neighbouring location beacons provide this information. Here, each location beacon periodically measures the RSS of its neighbouring location beacons that are within radio coverage, and then exchanges these measurements with them via the wireless interface in a collaborative fashion, such that each location beacon ends up with its own empirical RSS as measured by the neighbours. Each location beacon also exchanges its coordinate with the neighbours so they can compute how far the propagation distance is. Equipped with empirical RSS measurements and their corresponding propagation distances, the location beacon applies a regression function on the path loss model to calibrate its coefficients by minimizing the errors in a least-squared sense.

The location beacon broadcasts these path loss coefficients to assist client devices in calibrating the adaptive radio map prior to solving for locations. Firstly, the client listens for broadcast packets from nearby location beacons and measures their RSS to denote the unknown RF fingerprint. Secondly, it decodes and parses the coordinates and path loss coefficients of all detected location beacons and uses this information to compute their RSS for all chosen points in the target area, effectively calibrating the radio map "on-the-fly". Since each location beacon constantly self-models its radio propagation in real-time, the just calibrated radio map models the most recent RF fingerprint characteristics of the site, implicitly

adapting to the dynamicity of the indoor environment. Finally, the client searches the radio map to find the best RF fingerprint match, whose coordinate is returned as the location result.

Positioning performance of the proposed adaptive RF fingerprint algorithm is relatively consistent, even when the site is experiencing dynamic changes. In comparison, the performance of static RF fingerprint algorithm degrades significantly under the same condition. Moreover, the adaptive RF fingerprint algorithm always has complete RSS information prior to solving for locations because the calibrated radio map comprises of the same RSS components as the unknown RF fingerprint, which results in improved positioning performance as more location beacons are deployed. Static RF fingerprint algorithm, however, would not be able to take advantage of any additional signal information not present in the static radio map.

All things being equal, radio map calibrated using path loss radio propagation model generally produces less accurate positioning results compared to radio map calibrated using empirical RSS measurements (Bahl & Padmanabhan, 2000; Kwon, Dundar & Varaiya, 2004). This thesis observes similar results, where the median positioning error of the proposed adaptive RF fingerprint algorithm that uses model-based radio map is about twice as large compared to the static algorithm. Moreover, each location beacon has small number of empirical RSS measurements of which to self-calibrate its path loss coefficients, which results in less accurate path loss model being used to calibrate the radio map only exacerbates the problem. Nevertheless, the proposed algorithm biggest drawback is the long computing time to arrive at the location result because it must first calibrate a new radio map prior to performing an exhaustive search to find the best RF fingerprint match. The computing time increases proportionally with increasing number of calibration points and location beacons, which is unfortunate since the system works best under these conditions.

Chapter 4 : Incremental Radio Map Calibration and RF Fingerprint Search Algorithm

4.1. Introduction

The proposed adaptive RF fingerprint methodology stipulates calibration of a full radio map prior to doing an exhaustive RF fingerprint search as part of the location estimation algorithm. This is necessary to ensure the signal-spatial correlations of the RF fingerprints remain valid during the positioning phase. To construct the radio map, the RSS of all detected location beacons must be computed for each of the specified calibration location. If there are M such beacons and C such calibration locations, it would take $M*C$ computation cycles to construct a full radio map. Next, an exhaustive search of this newly constructed radio map must be performed to find the best RF fingerprint match whose coordinate is returned as the location result. This would incur another C computation cycles.

Both of these tasks are computationally expensive. For server-centric positioning systems, a natural solution is to offload these computationally expensive tasks to the server and wait for the location results to be relayed back to the client device, so the issue is less severe. Unfortunately, the same cannot be said for the proposed client-centric positioning system, where the client device must locally manage both computations. Client devices are typically small, embedded systems with limited computing resources and battery life, so any amount of processing should be kept to a minimum to relieve its computing burden and to conserve battery power.

One way to minimize the computing burden of client devices is to reduce the amount of calibration locations. With less number of RF fingerprints to calibrate and search, the computation cycles can be proportionately reduced. Reducing the number of location beacons can also help minimize the computing burden as well

since for each calibration location, less number of RSS needs to be computed. Unfortunately, both of these solutions would degrade the positioning accuracy of the proposed system.

To minimize computing costs of client device without adversely affecting positioning accuracy of the system, this thesis introduces a novel algorithm that achieves successive reduction in radio map calibration and RF fingerprint search computations. The idea is to compute RSS of one beacon at a time and simultaneously compare the estimate with the measured value. If they are significantly different, then the algorithm immediately knows that this particular calibration point has low probability of being the unknown location, so it can be excluded from subsequent computations. When computing the RSS of the next beacon, the algorithm only needs to compute RSS for the remaining calibration points identified in the previous iteration. This way the number of calibration points will reduce in a systematic manner, thus reducing the computing burden of the sensor. When all beacons have been processed, the average coordinate of the remaining calibration points is returned as the location result. It is also possible that only one point remains before all beacons have been processed, thus reducing the number of RSS values to compute. In this case, the algorithm no longer needs to continue and it can just return this point as the location result.

4.2. Algorithm Description

In this section, a formal description of the proposed incremental radio map calibration and RF fingerprint search algorithm is described.

Supposed a positioning infrastructure comprising of M location beacons $\{b_i; i \in M\}$ serving a two dimensional area R. Assume that the RSS of the ith beacon can be computed using function $\gamma_i(\cdot)$ for any arbitrary point k in R, such that $S_i(k) = \gamma_i(k); i \in M$. Hence, RF fingerprint of point k can be represented by the tuple $\{L(k), F(k)\}$ where $L(k) = (x_k, y_k)$ and $F(k) = \{S_i(k); i \in M\}$. If there are C such

points in R, then a full radio map consists of C-tuples of $\{L, F\}$ pairs. Next, supposed a client device is at unknown location $L(u)$ in R and the unknown RF fingerprint measured at this location is $F'(u) = \{S'_i; i \in M\}$. The objective is to find a reference RF fingerprint $F(j)$ in the radio map where $\min_{L(j)\in C}(F'(u) - F(j))$. Finally, the coordinate of $F(j)$ is returned as the location result i.e. $L(u) = L(j)$.

To construct a full radio map, the client device needs to perform *M*C* computations, while doing an exhaustive search of this map requires another C computations. If one unit of calibration and one unit of search cost the same, then the total computing costs of the client device is *C(M+1)*. To minimize this cost, the number of fingerprint calibration points (C) and/or the number of beacons (M) must be reduced. This thesis proposes a novel algorithm that achieves this reduction in a principled way.

In order to solve for location, the client device first acquires the unknown RF fingerprint by measuring the RSS of nearby location beacons. Assuming the sensor can detect M location beacons at the unknown location $L(u)$, thus the unknown RF fingerprint is $F'(u) = \{S'_i; i \in M\}$. The unknown RF fingerprint *F'(u)* is then sorted in descending order of RSS to yield *F'$_{sort}$(u)*.

Given *F'$_{sort}$(u)*, the algorithm starts by selecting the first location beacon B_1 corresponding to the strongest measured RSS. This also implies the chosen location beacon is closest to the unknown location. Next, the algorithm iterates over all calibration points in R. For each calibration point k, the algorithm first computes the RSS of the chosen location beacon at this location using the estimation function $\gamma(\cdot)$ i.e. $S(k) = \gamma(k)$. Next, the algorithm compares the computed value with the measured RSS i.e. $S(k)_{diff} = S' - S(k)$. If the difference is within the specified thresholds $S_{min} \leq S(k)_{diff} \leq S_{max}$, the algorithm keeps the point k for the next iteration, else it is removed. This is because distant locations compared to *L(u)* would result in vastly different RSS values. After all calibration points have been

evaluated, the algorithm substitutes the remaining points C', where $C' \ll C$ as the new calibration points C for the next iteration.

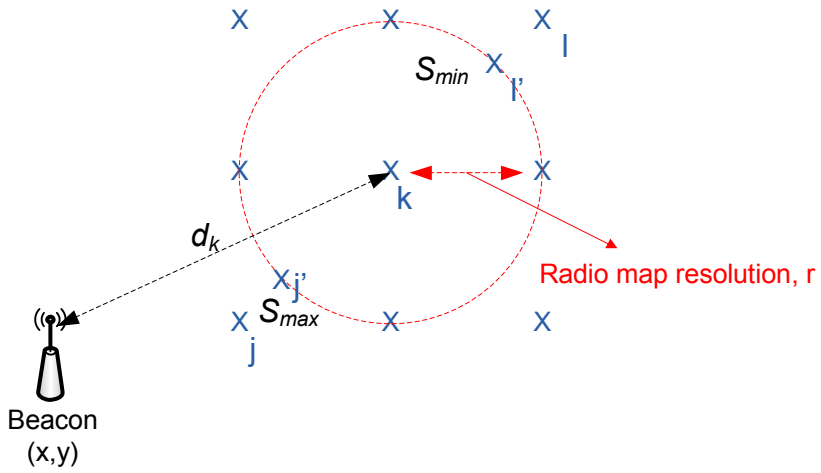


Fig. 4.1: Representative diagram for radio map calibration

The maximum and minimum thresholds for each of the calibration point are basically the RSS values of its closest and farthest neighbours as measured from the location beacon of interest. Consider the setup of Fig. 4.1, where point k is currently being calibrated. Its closest and farthest neighbours, as measured from location beacon B, are points j and l, respectively. Hence, the maximum threshold is computed being the RSS at point j i.e. $S_{max} = S(j) = \gamma(j)$, while the minimum threshold is the RSS at point l i.e. $S_{min} = S(l) = \gamma(l)$. To ease computation, the closest neighour of point k is approximated by subtracting its distance to location beacon B from the radio map resolution. Similarly, the farthest neighbour is approximated by adding the radio map resolution to its distance from location beacon B. The new approximated neighbours are shown as j' and l' in Fig. 4.1.

The algorithm repeats the computing cycle for the next beacon in $F'_{sort}(u)$. Note that the updated calibration points are now much lesser compared to the previous iteration since the algorithm has filtered out those points that have low probability of being the unknown location. If at any iteration cycle there is only one calibration point remaining i.e. $C' = 1$, the algorithm stops at this iteration and returns that point as the location result. Else, the algorithm moves to the next beacon until all

beacons in $F'_{sort}(u)$ have been surveyed. After the final iteration completes, the algorithm computes the average of the remaining points in C', which is then returned as the location result. It is also possible that there is zero calibration point remaining after certain iteration completes i.e. $C' = 0$. In this case, the algorithm averages the remaining calibration points of the previous iteration as the location result. Fig. 4.2 illustrates the processing flow of the proposed algorithm.

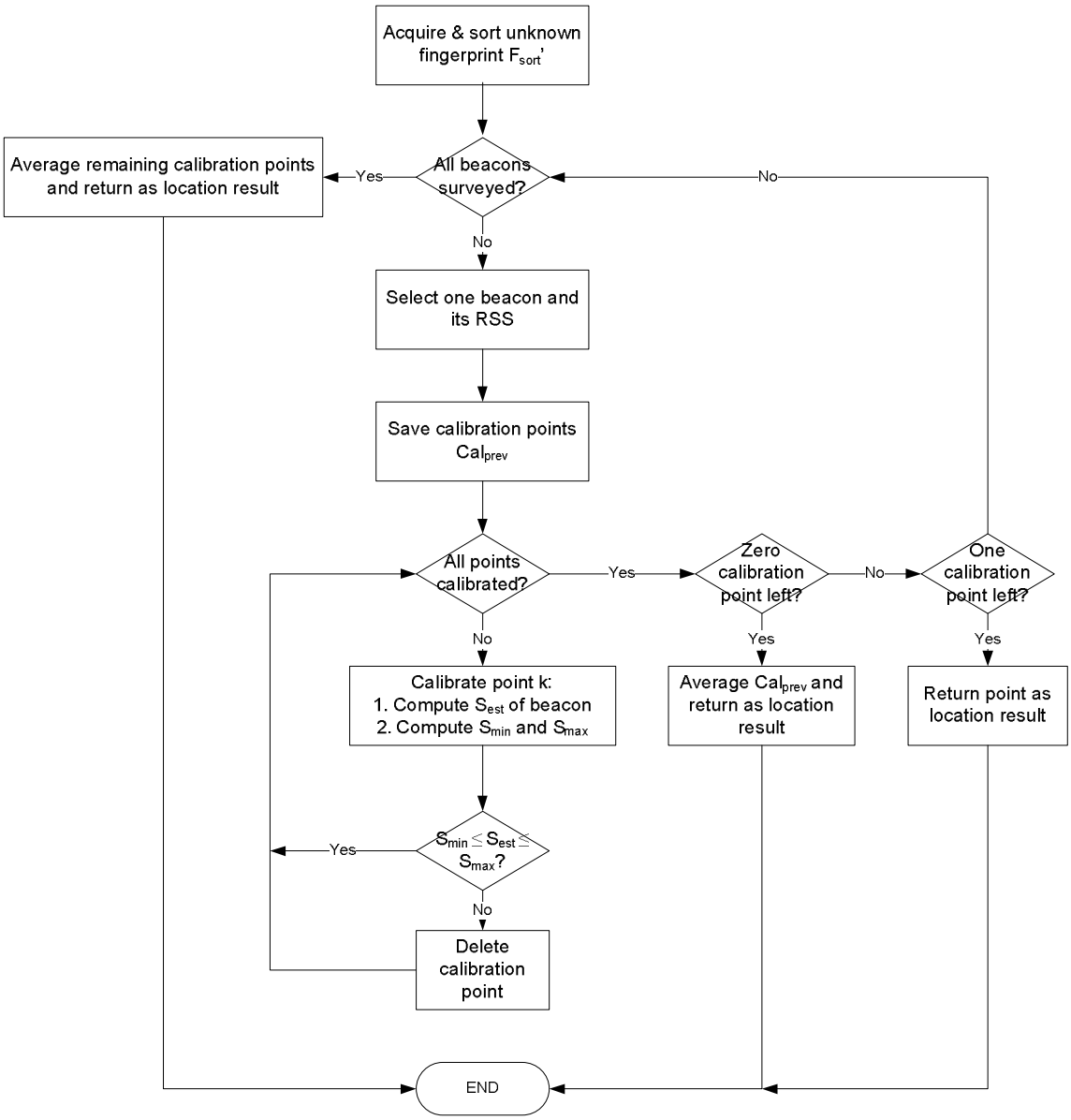


Fig. 4.2: Iterative radio map calibration and location estimation algorithm processing flow

For the first iteration, computation and comparison of the RSS consume C computations each, so the total computation costs of the client device is $2C$. But as the algorithm iterates over the detected beacon's list, the number of calibration points drops i.e. reduction in C dimension, so the client device performs less computation compared to previous iteration. It is also possible that the algorithm stops and returns a location result before all detected location beacons have been surveyed; hence this also reduces the computation costs in the M dimension. However, to arrive at the exact net savings in computing costs is difficult because the number of iteration and calibration points per iteration vary according to the dynamics of the location beacons' radio propagation characteristics.

4.3. Simulation & Discussion

4.3.1. Simulation Methodologies

Evaluation of the positioning accuracy and computing performance of the proposed algorithm is performed via simulation using Matlab version R2013a on an Intel-based computer with 2.8GHz processor and 4GB of RAM running 32-bit Windows 7 operating system.

The same simulation setup described in Section 3.4.1 is adopted, with the exception that path loss coefficients that characterize the RSS distributions of the ith location beacons are assumed ideal i.e. $\alpha_i = 2$ and $\sigma_i = 0$. This is because the objective of this simulation is to compare the performance between two RF fingerprint algorithms in terms of accuracy and computing costs. Hence, error contributions due to varying wireless channels are ignored to simplify analysis. Even if the parameters were assigned to simulate the varying channel conditions, the error contributions would equally affect both algorithms, so the net effect is still zero.

A total of $T = 1000$ simulation runs are performed. In each simulation run, the client device is assigned a random coordinate (u_x,u_y) in R to denote its actual location. Next, the RSS of M location beacons are computed at point (u_x,u_y) to simulate the

unknown RF fingerprint *(F')* using the assigned α_i and σ_i parameters. Finally, the client's location (u'_x,u'_y) is estimated based on its unknown RF fingerprint *(F')* using the proposed incremental algorithm. The positioning errors in terms of Euclidean distance between the actual and estimated locations are computed for all *T* iterations and plotted as a CDF plot. In addition, the computation costs in terms of processing time are measured using Matlab's tic-toc function and its average reported in the result section.

For comparative analysis, the results are compared against the exhaustive RF fingerprint algorithm. In each simulation run, a full radio map is first calibrated by computing the RSS of *M* location beacons for all calibration points in *R*. Next, an exhaustive search is performed on the newly calibrated radio map to find the closest RF fingerprint match to solve for location. The *kNN* algorithm with *k=4* is used as the RF fingerprint search function.

4.3.2. Results & Discussion

4.3.2.1. Positioning Accuracy and Computing Costs Comparison

This simulation scenario compares the positioning accuracy and processing times of the exhaustive and incremental RF fingerprint algorithms. A test site *R* of dimension 50m x 50m is defined and partitioned into *C* points with resolution of *k = 1m* for a total of 2,601 calibration points. *M = 5* location beacons are opportunistically positioned in *R* to achieve a well-balanced network topology as follows: $B_1 = (13,13)$, $B_2 = (37,13)$, $B_3 = (25,25)$, $B_4 = (13,37)$ and $B_5 = (37,37)$. Fig. 4.3 illustrates the simulation setup.

Fig. 4.4 plots the cumulative positioning error while Table 4.1 summarizes the average processing times of both exhaustive and incremental RF fingerprint algorithms. From the CDF plot, both algorithms exhibit similar error curves, indicating their positioning performance is identical. Meanwhile, the average processing time of the incremental RF fingerprint algorithm is about half compared

to the exhaustive algorithm. This indicates that the proposed incremental algorithm is computationally more efficient because it can produce faster location estimate with similar accuracy compared to the exhaustive algorithm.

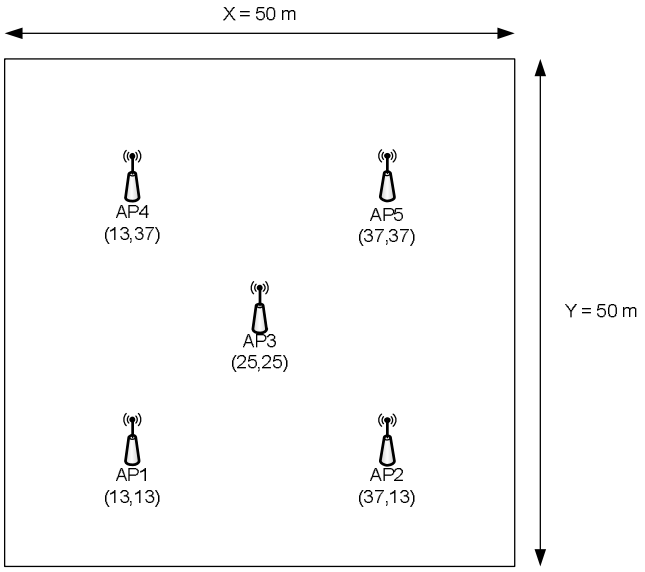


Fig. 4.3: Simulation setup to compare positioning accuracy and computing times of exhaustive and incremental RF fingerprint algorithms

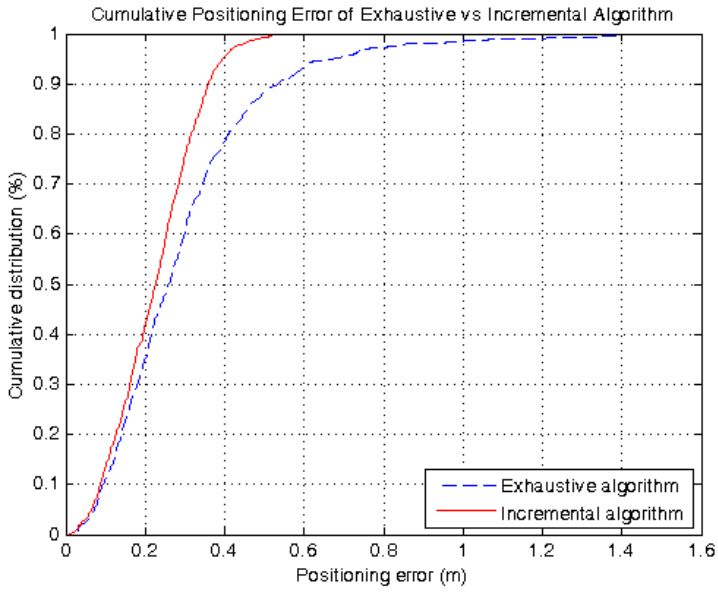


Fig. 4.4: Cumulative positioning error comparison between exhaustive and incremental RF fingerprint algorithms

Table 4.1: Comparison of average processing times between exhaustive and incremental RF fingerprint algorithms

RF Fingerprint Algorithm	Average Computing Time (ms)
Exhaustive	966.70
Incremental	393.72

4.3.2.2. Positioning Performance and Computing Cost Comparison as a Function of Location Beacon Density

This simulation compares the positioning performance and computing costs of exhaustive and incremental RF fingerprint algorithms as a function of beacon density. A test site R of dimension 50m x 50m is defined and partitioned into C points with resolution of $k = 1m$ for a total of 2,601 calibration points. Meanwhile, the number of location beacons is varied between 3 and 9 i.e. $M = [3,5,7,9]$ and are opportunistically positioned in R as follows:

1. For M = 3: $B_1 = (13,13)$, $B_2 = (37,13)$, $B_3 = (25,37)$
2. For M = 5: $B_1 = (13,13)$, $B_2 = (37,13)$, $B_3 = (25,25)$, $B_4 = (13,37)$, $B_5 = (37,37)$
3. For M = 7: $B_1 = (13,13)$, $B_2 = (25,13)$, $B_3 = (37,13)$, $B_4 = (25,25)$, $B_5 = (13,37)$, $B_6 = (25,37)$, $B_7 = (37,37)$
4. For M = 9: $B_1 = (13,13)$, $B_2 = (25,13)$, $B_3 = (37,13)$, $B_4 = (13,25)$, $B_5 = (25,25)$, $B_6 = (37,25)$, $B_7 = (13,37)$, $B_8 = (25,37)$, $B_9 = (37,37)$

Fig. 4.5 illustrates the simulation setup.

Fig. 4.6 and Fig. 4.7 plot the cumulative positioning errors of the exhaustive and incremental RF fingerprint algorithms as a function of location beacon density, respectively. For both algorithms, the results show that their positioning errors reduce with increasing number of location beacons. This is because both algorithms can better distinguish between nearby calibration points since each RF fingerprint is characterized by more signal components as the number of location beacon increases. Note that positioning improvement for incremental algorithm has saturated somewhat at higher location beacon density. This is possibly because

the algorithm has found the most probable location without the need to iterate over all detected location beacons. Hence, additional information provided by these extra location beacons is ignored. From the result, it appears that the positioning accuracy of the incremental algorithm saturates at 5-beacon density for the chosen simulation setup, thus system deployment with location beacon count above this value may not yield proportional increase in performance.

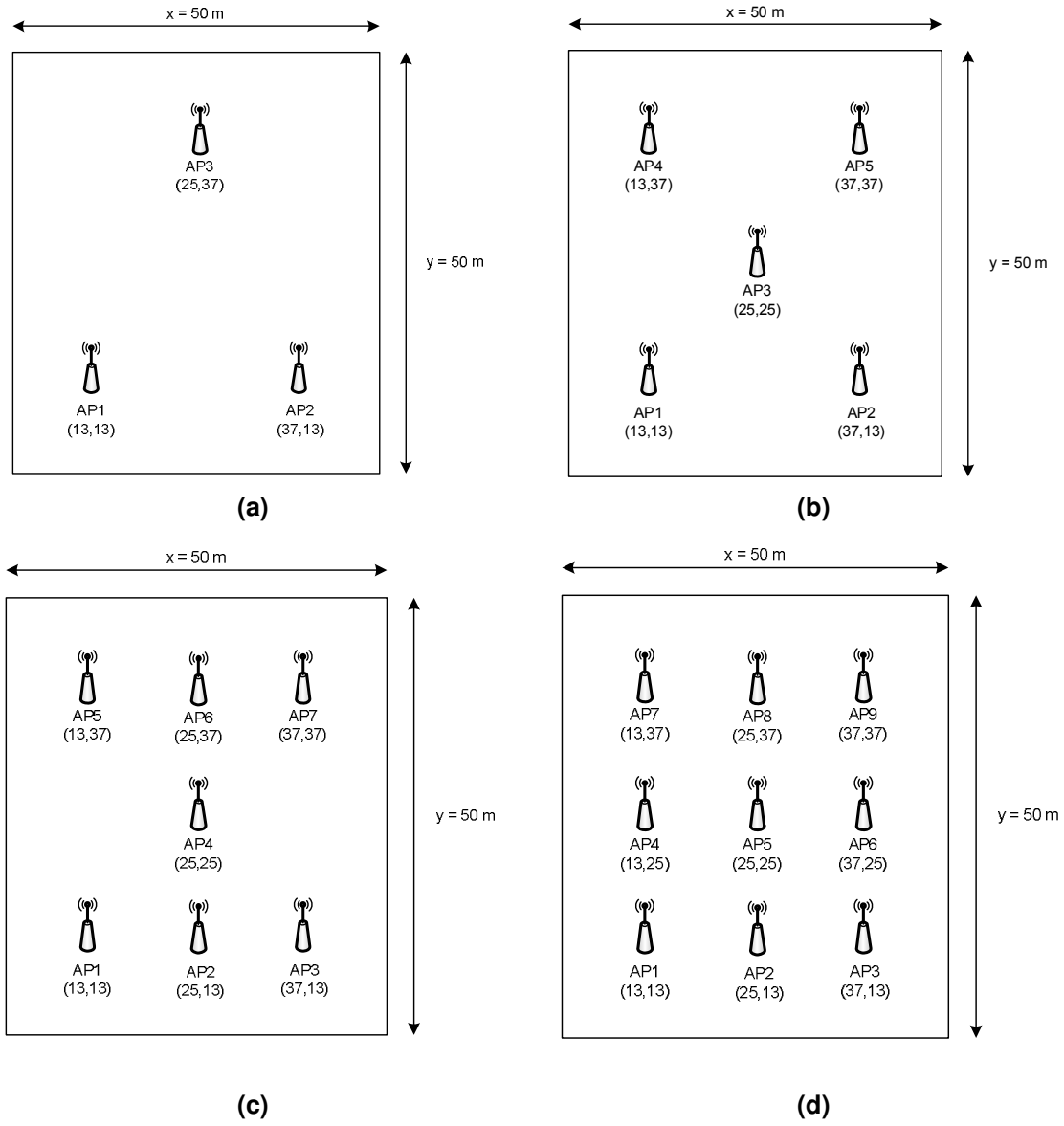


Fig. 4.5: Simulation setup to compare positioning accuracy and computing times of exhaustive and incremental RF fingerprint algorithms as a function of beacon density

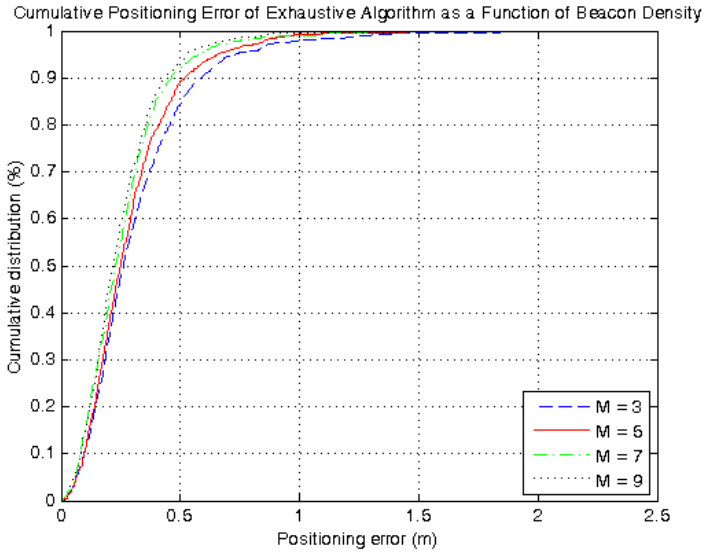


Fig. 4.6: Cumulative positioning error of exhaustive RF fingerprint algorithms as a function of location beacon density

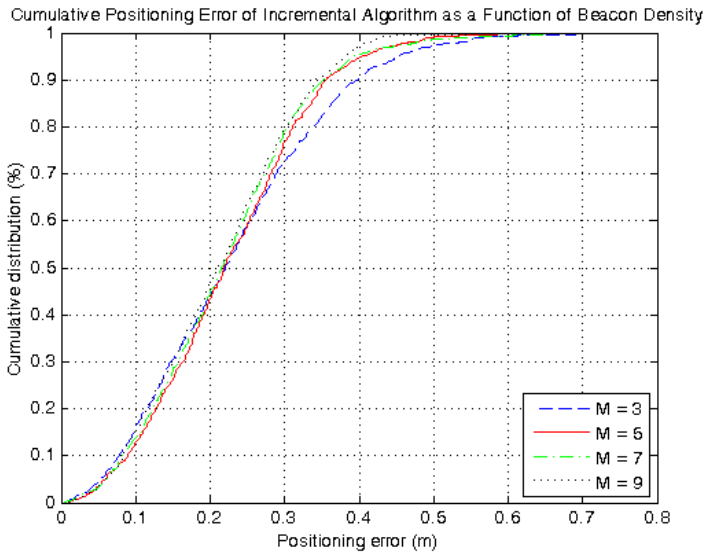


Fig. 4.7: Cumulative positioning error of incremental RF fingerprint algorithm as a function of location beacon density

Table 4.2 summarizes the average processing time of both algorithms. For exhaustive algorithm, its average processing time increases with increasing number of location beacons. This is expected because the algorithm must compute RSS values of all detected location beacons to calibrate the full radio map, thus with higher number of location beacons, the algorithm requires more processing time. On the other hand, the average processing time of incremental algorithm is almost

constant even when the number of location beacon increases. This is because for all location beacon density simulations, the algorithm only needs to iterate over all calibration points only once during the first iteration, which constitutes the bulk of its computing costs. For subsequent iterations, the number of calibration points has dropped significantly, thus their contributions to the overall processing time are relatively negligible. Hence, the differences in computation time between different location beacon densities are not so significant.

Table 4.2: Average processing times of exhaustive and incremental RF fingerprint algorithms as a function of location beacon density

Location beacon count, M	Average Computing Time (ms)	
	Exhaustive Algorithm	Incremental Algorithm
3	574.79	391.99
5	939.56	384.97
7	1385.80	403.62
9	1687.46	383.06

4.3.2.3. Positioning Performance and Computing Costs Comparison as a Function of Radio Map Resolution

This simulation compares the positioning performance and computing costs of exhaustive and incremental RF fingerprint algorithms as a function of radio map resolution. A test site R of dimension 50m x 50m is defined and partitioned into C points with varying radio map resolution between 1m and 10m i.e. $k = [1m, 2m, 5m, 10m]$. Note that as k increases, the radio map resolution decreases due to lesser number of calibration points. Based on the chosen radio map resolution, the number of calibration points is as follows:

1. For $k = 1m$, $C = 2{,}601$ points
2. For $k = 2m$, $C = 676$ points
3. For $k = 5m$, $C = 121$ points
4. For $k = 10m$, $C = 36$ points

Meanwhile, $M = 5$ location beacons are opportunistically positioned in R to achieve a well-balanced network topology as illustrated in Fig. 4.3 above.

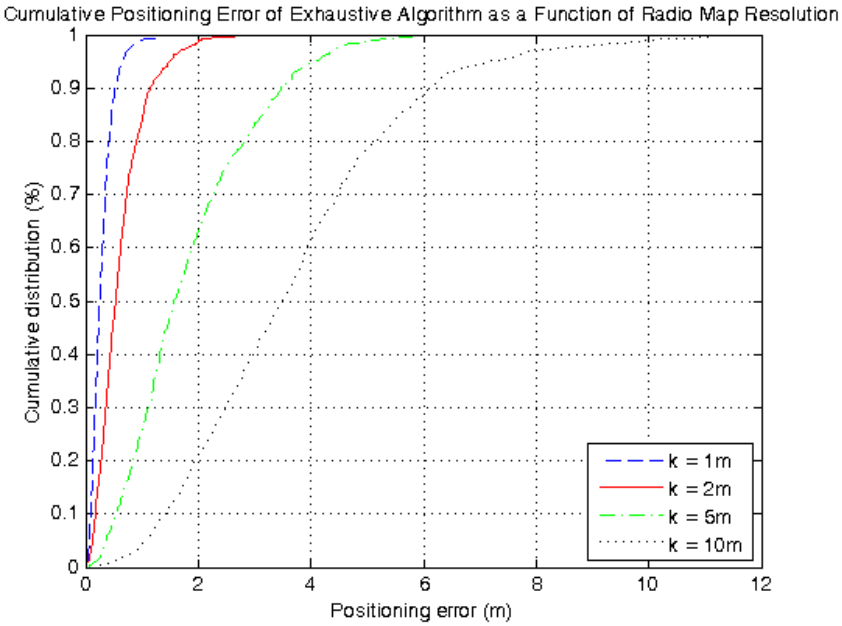


Fig. 4.8: Cumulative positioning error of exhaustive RF fingerprint algorithm as a function of radio map resolution

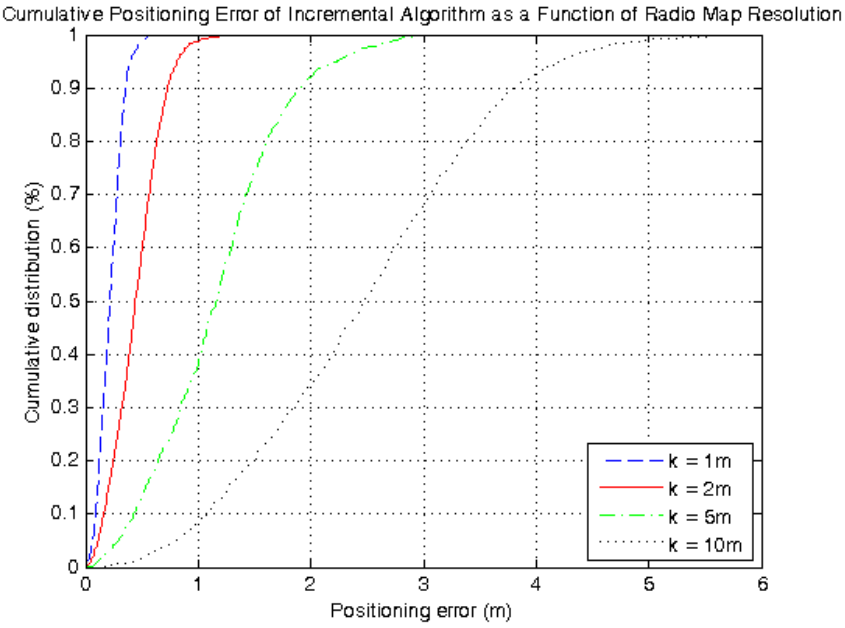


Fig. 4.9: Cumulative positioning error of incremental RF fingerprint algorithm as a function of radio map resolution

Table 4.3: Average processing time of exhaustive and incremental RF fingerprint algorithms as a function of radio map resolution

Radio Map Resolution, k	Average Computing Time (ms)	
	Exhaustive Algorithm	Incremental Algorithm
1m	940.16	384.18
2m	246.05	105.51
5m	43.89	22.59
10m	13.15	8.94

Fig. 4.8 and Fig. 4.9 plot the cumulative positioning error of the exhaustive and incremental RF fingerprint algorithms as a function of radio map resolution, respectively. As expected, the positioning performance of both algorithms degrades with lower radio map resolution. This is true for RF fingerprinting in general because the methodology treats the positioning problem as a classification problem. With lower resolution, the number of calibration points are limited and sparsely distributed within the test site. Thus, lesser numbers of RF fingerprints are available for the algorithm to make the best classification choice. As the number of calibration points increases, more RF fingerprints are available and the algorithm's positioning performance improves accordingly since it has access to more and better choice. However, this comes at the expense of increasing processing time since the algorithm must calibrate and search many more RF fingerprints, although the incremental algorithm can compute about twice as fast compared to the exhaustive algorithm for the same radio map resolution, as shown in Table 4.4. Again, this is due to lesser number of overall points the algorithm has to calibrate and search since most points have been filtered out during the first iteration.

4.4. Summary

A novel positioning algorithm suitable for implementation in client-centric adaptive RF fingerprint positioning system is proposed. Instead of calibrating the

full radio map up-front and then searching for the best RF fingerprint match in an exhaustive manner, the proposed algorithm performs these tasks incrementally. Given the RF fingerprint measured at the unknown location, the algorithm first sorts the RSS in descending order. The algorithm then iterates over all calibration points, where it computes RSS of the strongest beacon and simultaneously compares the estimate with actual measured value. If the computed RSS falls outside the specified threshold values, that calibration point is filtered out since this is likely to be distant points relative to the unknown location. In subsequent iterations, the algorithm computes and compares the RSS of the next strongest beacon but only for the remaining calibration points from the previous iteration. As a result, the overall amounts of calibration points the algorithm needs to compute the RSS are significantly reduced, thus minimizing its computing costs. Once all beacons in the unknown RF fingerprint have been surveyed, the algorithm averages the remaining calibration points and returns the result as the location estimate. From simulation results, the proposed incremental algorithm yields similar positioning accuracy compared to the exhaustive algorithm, but this level of performance is achieved twice as fast using the same computing resources.

Chapter 5 : Development of an Indoor Positioning System Prototype

5.1. Introduction

This chapter discusses the development of the proposed indoor positioning system prototype to investigate its performance and demonstrate its applicability in real-world deployment. The system prototype uses commodity Wi-Fi hardware to implement its self-modelling location beacon and client sensor components, which lowers development costs since it does not require any purpose-built hardware. This also means the proposed system can readily integrate on top of existing Wi-Fi networks with just software modifications to the AP and MS, allowing the networks to offer both networking and positioning services using the same infrastructure.

The main feature of Wi-Fi technology exploited in this work is its network discovery mechanism. In Wi-Fi, AP periodically broadcasts beacon frames to announce its presence and advertise its network capabilities. An MS passively listens for these beacons in each Wi-Fi channel, and buffers them to measure their RSS and to extract network information, such as their MAC addresses, prior to establishing connection. Alternatively, MS can actively solicit response from the AP rather than wait for it to announce itself. In this case, the MS moves into each channel and after gaining access to the wireless medium, transmits a probe request frame, either to specific or all nearby APs. The corresponding APs shall reply with a probe response frame. The content of probe response frame is largely similar to beacon frame, except beacon frame has additional field that conveys information about buffered data packets addressed to the MS while it sleeps (Gast, 2002).

Every beacon and probe response frame contains blobs of data of variable lengths, called Information Element (IE). Each one consists of one-byte identifier field, one-byte data length field and up to 255 bytes of data. Moreover, the Wi-Fi

standard defines a vendor-specific IE identified by element identifier 221 (Gast, 2002). Its purpose is to allow wireless devices to carry non-standard information within a predefined format to achieve interoperability between devices. Typical usage is to embed special signalling message that describes extended hardware options to enhance operation between devices made by the same vendor. The first three bytes of vendor-specific IE carry the Organization Unique Identifier (OUI) that identifies the AP's manufacturer. This leaves only 252 bytes of custom data remaining, although each beacon and probe response frame can carry multiple vendor-specific IEs, as long as the total length is within the maximum frame body length specified by the standard.

This work exploits the flexibility offered by vendor-specific IE to embed positioning information into beacon and probe response frames while relying on Wi-Fi's standard network discovery mechanism to transmit them to client devices. There are three different types of positioning data embedded into the vendor-specific IE. In order to distinguish between them, the first available byte after the OUI is reserved to carry control information, as follows:

a. If the control byte equals 0x00, this vendor-specific IE carries location coordinate of the AP
b. If the control byte equals 0x01, this vendor-specific IE carries the MAC address and RSS information of a single neighbouring AP
c. If the control byte equals 0x11, this vendor-specific IE carries the MAC address and path loss model coefficients of the AP
d. Other values of the control byte are reserved

Fig. 5.1 summarizes the mapping of proprietary protocol of the proposed positioning system on to Wi-Fi's vendor-specific IE of beacon and probe response frames.

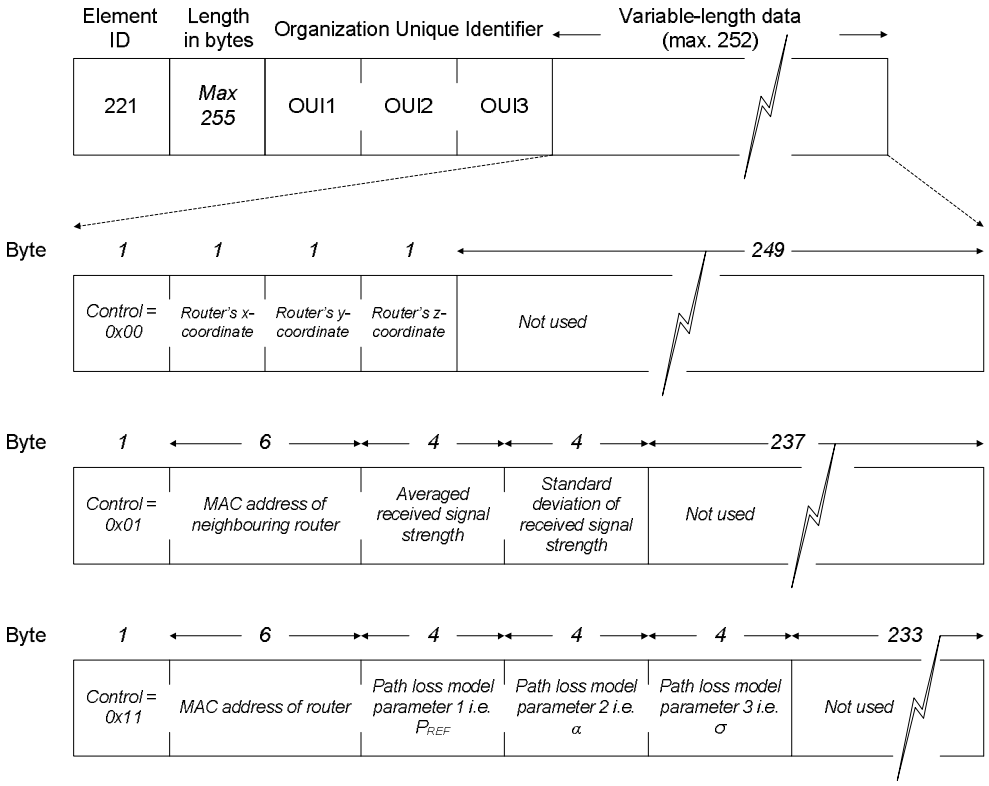


Fig. 5.1: Format of vendor-specific IE and mapping of the proprietary positioning protocol of this work

To realize the proposed location beacon on commodity Wi-Fi AP, it must be able to perform network discovery by scanning for nearby APs, extracting their MAC addresses and measuring their RSS. Furthermore, the AP must be able to parse vendor-specific IE of beacon or probe response frames received from nearby APs to acquire their coordinates, as well as its own RSS as measured by these APs in order to self-calibrate its path loss coefficients. Finally, the AP must be able to embed all these data, together with its coordinate into vendor-specific IE of beacon and probe response frames prior to broadcast. Unfortunately, standard Wi-Fi AP only operates in the so-called Master mode where network discovery is not possible. Moreover, embedding and parsing the vendor-specific IE of beacon and probe response frames to add and retrieve data, as well as self-calibrating the path loss coefficients require modifications to the AP's default factory firmware. This is

difficult, if not impossible, to do due to lack of manufacturer-provided software tools to reprogram the hardware.

To address these challenges, this work utilizes a Linux-based open-source wireless networking firmware called OpenWRT to replace the AP's default factory firmware. OpenWRT uses wireless stack that allows configuration of the wireless interface from Master to Station mode via software, which is vital in order to enable the AP to perform network discovery (Dujovne et al., 2010). Moreover, the OpenWRT distribution comes with software development toolchain that enables the development, cross-compilation and installation of new software packages for customizing the functionality of the hardware device. OpenWRT currently supports various commodity Wi-Fi APs ("OpenWRT Table of Hardware", n.d.). In this work, the AP of choice is WRT54GL wireless router manufactured by Linksys. This router complies with Wi-Fi's b and g standards, has 16MB of RAM and 4MB of flash storage, and is equipped with wireless chipset made by Broadcom (Asadoorian & Pesce, 2007). With OpenWRT configured and installed on the chosen WRT54GL router, this thesis develops novel software that implements the proposed self-modelling location beacon, which is one of its contributions.

The client device that implements the location sensor functionality is realized using a Linux-based Wi-Fi enabled laptop. An open-source wireless configuration tool called *iw* that comes pre-installed in the OS is used for scanning the wireless networks to obtain the unknown RF fingerprints and to parse the vendor-specific IEs of beacon and probe response frames to obtain the positioning data broadcasted by the location beacons. This information is logged into text files and later post-processed using the proposed incremental RF fingerprint algorithm in Matlab to compute the location estimates.

In the next section, an overview of wireless networking support in Linux is presented. Section 5.3 describes the implementation details of the prototype, focusing on software development, which is the core component that enables

positioning functionality of the proposed system. The system prototype is then deployed in real-world settings to evaluate its performance, and the results are presented and discussed in Section 5.4. Finally, Section 5.5 concludes this chapter.

5.2. Wireless Networking Support in Linux

Early wireless networking devices implement the MAC layer functionality, in addition to other low level processing, in hardware. These "Full MAC" devices, as they are known, present themselves to the host processor as another local area network device, most commonly resembling an Ethernet device (Linville, 2008). This generally involves accepting Ethernet frames from the host, converting them to wireless frames for transmission, and doing the reverse for reception.

These "Full MAC" devices typically utilize an on-board controller running firmware that handles all these transmit and receive frame management tasks, in addition to controlling access to the wireless medium. A drawback of this design is that dedicated controller and memory for the firmware add to the overall costs of the hardware. Thus, newer wireless devices, especially for consumer applications, minimize or eliminate these components by moving some or all of the wireless networking functionalities to the host processor. This new architecture is known as "Soft MAC" because it implements the MAC layer functionality in software. Such design allows finer control of the hardware and provides easier upgrade path to support new MAC protocols or services.

The wireless stack that implements MAC layer processing in software is known as *mac80211*. It handles two main tasks, which are transmit/receive and MAC layer management entity (MLME). In the transmit path, the packets are handed to the virtual interface's transmit function (*ieee80211_subif_start_xmit*), converted into 802.11 format, and multiplexed into master interface (*ieee80211_master_start_xmit*) (Vipin & Srikanth, 2010). After this the transmit handlers are invoked, which select the key and transmission rate, insert sequence

number based on hardware capability, select encryption algorithm, fragment the frames, calculate transmission time and generate control information (Vipin & Srikanth, 2010). In the receive path, *mac80211* first checks the type of packet and receive status, and then prepares the receive handler that verifies the packet alignment for proper processing, followed by decryption and defragmentation (Vipin & Srikanth, 2010). Data packets are converted into 802.3 formats and delivered to the networking stack, while management packets are delivered to the MLME (Tabassam, Trsek, Heiss, & Jasperneite, 2009). In MLME, user requests, such as probe request/response, authentication request/response or association request/response, are first translated into internal variables, followed by running the appropriate state machine. Once finished, it sends notification to the users.

For greater flexibility, the *mac80211* stack is partitioned into two parts. All the core MAC features that are unlikely to change are maintained inside the kernel, while other functionalities such as management, control, etc. are realized as user space applications. By doing so, new features or services can be easily supported since it is much easier to replace application packages than to recompile the kernel.

mac80211 is a software framework with whom device drivers interact to realize the wireless networking functionality. To take advantage of these facilities, wireless device drivers must be rewritten. Fortunately, *mac80211* provides well-defined interfaces for this purpose, which greatly simplifies driver development. To date, wireless devices from various vendors are already *mac80211*-compliant (Linville, 2008). By providing a single standardized wireless stack, not only different wireless devices can be easily supported, managing this single code base takes much less work as well. Additionally, user space programs that are written for any *mac80211*-compliant wireless device can be easily ported to other *mac80211*-compliant wireless devices.

Traditionally, wireless devices are configured in Linux using Application Programming Interface (API) known as Wireless Extensions (*wext*) (Linville, 2008).

This API is based on a series of Linux *ioctl* calls that allow the device driver kernel modules to expose configuration parameters and network attributes to the user space programs. Typical usage examples include accessing the aggregate data statistics, setting of specific driver-level parameters and listing the results of APs in range. *wext* has worked sufficiently well for configuring early wireless devices that were prevalent when it was first introduced, and it remains at least minimally serviceable for newer wireless interface designs (Linville, 2008).

Unfortunately, *wext* has several shortcomings that make it difficult to extend and maintain, especially as wireless networking technology continues to evolve. The main problem is the lack of details on various network attributes, such as the default behaviours, operational timing and order of configuration steps (Linville, 2008). For some parameters, it does not even provide the exact meaning of what they are intended to be (Linville, 2008). Furthermore, reliance on individual configuration actions for what otherwise might be considered atomic operations introduces the possibility of race conditions when configuring the devices (Linville, 2008). In addition, *wext* have proven to be difficult to extend without breaking the consistency of user space Application Binary Interface (ABI) between kernel releases (Linville, 2008). Finally, *wext* implementation in the kernel is mostly transparent, which forces individual drivers to re-implement a number of features that might otherwise be shared (Linville, 2008). *wext* has been deprecated but it is still available in the kernel since there are older applications that still rely on it.

To address the drawbacks of *wext*, another mechanism for configuring and accessing wireless devices from user space is developed, known as *cfg80211*. The operations handled by *cfg80211* include device registration, regulatory enforcement, station management, key management, mesh management, virtual interface management and network scanning (Vipin & Srikanth, 2010). The main improvement offered by *cfg80211* over *wext* is it provides interfaces that group logical configuration parameters together so that logically atomic operations are actually handled atomically in the kernel. In addition, *cfg80211* utilizes Netlink

messaging infrastructure as the user-to-kernel inter process communication mechanism, which is implemented as *nl80211* kernel module. Unlike *ioctl* calls, Netlink offers greater flexibility, allowing new features or commands to be easily added without breaking backward compatibility. This allows *cfg80211* to be easily extensible and maintainable. Netlink also provides event-based notifications and allows large data transfers in an efficient way (Ayuso, Gasca, & Lefevre, 2010).

Fig. 5.2 illustrates the architecture of wireless networking subsystem in Linux.

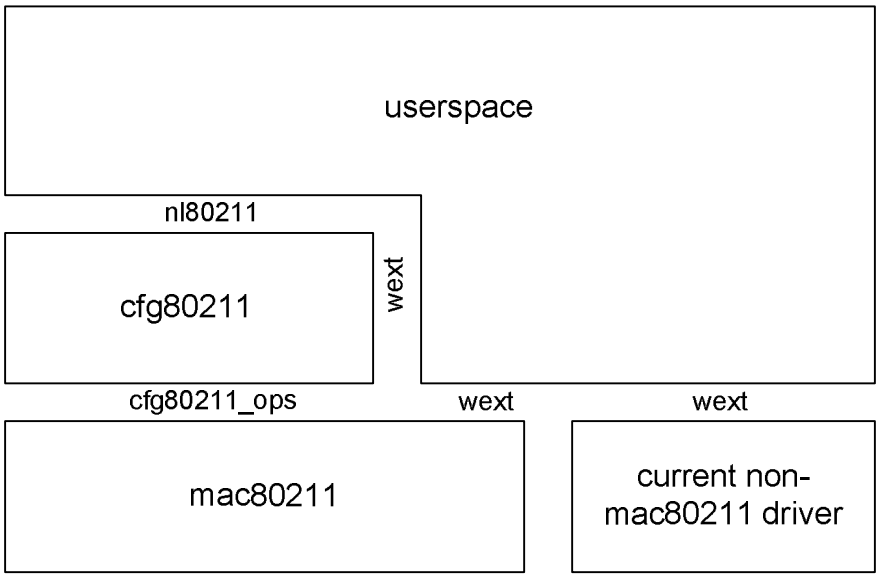


Fig. 5.2: Linux wireless networking stack

Most Linux distribution comes with configuration utility for setting up the wireless interface, for example scanning for nearby wireless networks and connecting to the chosen AP. An example is *iw*, which is a command line based user space program that interfaces to *nl80211* and supports all new *mac80211*-compliant wireless drivers added to the recent Linux kernel. Being a command line tool, all wireless configuration tasks are performed by entering specific commands. Some tasks, such as connecting to a wireless network, require certain sequence of multiple commands to be executed.

5.3. Prototype Design & Implementation Details

5.3.1. OpenWRT Configuration, Build & Installation

There are two ways to obtain the OpenWRT firmware. One is to download and install pre-built image targeted for WRT54GL available from OpenWRT's website. Another is to build the image from source. Building from source has its advantages. Typical embedded platforms have limited storage space. By building from source, developers are able to fully customize the device and only support necessary features for each type of application. Building from source requires proper setup of the build environment.

Due to various customization options offered by OpenWRT, the minimal configuration related to implementing the AP function on WRT54GL is described. Linux implements the IEEE802.11 MAC layer functionality in software, as a kernel module called *mac80211*. For Broadcom wireless chipset, the *b43* driver is *mac80211*-compliant. Another kernel module called *cfg80211* implements the Application Programming Interface (API) used by user space applications to configure the wireless device to perform various functions such as registration, regulatory enforcement, station management and network scanning. Together, *mac80211* and *cfg80211* allow user space applications control of wireless device.

Additionally, Linux implements Netlink-based sockets called *nl80211* as communication channels between user and kernel space to configure the wireless system. OpenWRT provides a lightweight version of Netlink library suitable for resource limited embedded platforms, called *libnl-tiny*. This library provides useful utilities for using Netlink sockets.

In *mac80211*-based devices, the MAC Layer Management Entity (MLME) function of the AP is handled in user space via a software daemon called *hostapd*. The MLME is responsible for authenticating clients, setting encryption and decryption keys and

other aspect of wireless infrastructure. Fig. 5.3 shows a simplified view of Wi-Fi's AP configuration in OpenWRT.

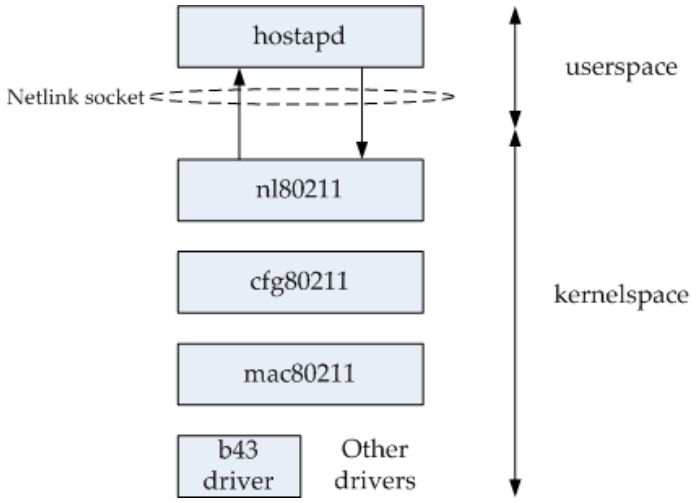


Fig. 5.3: Linksys WRT54GL's AP configuration in OpenWRT

Other than these wireless specific configurations, other hardware settings related to WRT54GL need to be included for the firmware build. In the main configuration menu, choosing Target System as Broadcom BCM947xx/953xx and Target Profile as Broadcom BCM43xx WiFi sets the default hardware settings. Additionally, the main configuration menu also provides option to generate the Software Development Kit (SDK) used to cross-compile new software packages.

Installing the firmware on WRT54GL can be done in two ways. One is to use the web interface's upgrade utility of the default factory firmware. Note that this can only be done once when upgrading from the default firmware. Another way is to install the firmware via TFTP. During boot up, the device's bootloader first validates the actual firmware and then loads it. If this process is interrupted, the bootloader then waits for new firmware to be uploaded over TFTP. OpenWRT provides instructions based on operating system of the host to perform the installation via TFTP.

5.3.2. Location Beacon Implementation

The OpenWRT firmware allows device customization by adding new functionality through software. This thesis develops a user space daemon to implement the location beacon functionality on WRT54GL router. It is developed using C programming language, cross-compiled using OpenWRT's SDK toolchain, and installed on the router using the provided package management utility.

Upon first start up, the daemon initializes the router's network settings and configures the basic header information of its Beacon frame. Next, and during subsequent wake-ups, the daemon initiates network discovery process in order to acquire the Beacon frames of nearby wireless routers and measure their RSS. By default, *hostapd* configures the wireless interface in Master mode. Unfortunately, in this mode, the interface is not able to perform network discovery because such function is undefined. As a workaround, the daemon temporarily switches the wireless interface's operating mode from Master to Managed by killing the *hostapd* process. During this period, the AP functionality ceases to exist so wireless network connectivity is down. While in Managed mode, the daemon configures *cfg80211* via *nl80211* to trigger network scan. The wireless interface's *b43* driver then passively listens for Beacon frames broadcasted by nearby networks. Once the scan is completed, the *b43* driver forwards the received Beacon frames to the *mac80211* stack. The daemon then requests and saves these data through another call to *cfg80211* via *nl80211*. This network scan process is repeated for *ScanCount* times, which is a system level option specified by user. Once completed, the daemon restarts the *hostapd* process to resume its AP functionality.

The daemon's next task is to calibrate the router's path loss coefficients. In order to accomplish this, the router's RSS as measured by the neighbouring routers and their location coordinates are required. This information is embedded into vendor specific IEs of the respective router's Beacon frames, as specified by control byte 0x01 and 0x00, respectively. Note that each Beacon frame received from a

particular router may contain multiple vendor specific IEs with control byte 0x01 since that router may detect and measure RSS of several of its neighbouring routers. Thus, for each vendor specific IE with control byte 0x01, the daemon first compares the MAC address information contained in that IE. If it matches the router's own MAC address, the corresponding mean and standard deviation RSS values are saved by the daemon for subsequent processing; else, that vendor specific IE is ignored. Having obtained the router's empirical RSS and locations where they are measured, the daemon then models the router's path loss characteristics using linear regression, as described in Section 3.2.

During the network discovery process, the wireless *b43* driver also measures the signal strengths of all received Beacon frames via its power measurement circuitries. These measurements are then passed to *mac80211* stack. When the daemon requests the Beacon frames from *cfg80211* via *nl80211* earlier, the RSS measurements are also provided. If the network scan is performed more than once, there could be several RSS returned for each detected router. The daemon next task is to compute the mean and standard deviation of these signal strength measurements.

In the final step, the daemon embeds all these information into multiple vendor specific IEs of the router's Beacon frame. First, it programs the router's location by setting the control byte of the IE to 0x00 and then writing the coordinates into the 3 subsequent bytes. Next, the daemon iteratively programs the MAC address of each detected neighbouring router and its computed RSS mean and standard deviation values into individual vendor specific IE and setting the control byte to 0x01. Finally, the daemon programs the router's MAC address and its calibrated path loss model parameters into another vendor specific IE and sets the control byte to 0x11. Once all the IEs have been programmed, the daemon configures the router's Beacon frame to include these IEs and schedules its transmission as required. This is accomplished via another call to *cfg80211* via *nl80211*. The daemon then goes to

sleep for a specified period. When it wakes up, the whole cycle repeats.

Fig. 5.4 illustrates the overall processing flow of the software daemon.

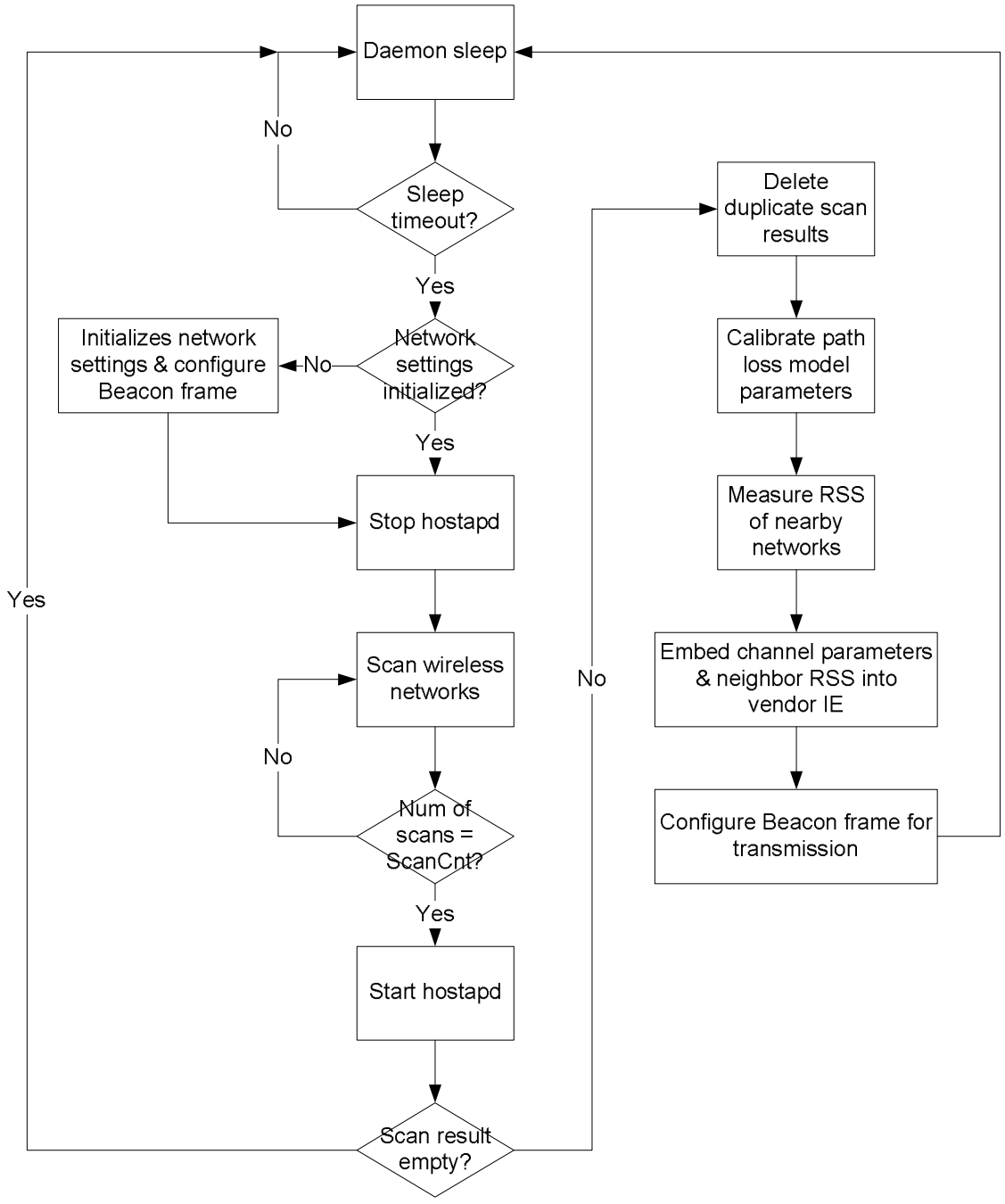


Fig. 5.4: Overall process flow of software daemon implementing the self-modelling location beacon functionality

5.3.3. Location Sensor Implementation

In its current form, the location sensor is realized using a Wi-Fi enabled laptop running open-source wireless configuration tool. The chosen laptop is equipped with AMD's 64-bit 2.4GHz dual core processor with 4GB of RAM and Broadcom's wireless interface card, running Linux Ubuntu version 12.04 OS. The wireless configuration tool, called *iw*, comes pre-installed with the OS and is used to scan for nearby location beacons to obtain their RSS that forms the unknown RF fingerprints, plus their coordinates and path loss coefficients that are embedded into vendor-specific IE of Beacon frames.

To perform a network scan, the following command is executed i.e. *iw dev wlan0 scan –u >> filename*. Here, *wlan0* is the name of the wireless interface, while the *–u* option is to instruct the *mac80211* kernel to return all IE fields inside the received Beacon frame because by default the vendor specific IEs are ignored. A typical scan result returned by *iw* tool provides information regarding the wireless networks in range and their capabilities. For each detected network, the main parameters reported include MAC address of the AP, its received signal level in dBm, timing information for network synchronization, the operating frequency and channel number, the supported basic and extended data rates, type of encryption, and other miscellaneous data described by the various IE, including the vendor-specific IE. Fig. 5.5 shows a screenshot of a typical scan result returned by *iw*.

All the scan results returned by *iw* are first saved into a text file i.e. *filename*. A Matlab script then parses this file to extract the necessary information i.e. MAC addresses, RSS, coordinates and path loss coefficients of location beacons heard on site and proceeds with computing the location results using the proposed incremental radio map calibration and RF fingerprint search algorithm. This work uses Matlab version R2013a running on an Intel-based computer with 2.8GHz processor and 4GB of RAM with 32-bit Windows 7 OS to perform the location estimation computation.

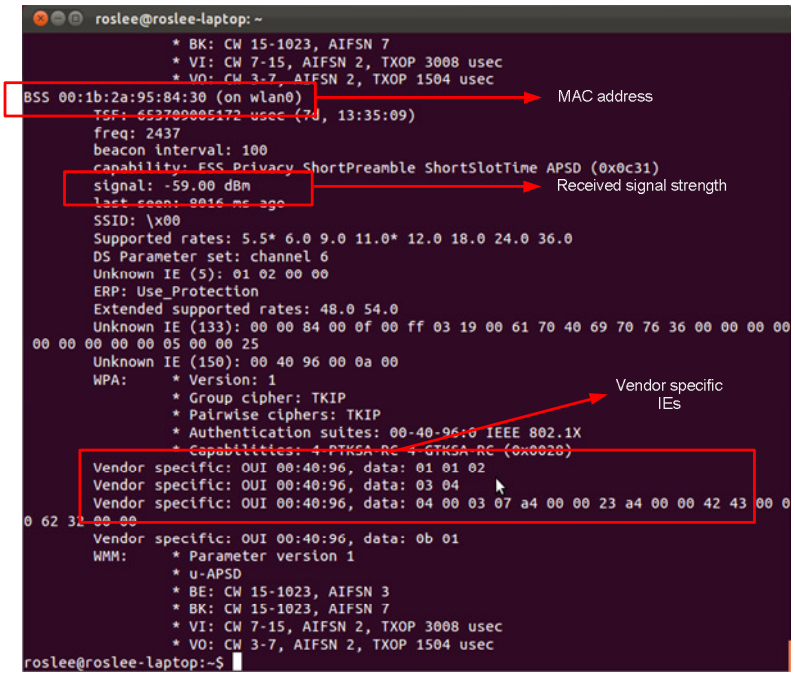


Fig. 5.5: Typical wireless network scan result of *iw* tool

5.4. Experiment Results & Discussions

5.4.1. Experimental Setup and Test Scenarios

An experiment was conducted to demonstrate real-world usability of the positioning system prototype. The chosen test site is a one-storey university building of brick and dry walls with concrete floor construction. It is approximately 510m^2 in size and has several classrooms separated by a hallway in the middle and a lounge area at one end of the building. The test site has five existing Wi-Fi APs. During the day, the site typically experiences significant student traffic with frequent reorganizations of desks/chairs and opening/closing of doors, all of which affect the RF fingerprints in temporal manner, thus providing an ideal test bed for evaluating the performance of the proposed system. In addition to temporal changes, the site's RF fingerprints were also altered spatially by adding, removing and relocating the APs. Fig. 5.6 shows its floor plan.

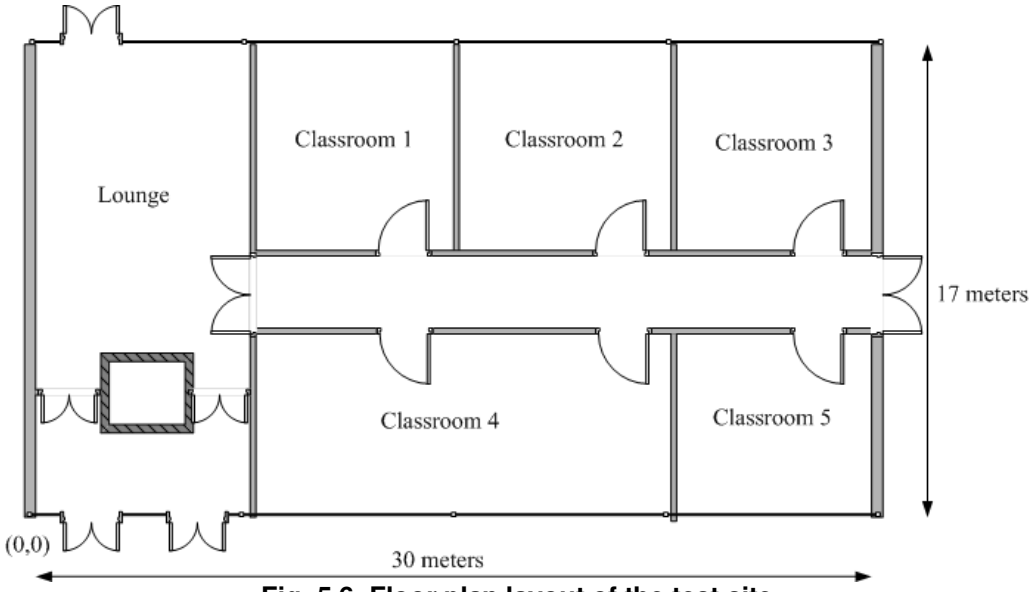


Fig. 5.6: Floor plan layout of the test site

The first experiment investigates the positioning system performance under varying wireless channel conditions. To simulate these conditions, the experiment was conducted at two different times, namely during and after office hours. During office hours, the test site experiences substantial crowd movements, as well as opening and closing of doors, simulating a dynamically changing radio propagation characteristic. In comparison, the site is usually empty after office hours, so the wireless channel should remain relatively calm.

The experiment setup comprises of $M = 5$ software-enhanced wireless routers that are deployed within the test site following an initial network topology as shown in Fig. 5.7. Next, 40 random points were selected to serve as the test locations. Using the location sensor prototype, unknown RF fingerprints of these test points are obtained, along with the coordinates and real-time path loss coefficients of the wireless routers embedded into the vendor-specific IEs of beacon and probe response frames. These data are then post-processed in Matlab using the proposed algorithm to yield the location estimates. The system's performance is then assessed in terms of positioning errors between the actual and estimated locations.

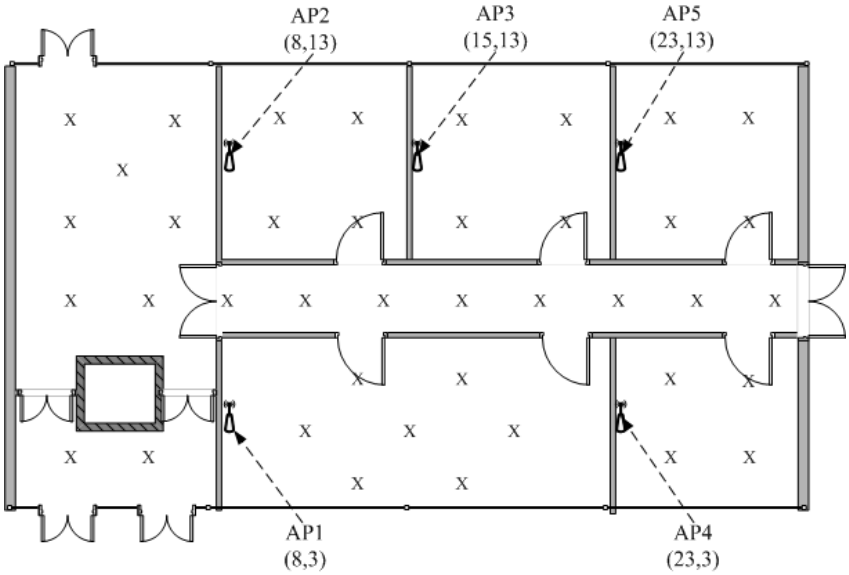


Fig. 5.7: Experimental setup for evaluating the system prototype under varying channel conditions. The "x" denotes the test points

In actual Wi-Fi deployments, new APs can be commissioned to improve coverage and bandwidth, while existing ones removed or relocated to minimize interference. Unfortunately, changing the wireless network topology alters the radio map of the site, which may degrade the accuracy of RF fingerprint positioning systems. Thus, a second experiment is conducted to assess the performance of the positioning system prototype in these scenarios, specifically relocation and removal/addition of wireless routers relative to an initial topology shown in Fig. 5.7. Furthermore, the experiment was conducted after office hours to minimize possible errors caused by varying wireless channel conditions.

First, three out of five wireless routers were relocated as shown in Fig. 5.8(a). Based on this new topology, another round of positioning tests was conducted on the 40 chosen test locations. Next, two wireless routers were removed from the initial topology to yield a new topology as shown in Fig. 5.8(b). Again, online positioning on the 40 chosen test locations were repeated based on this three-router setup. The results of these two experiments were compared against the results of the five-router setup of the initial network topology from the first experiment. Notice that

the setup of the last experiment also implicitly demonstrates the scenario when new APs are added to the site. This can be seen by assuming the three- and five-router setup as the old and new network topology, respectively. Hence, for analysing the prototype positioning performance when new APs are commissioned, the corresponding positioning results can be simply reversed.

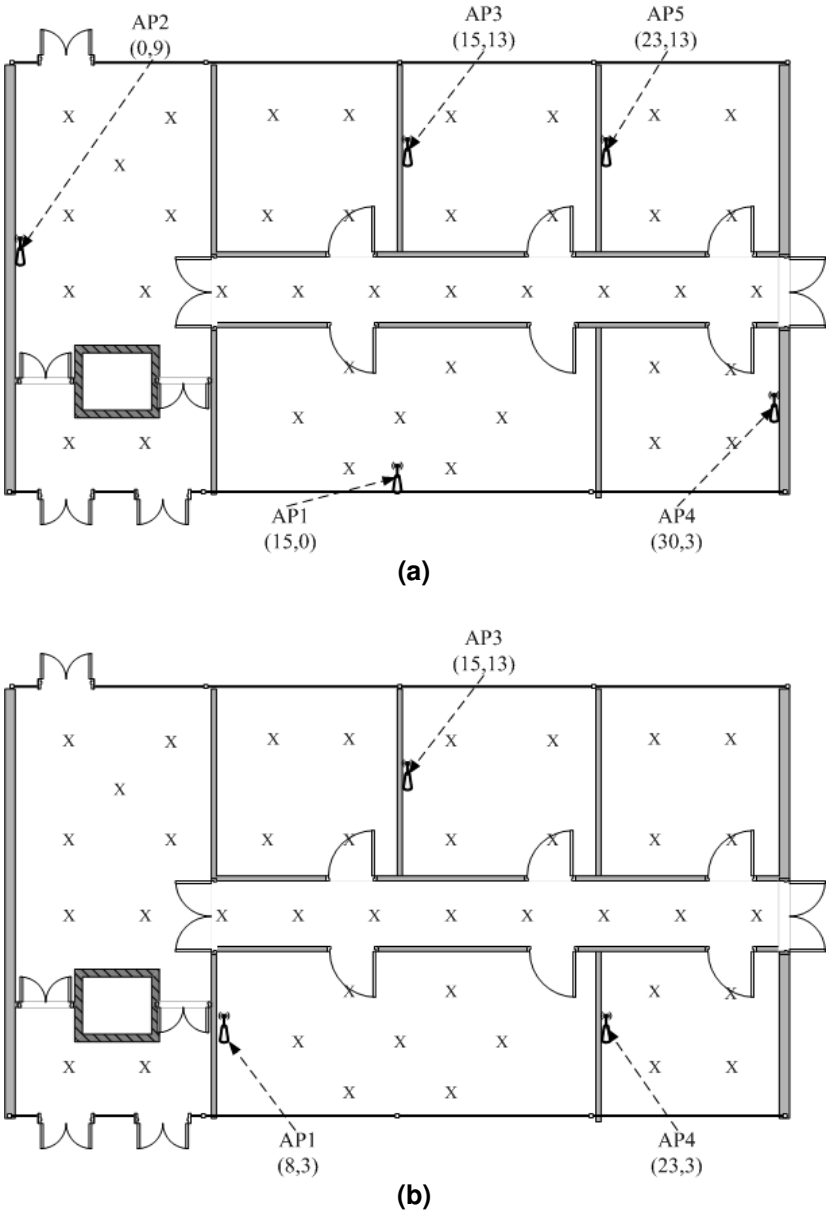


Fig. 5.8: Experimental setup for evaluating the system prototype under changing wireless network topologies. The "x" denotes the test points

5.4.2. Results and Discussions

Fig. 5.9 plots the cumulative positioning error of the indoor positioning system prototype during and after office hours, representing the "busy" and "quiet" wireless channel conditions, respectively. Due to the small number of test points i.e. 40, the cumulative positioning error curve appears step-like when plotted. The result shows that the error values are relatively similar to one another, which demonstrates the ability of the system to maintain its positioning performance despite the dynamic indoor radio propagation environments. This is due to the system's ability to monitor and adapt to the site's RF fingerprint variations in real-time, through the path loss self-modelling feature of the location beacons.

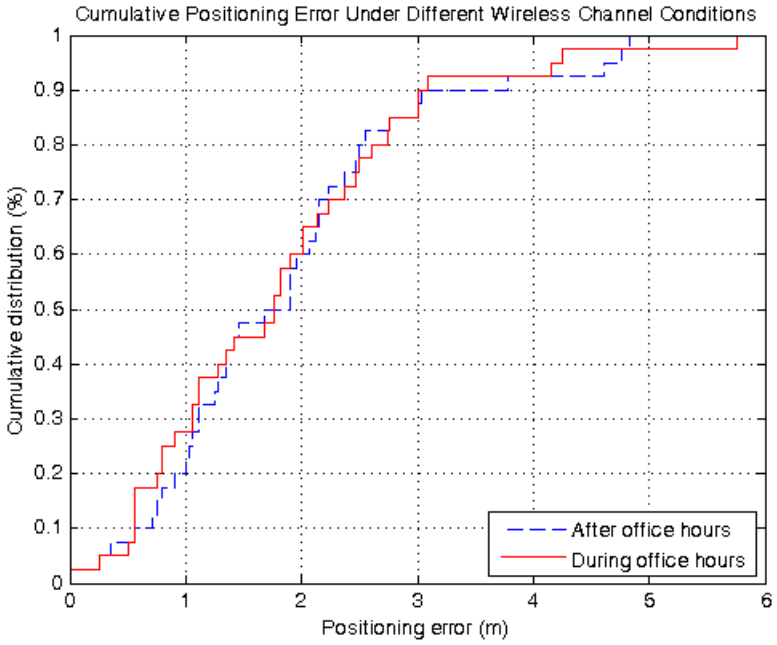


Fig. 5.9: Cumulative positioning error of the positioning system prototype under changing wireless channel conditions

Fig. 5.10 plots the cumulative positioning errors of the system prototype as a function of changing network topology due to relocation of location beacons. Relocating the location beacons also changes the signal-spatial correlations of the RF fingerprints, even when the indoor radio propagation channel does not experience any changes. This is because the radio signal from each relocated

location beacon must now travel different path and possibly encounter different obstructions to reach the same point, resulting in different RSS values compared to previous topology. Since the proposed algorithm always recalibrates a new radio map prior to computing the location result, changing of location beacon topologies does not affect positioning performance of the system because the newly calibrated radio map would have produce a new set of RF fingerprints that reflects the variations caused by topology changes. As can be seen from the results, positioning errors of the system prototype are still relatively similar despite the different location beacon topologies.

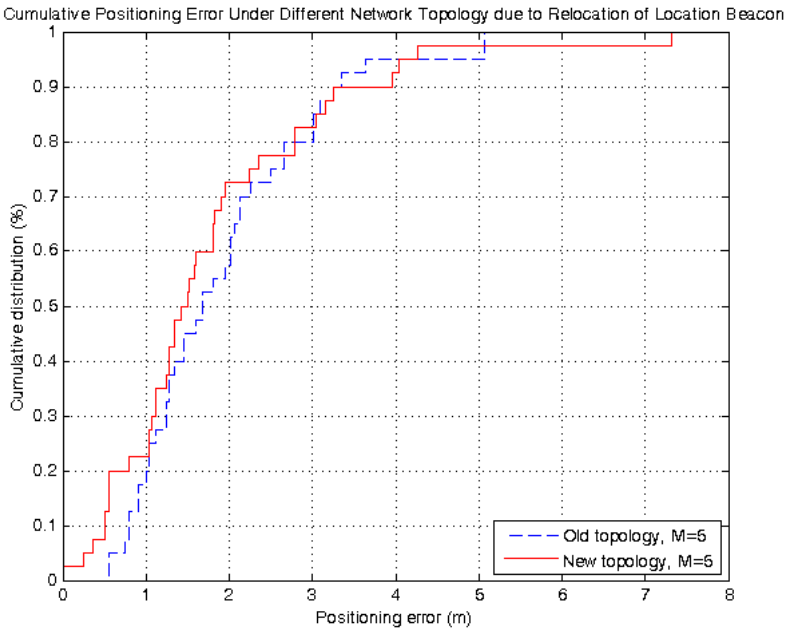


Fig. 5.10: Cumulative positioning error of the positioning system prototype under changing network topology due to relocation of location beacons

Meanwhile, Fig. 5.11 plots the cumulative positioning errors of the system prototype as a function of changing network topology caused by removal or addition of location beacons. From the results, Topology A with 3 location beacons has higher positioning error compared to Topology B with 5 location beacons, indicating the performance of the system prototype degrades as the location beacon density is reduced, and vice-versa. This is consistent with the simulation results from Chapter 3, as well as the findings in (Krishnakumar & Krishnan, 2005).

Moreover, in the context of the proposed system, each location beacon has lesser reciprocal RSS measurements to self-model its path loss radio propagation channel when the number of location beacon is reduced, which might yield less accurate model coefficients. Consequently, this error is propagated to the RF fingerprints when calibrating the radio map, resulting in increased positioning error.

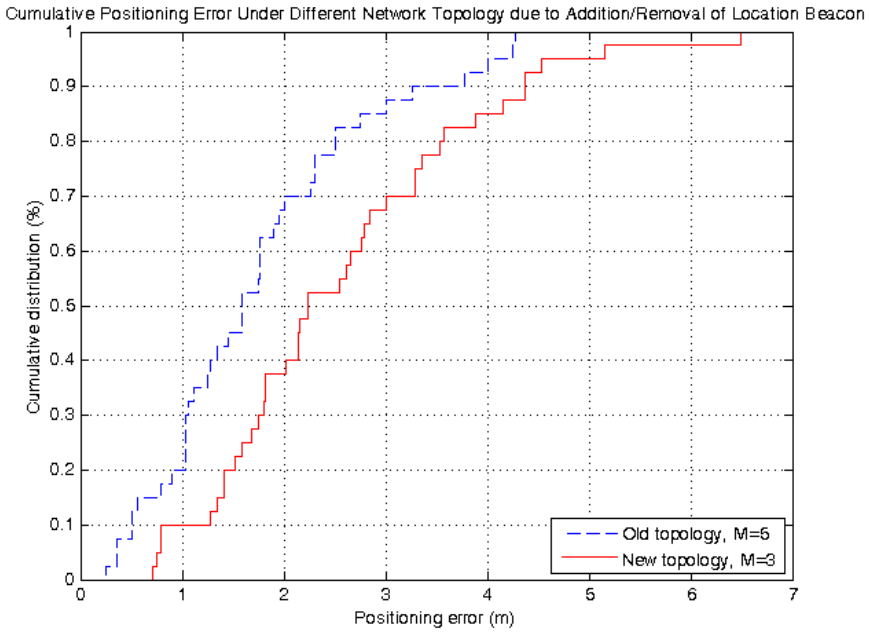


Fig. 5.11: Cumulative positioning error of the positioning system prototype under changing network topology due to addition/removal of location beacons

5.5. Summary

This thesis develops a prototype of the proposed distributed and client-centric adaptive RF fingerprint indoor positioning system to demonstrate its feasibility and evaluate its performance in real-world settings. The prototype is realized using only commodity Wi-Fi hardware, while all positioning-related functionalities are implemented in a two-part software developed using open-source framework, libraries and tools provided by the Linux-based OpenWRT and Ubuntu operating systems, together with Matlab to implement the proposed location estimation algorithm.

The prototype was deployed in a small university building to assess its positioning performance. Experimental results show the prototype can yield consistent positioning accuracy, despite the dynamically changing indoor radio propagation environment. Even when subjected to changing reference topology caused by relocation of location beacons, its positioning performance remains consistent. Moreover, the prototype works best where there are many location beacons due to better fingerprint resolution and the ability of each location beacon to self-calibrate its path loss coefficients more accurately.

Chapter 6: Conclusion

6.1. Introduction

Adaptive RF fingerprint methodology addresses the issues of costly radio map calibration and outdated RF fingerprints caused by temporal variations of the indoor radio propagation environments. From literature review, state-of-the-art implementations are highly server-centric, which are less robust, do not scale well and suffer from privacy issues. This thesis aims to address these drawbacks via implementation of the adaptive RF fingerprint methodology following distributed system architecture with client-centric positioning.

The reference infrastructure of the proposed indoor positioning system comprises a network of location beacons deployed at known locations within the target area. Each location beacon constantly advertises its coordinate and coefficients that model its RSS distribution around the transmitting antenna; this work employs the log-distance path loss with lognormal shadowing model to characterize the RSS distribution. Client devices listen for these parameters from nearby location beacons and use them to compute the beacons' RSS values at arbitrary points inside the target area, effectively calibrating the site's radio map on-demand.

Unfortunately, strengths of radio signals fluctuate over time due to physical and environmental factors. This reduces the signal-spatial correlation of the radio map and leads to degradation in positioning performance. In view of this, the proposed location beacon has the capability to calibrate its path loss coefficients in real-time to adapt the model to these RSS fluctuations. This ensures the radio map simulated using radio propagation modelling accurately depicts the site's actual RF fingerprints for improved positioning accuracy. The path loss self-modelling takes place in a fully distributed fashion via collaborative assistance from neighbouring location beacons, which is the first of its kind in adaptive RF fingerprint indoor positioning research.

Each time client device needs to solve for locations, it must first calibrate a full radio map and then perform an exhaustive search to find the best RF fingerprint match. These tasks impose a significant computing burden on client devices, where the amount of processing time is linearly proportional to the number of calibration points and the number of detected location beacons. As system improvement, this thesis proposes a novel algorithm that reduces the amount of computations in both dimensions in a principled way to minimize computing burden of client devices. Based on simulation results, the proposed algorithm only requires half the time to yield similar positioning accuracy compared to conventional RF fingerprint algorithm.

As part of its contributions, this thesis implements a prototype of the proposed adaptive RF fingerprint indoor positioning system using commodity Wi-Fi hardware to demonstrate its feasibility and evaluate its performance in real-world deployment. An off-the-shelf Linksys WRT54GL wireless router implements the self-modelling location beacon role via novel software developed as part of this thesis contribution. Meanwhile, a Wi-Fi enabled laptop running Linux Ubuntu OS implements the client device role. The laptop mainly performs network scans using Linux default wireless configuration tool to obtain the unknown RF fingerprints, as well as coordinates and path loss coefficients of all location beacons heard on site and saves these data into text files. Location estimation takes place offline using the saved data as inputs to the proposed incremental adaptive RF fingerprint algorithm implemented in Matlab. Based on experimental results, the prototype yields consistent positioning performance even when subjected to varying wireless channel conditions and changing network topologies.

6.2. Summary of Thesis Contributions

This thesis proposes a novel implementation of adaptive RF fingerprint indoor positioning system, both on architectural and algorithmic levels, to address the

robustness, scalability and privacy drawbacks associated with server-centric architecture of state-of-the-art implementations.

Architecturally, the proposed system implements the adaptive RF fingerprint methodology in a distributed manner with client-centric positioning, which is a novel approach not explored in previous works. This eliminates the needs to maintain location server and networking infrastructure for server access, which allows highly flexible and scalable deployment while improving the system's robustness. Together with its client-centric positioning model, the proposed system also protects user privacy.

Its distributed positioning infrastructure comprises a network of standard Wi-Fi routers that are software-enhanced to double as location beacons capable of self-modelling its path loss radio propagation in real-time. Development of this novel software is one contribution of this thesis.

The software development starts of by replacing the router's factory default firmware with an open-source Linux-based version called OpenWRT, which also comes with development tool chain that allows for development, compilation and installation of new software packages on the device. The location beacon functionality is implemented as a user space daemon and interacts with the wireless device driver inside the Linux kernel. When the daemon wakes up, it either listens for beacon or request for probe response frames of nearby wireless routers to measure their RSS. These measurements, together with its coordinate, are embedded inside the vendor-specific IE of beacon and probe response frames of the host router to be broadcasted later, so that neighbouring routers can use the measurements to calibrate their own path loss model coefficients. Similarly, the daemon parses the received packets looking for its own RSS as measured by its neighbours in order to calibrate its path loss model coefficients. Finally, these path loss coefficients are also embedded inside vendor-specific IE of beacon and probe

response frames for transmission to client devices. The daemon then sleeps for a specified period and repeats the whole process when it wakes up.

On the algorithmic level, this thesis proposes a lightweight radio map calibration and location estimation algorithm suitable for implementation on resource and power limited client devices. The proposed algorithm offers considerable savings in processing time while still delivering acceptable positioning accuracy compared to conventional RF fingerprint algorithm. It achieves this by limiting the radio map calibration and search to an optimal number of data points.

Given the unknown RF fingerprint to solve for location, the algorithm first sorts the RSS in descending order. Starting with the strongest location beacon heard on site, the algorithm first computes its RSS at each chosen calibration point and simultaneously compares it with the measured value. If the comparison falls outside a specified range, the algorithm removes that particular calibration point so there will be lesser RSS computation and comparison for the next strongest location beacon. Once the algorithm has computed and compared the RSS of all detected location beacons, it returns the average of remaining calibration points as the location result. In certain cases, the algorithm may already converge to produce location result without having to evaluate all detected location beacons, which further reduces the computing burden of client devices.

6.3. Future Work

6.3.1. Confidence Metric

Location errors are inherent in all positioning systems. Even the highly successful GNSS has positioning error of about 5-7 meters (Langley, 1999). Thus, providing a confidence metric that can quantify the severity of location errors is crucial for positioning systems.

Majority of RF fingerprint positioning systems proposed in the literature provide accuracy and precision metrics derived from cumulative distribution of location errors over many trials to evaluate their positioning performance, typically quoted as having *X* meter accuracy in *Y* percentage of the time (Elnahrawy et al., 2004). The information provided is probabilistic in nature and assumes enough positioning trials so the quoted performance figures are statistically applicable. This type of information offers little value for location-aware applications in evaluating the suitability of provided location estimates at arbitrary point in time. Moreover, conducting multiple positioning requests to arrive at statistically accurate location estimates is not possible for location-aware applications that provide real-time services.

There are two benefits of integrating the confidence metrics into the proposed indoor positioning algorithm. Firstly, it allows location-aware applications to decide whether the location results are accurate enough, since most applications have specific range of errors that they can tolerate where the service provided still makes sense. Secondly, a system with multiple positioning sources can weigh the various location results and choose the best one to use for a particular situation. Alternatively, using the confidence metric, multiple location estimates can be combined in an optimal manner using filtering algorithm such as Kalman or particle filter to yield a much improved result (Chai, Chen, Edwan, Zhang, & Loffeld, 2012; Kothari, Kannan, & Dias, 2011; Rodriguez & Gomez, 2009). Such techniques will be crucial for successful implementation of ubiquitous computing with the introduction of hybrid positioning systems (Bekkelien & Deriaz, 2012; Chen, 2011).

6.3.2. Smartphone-based Location Sensor

Although the location sensor is already operational on a laptop platform somewhat, it is still important to realize a smartphone version for two reasons. First is to conduct usability and performance evaluations that are more representative of real-world usages. Due to its status as a lifestyle device, users always carry their

smartphones everywhere and at all times, thus location information would be much more valuable on this platform. Secondly, hardware resources on smartphones are generally much more limited compared to laptops. This provides a good opportunity to investigate and improve the efficiency and power-awareness of the proposed adaptive RF fingerprint algorithm.

A hot research topic is hybrid positioning that fuses two or more location technologies together to solve the indoor positioning problem (Bekkelien & Deriaz, 2012). This is mainly due to proliferations of Wi-Fi enabled smartphones in recent years that come integrated with various sensors, such as GNSS receiver, accelerometer, digital compass and gyroscope (Chen, 2011). With the help of GNSS receiver, it is not difficult to locate an outdoor mobile user with several meters accuracy, but the positioning service quickly degrades once the user moves indoors. Pedestrian indoor navigation based on dead reckoning using low cost inertial sensors embedded into smartphones can fill-in the void left by GNSS somewhat but the methodology also has its own set of technical challenges. For one, dead reckoning requires an initial position to work, which GNSS receiver can provide while the service has not degraded yet; for example, when user is at the entrance of a building. Nevertheless, the main challenge is ensuring the position information remains valid as user moves deep inside the building where GNSS service is no longer available. This is because dead reckoning algorithm uses displacement and rotational information from the sensors to compute the user's current position, so any errors in the sensors' outputs will accumulate and increases the positioning error over time until it is no longer reliable. Thus, dead reckoning requires periodic position inputs to reset its error. This is where the proposed indoor positioning system can help. Its decentralized and client-centric architecture mirrors those of GNSS, so it can naturally take over where GNSS left off. It would be interesting to investigate how the proposed system can complement existing hybrid positioning systems towards realizing the ubiquitous

positioning solution that will be the catalyst of growth for innovative Location Based Services (LBS).

6.4. Final Comments

This thesis presents a novel adaptive RF fingerprint indoor positioning system, designed to integrate seamlessly with Wi-Fi's infrastructure to take advantage of the technology's ubiquitous presence in many indoor environments. It also introduces a novel client-centric location estimation algorithm as system improvement. The proposed system requires minimal costs for deployment due to its software-centric nature. A basic system prototype was developed to demonstrate its feasibility and usability in real world conditions, and the results were quite promising. With this thesis' contributions, it is hoped that the vision of ubiquitous positioning may move one-step closer to reality.

References

Akl, R., Tummala, D., & Li, X. (2006, July 3-5, 2006). *Indoor Propagation Modeling at 2.4 GHz for IEEE 802.11 Networks.* Paper presented at the Proceedings of 6th International Multi Conference on Wireless and Optical Communications, IASTED, Banff, Canada.

Alonso, P. T., Rodriguez, H. M., & Barbolla, A. M. B. (2009, June 10, 2009). *Enhancing the Performance of Propagation Model-Based Positioning Algorithms.* Paper presented at the Proceedings of 3rd International Workshop on User-Centric Technologies and Applications, Salamanca, Spain.

Andersen, J. B., Rappaport, T. S., & Yoshida, S. (1995). Propagation measurements and models for wireless communications channels. *Communications Magazine, IEEE, 33*(1), 42-49.

Asadoorian, P., & Pesce, L. (2007). *Linksys WRT54G Ultimate Hacking*. Burlington, MA: Syngress Publishing.

Atia, M. M., Noureldin, A., & Korenberg, M. J. (2013). Dynamic Online-Calibrated Radio Maps for Indoor Positioning in Wireless Local Area Networks. *Mobile Computing, IEEE Transactions on, 12*(9), 1774-1787.

Ayuso, P. N., Gasca, R. M., & Lefevre, L. (2010). Communicating Between the Kernel and User-space in Linux Using Netlink Sockets. *Software Practice and Experience, 00*, 1-7.

Bahl, P., & Padmanabhan, V. N. (2000, 2000). *RADAR: an in-building RF-based user location and tracking system.* Paper presented at the Proceedings of 19th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2000.

Barsocchi, P., Lenzi, S., Chessa, S., & Giunta, G. (2009, April 26-29, 2009). *A Novel Approach to Indoor RSSI Localization by Automatic Calibration of the Wireless Propagation Model.* Paper presented at the IEEE 69th Vehicular Technology Conference, VTC Spring 2009.

Battiti, R., Nhat, T. L., & Villani, A. (2002). *Location-Aware Computing: A Neural Network Model For Determining Location In Wireless LANs* (No. DIT-02-0083): University of Trento.

Bekkelien, A., & Deriaz, M. (2012, Oct. 3-4, 2012). *Hybrid positioning framework for mobile devices.* Paper presented at the Ubiquitous Positioning, Indoor Navigation, and Location Based Service (UPINLBS).

Bhasker, E. S., Brown, S. W., & Griswold, W. G. (2004, Mar. 14-17, 2004). *Employing user feedback for fast, accurate, low-maintenance geolocationing.* Paper presented at the Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications, PerCom 2004.

Brunato, M., & Battiti, R. (2005). Statistical learning theory for location fingerprinting in wireless LANs. *Comput. Netw., 47*(6), 825-845.

Brunato, M., & Kalló, C. K. (2002). *Transparent Location Fingerprinting for Wireless Services.* Paper presented at the Proceedings of Med-Hoc-Net.

Capulli, F., Monti, C., Vari, M., & Mazzenga, F. (2006, Sept. 6-8 , 2006). *Path Loss Models for IEEE 802.11a Wireless Local Area Networks.* Paper presented at the 3rd International Symposium on Wireless Communication Systems, ISWCS '06.

Chai, W., Chen, C., Edwan, E., Zhang, J., & Loffeld, O. (2012, Oct. 3-4, 2012). *2D/3D indoor navigation based on multi-sensor assisted pedestrian navigation in Wi-Fi environments.* Paper presented at the Ubiquitous Positioning, Indoor Navigation, and Location Based Service (UPINLBS).

Chen, R. (2011). Approaching an Ubiquitous Positioning Solution for Indoor Navigation and Location-Based Service. *Journal of Global Positioning Systems, 10*(1), 1-2.

Dao, D., Rizos, C., & Wang, J. (2002). Location-based services: technical and business issues. *GPS Solutions, 6*(3), 169-178.

de Souza, R. S., & Lins, R. D. (2008, Oct. 14-16, 2008). *A new propagation model for 2.4 GHz wireless LAN.* Paper presented at the 14th Asia-Pacific Conference on Communications, APCC 2008.

Debus, W. (2006). *RF Path Loss & Transmission Distance Calculations* (No. 8545-0003-01): Axonn LLC.

Derr, K., & Manic, M. (2008, June 2008). *Wireless Indoor Location Estimation Based on Neural Network RSS Signature Recognition (LENSR).* Paper presented at the Proceedings of 3rd IEEE Conference on Industrial Electronics and Applications.

Dujovne, D., Turletti, T., & Filali, F. (2010). A Taxonomy of IEEE 802.11 Wireless Parameters and Open Source Measurement Tools. *Communications Surveys & Tutorials, IEEE, 12*(2), 249-262.

Elnahrawy, E., Xiaoyan, L., & Martin, R. P. (2004, Oct. 4-7, 2004). *The limits of localization using signal strength: a comparative study.* Paper presented at the First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, IEEE SECON 2004.

Gami, S., Krishnakumar, A. S., & Krishnan, P. (2004, March 21-25, 2004). *Infrastructure-based location estimation in WLAN.* Paper presented at the IEEE Wireless Communications and Networking Conference, WCNC 2004.

Gast, M. (2002). *802.11 Wireless Networks: The Definitive Guide*: O'Reilly.

Gschwandtner, F., & Schindhelm, C. K. (2011, Sept. 21-23, 2011). *Spontaneous privacy-friendly indoor positioning using enhanced WLAN beacons.* Paper presented at the International Conference on Indoor Positioning and Indoor Navigation (IPIN).

Gwon, Y., & Jain, R. (2004). *Error characteristics and calibration-free techniques for wireless LAN-based location estimation*. Paper presented at the Proceedings of the 2nd International Workshop on Mobility Management & Wireless Access Protocol, MobiWAC 2004, New York, USA.

Hatami, A. (2006). *Application of Channel Modeling for Indoor Localization Using TOA and RSS.* Worcester Polytechnic Institute.

Hyuk, L., Lu-Chuan, K., Hou, J. C., & Haiyun, L. (2006, April 2006). *Zero-Configuration, Robust Indoor Localization: Theory and Experimentation.* Paper presented at the 25th IEEE International Conference on Computer Communications, INFOCOM 2006.

Ivanov, S., Nett, E., & Schemmer, S. (2008, May 21-23, 2008). *Automatic WLAN localization for industrial automation.* Paper presented at the IEEE International Workshop on Factory Communication Systems, WFCS 2008.

Jie, Y., Qiang, Y., & Lionel, N. (2005, March 8-12, 2005). *Adaptive Temporal Radio Maps for Indoor Location Estimation.* Paper presented at the 3rd IEEE International Conference on Pervasive Computing and Communications, PerCom 2005.

Kaemarungsi, K. (2005). Design of Indoor Positioning Systems Based on Location Fingerprinting Technique.

Kao, K.-F., Liao, I. E., & Lyu, J.-S. (2010, Sept. 15-17, 2010). *An indoor location-based service using access points as signal strength data collectors.* Paper presented at the International Conference on Indoor Positioning and Indoor Navigation (IPIN).

Kim, Y., Chon, Y., & Cha, H. (2012). Smartphone-Based Collaborative and Autonomous Radio Fingerprinting. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 42*(1), 112-122.

Kothari, N., Kannan, B., & Dias, M. B. (2011). *Robust Indoor Localization on a Commercial Smartphone* (No. CMU-RI-TR-11-27): Carnegie Mellon University.

Krishnakumar, A. S., & Krishnan, P. (2005, March 13-17, 2005). *On the accuracy of signal strength-based estimation techniques.* Paper presented at the Proceedings of IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2005.

Krishnakumar, A. S., & Krishnan, P. (2005, 2005). *The theory and practice of signal strength-based location estimation.* Paper presented at the International Conference on Collaborative Computing: Networking, Applications and Worksharing.

Krishnan, P., Krishnakumar, A. S., Ju, W. H., Mallows, C., & Gamt, S. N. (2004, March 7-11, 2004). *A system for LEASE: location estimation assisted by stationary emitters for indoor RF wireless networks.* Paper presented at the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2004.

Krumm, J., & Platt, J. C. (2003). *Minimizing Calibration Effort for an Indoor 802.11 Device Location Measurement System* (In proceedings No. MSR-TR-2003-82): Microsoft Research.

Kwon, J., Dundar, B., & Varaiya, P. (2004, Sept. 26-29, 2004). *Hybrid algorithm for indoor positioning using wireless LAN.* Paper presented at the IEEE 60th Vehicular Technology Conference, VTC2004-Fall.

LaMarca, A., Chawathe, Y., Consolvo, S., Hightower, J., Smith, I., Scott, J., Sohn, T., Howard, J., Hughes, J., Potter, F., Tabert, J., Powledge, P., Boriello, G., & Schilit, B. (2005, May 8-13, 2005). *Place Lab: Device Positioning Using Radio Beacons in the Wild.* Paper presented at the 3rd International Conference on Pervasive Computing.

Langley, R. B. (1999). Dilution of Precision [Electronic Version]. *GPS World*, 52 - 59. Retrieved 20 December 2014 from http://www.gpsworld.com.

Lei, Y., Laaraiedh, M., Avrillon, S., & Uguen, B. (2011, Dec. 14-17, 2011). *Fingerprinting localization based on neural networks and ultra-wideband signals.* Paper presented at IEEE International Symposium on the Signal Processing and Information Technology (ISSPIT).

Li, B., Salter, J., Dempster, A. G., & Rizos, C. (2006). *Indoor positioning techniques based on wireless LAN.* Paper presented at the Proceedings of IEEE 1st International Conference on Wireless Broadband and Ultra Wideband Communications.

Linville, J. W. (2008). *Tux on the Air: The State of Linux Wireless Networking*. Paper presented at the Linux Conference, Ottawa, Canada.

Lo, C.-C., Hsu, L.-Y., & Tseng, Y.-C. (2012). Adaptive radio maps for pattern-matching localization via inter-beacon co-calibration. *Pervasive Mob. Comput., 8*(2), 282-291.

Lorincz, K., & Welsh, M. (2007). MoteTrack: a robust, decentralized approach to RF-based location tracking. *Personal Ubiquitous Computing., 11*(6), 489-503.

Mohd Sabri, R., & Arslan, T. (2011, November 29 – December 1, 2011). *A Real-Time Wi-Fi Indoor Positioning System*. Paper presented at the European Navigation Conference, ENC 2011.

Mohd Sabri, R., & Arslan, T. (2011, September 19–23, 2011). *Inferring Wi-Fi Angle-of-Arrival from Received Signal Strength Distribution*. Paper presented at the 24th International Technical Meeting of the Satellite Division of Institute of Navigation.

Mohd Sabri, R., & Arslan, T. (2010, September 21-24, 2010). *Accurate Mapping of Wi-Fi Access Point Location for Indoor Positioning*. Paper presented at the 23rd International Technical Meeting of the Satellite Division of Institute of Navigation.

Moraes, L. F. M. d., & Nunes, B. A. A. (2006). *Calibration-free WLAN location system based on dynamic mapping of signal strength.* Paper presented at the Proceedings of the 4th ACM international workshop on Mobility management and wireless access, Terromolinos, Spain.

OpenWRT Table of Hardware. (n.d.). Retrieved from https://wiki.openwrt.org/toh/start/

Pahlavan, K., Xinrong, L., & Makela, J. P. (2002). Indoor geolocation science and technology. *Communications Magazine, IEEE, 40*(2), 112-118.

Park, J.-g., Charrow, B., Curtis, D., Battat, J., Minkov, E., Hicks, J., et al. (2010). *Growing an organic indoor location system*. Paper presented at the Proceedings of the 8th International Conference on Mobile Systems, Applications and Services, San Francisco, California, USA.

Philipp, B. (2008). *Redpin - adaptive, zero-configuration indoor localization through user collaboration*. Paper presented at the Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environment, San Francisco, California, USA.

Rodriguez, M., & Gomez, J. (2009). Analysis of Three Different Kalman Filter Implementations for Agricultural Vehicle Positioning. *The Open Agriculture Journal, 3*, 13 - 19.

Roos, T., Myllymaki, P., Tirri, H., Misikangas, P., & Sievanen, J. (2002). A Probabilistic Approach to WLAN User Location Estimation. *International Journal of Wireless Information Networks, 9*(3), 155-164.

Saha, S., Chaudhuri, K., Sanghi, D., & Bhagwat, P. (2003, March 16-20, 2003). *Location determination of a mobile device using IEEE 802.11b access point signals.* Paper presented at the IEEE Wireless Communications and Networking, WCNC 2003.

Sarkar, T. K., Zhong, J., Kyungjung, K., Medouri, A., & Salazar-Palma, M. (2003). A survey of various propagation models for mobile communication. *Antennas and Propagation Magazine, IEEE, 45*(3), 51-82.

Seidel, S. Y., & Rappaport, T. S. (1992). 914 MHz Path Loss Prediction Models for Indoor Wireless Communications in Multifloored Buildings. *IEEE Transactions on Antennas and Propagation, 40*(2), 207-217.

Shih-Hau, F., & Tsung-Nan, L. (2008). Indoor Location System Based on Discriminant-Adaptive Neural Network in IEEE 802.11 Environments. *Neural Networks, IEEE Transactions on, 19*(11), 1973-1978.

Tabassam, A. A., Trsek, H., Heiss, S., & Jasperneite, J. (2009, Oct. 20-23, 2009). *Fast and seamless handover for secure mobile industrial applications with 802.11r.* Paper presented at the IEEE 34th Conference on Local Computer Networks, LCN 2009.

Vipin, M., & Srikanth, S. (2010, Jan. 2-4, 2010). *Analysis of open source drivers for IEEE 802.11 WLANs.* Paper presented at the International Conference on Wireless Communication and Sensor Computing, ICWCSC 2010.

Wang, H., Ma, L., Xu, Y., & Deng, Z. (2011, Aug. 26-27, 2011). *Dynamic Radio Map Construction for WLAN Indoor Location.* Paper presented at the International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)

Xiaoyong, C., & Qiang, Y. (2007). Reducing the Calibration Effort for Probabilistic Indoor Location Estimation. *Mobile Computing, IEEE Transactions on, 6*(6), 649-662.

Ximei, L., Chao, Z., & Jizhen, H. (2008, Oct. 12-14, 2008). *Adaptive Weights Weighted Centroid Localization Algorithm for Wireless Sensor Networks.* Paper presented at the 4th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM '08.

Youssef, M., Agrawala, A. (2003). *On the Optimality of WLAN Location Determination Systems* (No. UMIACS-TR 2003-29 and CS-TR 4459): University of Maryland.

Youssef, M. A., Agrawala, A., & Udaya Shankar, A. (2003, March 26-26, 2003). *WLAN location determination via clustering and probability distributions.* Paper presented at the Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications, PerCom 2003.

Zanetti, G. P., & Palazzi, C. E. (2010, Oct. 20-22, 2010). *Non-invasive node detection in IEEE 802.11 wireless networks.* Paper presented at the Wireless Days (WD), 2010 IFIP.

Appendices

A. Self-Modelling Location Beacon Source Codes i.e. minavd.c

```
#include <errno.h>
#include <stdio.h>
#include <string.h>
#include <net/if.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdbool.h>
#include <stdlib.h>
#include <syslog.h>
#include <ctype.h>
#include <stdint.h>
#include <time.h>
#include <getopt.h>
#include <math.h>
#include <pwd.h>
#include <signal.h>

#include <netlink/msg.h>
#include <netlink/attr.h>
#include <netlink/netlink.h>
#include <netlink/genl/genl.h>
#include <netlink/genl/family.h>
#include <netlink/genl/ctrl.h>
#include <curl/curl.h>
#include <linux/nl80211.h>

#define DAEMON_NAME "minavd"
#define MAX_BSSID 16
#define REF_PATHLOSS 40
#define VENDORIE_SIZE_APINFO 9
#define VENDORIE_SIZE_SCAN 20
#define VENDORIE_SIZE_DRONE 24
#define ETH_ALEN 6
#define DEFAULT_COEFF1 -40
#define DEFAULT_COEFF2 2
#define DEFAULT_COEFF3 0
#define DEFAULT_SCANCOUNT 1
#define DEFAULT_SLEEPTIME 10
#define DEFAULT_TXPOWER 1

typedef uint64_t u64;
typedef uint32_t u32;
typedef uint16_t u16;
typedef uint8_t u8;
```

```
static struct node *scanNode = NULL;
static struct node *modelNode = NULL;
static struct node *droneNode = NULL;
static char *own_bssid = NULL;

struct node {
    bool isDrone;
    bool isModel;
    char bssid[13];
    int coordx;
    int coordy;
    int coordz;
    float data1;
    float data2;
    float data3;
    struct node *next;
};

/* Definition of a beacon frame head */
struct mgmt_beacon_head {
    u16 frame_control;
    u16 duration;
    u8 da[6];
    u8 sa[6];
    u8 bssid[6];
    u16 seq_ctrl;
    u64 timestamp;
    u16 beacon_int;
    u16 capab_info;
    /* Tagged parameters */
    u8 tags[64];
};

/*************************************************************************
    Description: Format given string based on mode (used to remove
    colon (:) from MAC address i.e. 01:23:45:67:89:ab -> 0123456789ab)

    Params:
        *str - pointer to string
        mode - trim mode

    Returns: doesn't return anything
*************************************************************************/
void FormatString ( char *str, char mode ) {
    char ptr[strlen( str )+1];
    int i = 0, j = 0;
    while ( str[i] != '\0' ) {
        if ( str[i] != mode )
            ptr[j++] = str[i];
        i++;
    }

    ptr[j] = '\0';
    strcpy( str,ptr );
```

```
    return;
}

/*****************************************************************
    Description: Convert hex string to integer value

    Params:
        hexStr - hexadecimal string

    Returns: long integer result
*****************************************************************/
long int hex2int ( char hexStr[] ) {
    int bits = strlen( hexStr ) * 4;
    char *pEnd;
    long long int result = strtoll( hexStr,&pEnd,16 );

    if ( result >= ( 1LL << ( bits-1 ) ) )
        result -= ( 1LL << bits );

    return result;
}

/*****************************************************************
    Description: Parse MAC address from stream of bytes

    Params:
        *mac_addr - pointer to MAC address variable
    *arg - stream of bytes

    Returns: N/A
*****************************************************************/
void mac_addr_n2a ( char *mac_addr, unsigned char *arg ) {
    int i, l;

    l = 0;
    for ( i=0; i<ETH_ALEN; i++ ) {
        if ( i==0 ) {
            sprintf( mac_addr+l, "%02x", arg[i] );
            l += 2;
        } else {
            sprintf( mac_addr+l, ":%02x", arg[i] );
            l += 3;
        }
    }
}

/*****************************************************************
    Description: Convert hex byte to number

    Params:
        hexStr - hexadecimal byte

    Returns: hex byte number equivalent or -1 if error
*****************************************************************/
```

```c
static int hex2num ( char hexByte ) {
    if ( hexByte >= '0' && hexByte <= '9' )
        return hexByte - '0';
    if ( hexByte >= 'a' && hexByte <= 'f' )
        return hexByte - 'a' + 10;
    if ( hexByte >= 'A' && hexByte <= 'F' )
        return hexByte - 'A' + 10;
    return -1;
}

/**********************************************************************
    Description: Convert ASCII string to MAC address (no colon
    delimitors format)

    Params:
        txt - MAC address as a string (e.g., "001122334455")
    addr - Buffer for the MAC address (ETH_ALEN = 6 bytes)

    Returns: 0 on success, -1 on failure (e.g., string not a MAC
    address)
**********************************************************************/
int hwaddr_compact_aton ( const char *txt, u8 *addr ) {
    int i;

    for ( i = 0; i < 6; i++ ) {
        int a, b;

        a = hex2num( *txt++ );
        if ( a < 0 )
            return -1;
        b = hex2num( *txt++ );
        if ( b < 0 )
            return -1;
        *addr++ = ( a << 4 ) | b;
    }

    return 0;
}

/**********************************************************************
    Description: Get current system time and date

    Params: N/A

    Returns: Time and date struct
**********************************************************************/
struct tm disp_time() {

    time_t t = time( NULL );
    struct tm tm = *localtime( &t );
    return tm;
}
```

```
/*****************************************************************
Description: Execute shell command

    Params:
    cmd - shell command to execute

    Returns: 0 -> success, -ve -> error
*****************************************************************/
int ShellCmd ( char *cmd ) {

    FILE *pipe;
    char line[256];
    int err = 0;

    if ( !( pipe = popen( cmd,"r" ) ) ) {
        syslog( LOG_ERR, "Error opening pipe to shell" );
        err = -ENOMEM;
        goto exit;
    }

    while ( fgets( line, sizeof( line ), pipe ) ) {
        continue;
    }
    pclose( pipe );

exit:
    return err;
}

/*****************************************************************
    Description: Starting and stopping hostapd

    Params:
    mode - 0 -> stop hostapd, 1 -> start hostapd
    interface - wireless interface i.e. wlan0
    pidfile - path to hostapd process ID file
    confile - path to hostapd configuration file

    Returns: 0 -> success, -ve -> error
*****************************************************************/
int StartStopHostapd ( bool mode, char *interface, char *pidfile, char
    *confile ) {

    FILE *file;
    char line[8], cmd[128];
    int err = 0, pid;

    if ( mode ) {   // Start
        sprintf( cmd,"hostapd -P %s -B %s",pidfile,confile );
        err = ShellCmd( cmd );
    } else {    // Stop
        /* hostapd not running */
        if ( ( file = fopen( pidfile,"r" ) ) == NULL ) {
            goto enableIface;
```

```
        }

        while ( fgets( line, sizeof( line ), file ) ) {
            sscanf( line,"%d",&pid );
        }
        fclose( file );

        sprintf( cmd,"kill %d",pid );
        err = ShellCmd( cmd );
        if ( err < 0 ) {
            goto exit;
        }

        sleep( 1 );
enableIface:
        sprintf( cmd,"ifconfig %s up",interface );
        err = ShellCmd( cmd );
    }

exit:
    return err;
}

/*****************************************************************
Description: Update location server

    Params:
    mode - 0 -> stop hostapd, 1 -> start hostapd
    interface - wireless interface i.e. wlan0
    pidfile - path to hostapd process ID file
    confile - path to hostapd configuration file

    Returns: 0 -> success, -ve -> error
*****************************************************************/
static size_t write_file ( void *ptr, size_t size, size_t nmemb, FILE
    *stream ) {
    return size*nmemb;
}

int UpdateLocationServer ( CURL *curl, char *bssid, int x, int y, float
    data1, float data2, float data3 ) {

    FILE *file;
    CURLcode res;
    char line[256];
    int err = 0;

    /* temp file to store server response */
    if ( ( file = fopen( "/var/log/tmpbuf","w" ) ) == NULL ) {
        syslog( LOG_ERR, "Error opening temporary log file" );
        err = -1;
    }

    /* specify the POST data */
```

```
    sprintf( line,"action=write&mac=%s&data=%d,%d,%6.4f,%6.4f,%6.4f\n",
        bssid,x,y,data1,data2,data3 );
    curl_easy_setopt( curl, CURLOPT_POSTFIELDS, line );

    /* write response to temporary file */
    curl_easy_setopt( curl, CURLOPT_WRITEDATA, file );
    curl_easy_setopt( curl, CURLOPT_WRITEFUNCTION, write_file );

    /* perform the request, res will get the return code */
    res = curl_easy_perform( curl );
    if ( res != 0 ) {
        err = -2;
    }

    fclose( file );

    return err;
}

/*****************************************************************************
    Description: Compute average and standard deviation of measurements

    Params:
        *dataArray - pointer to data array
        count - number of data measurements
        *average - pointer to average variable
        *stdDev - pointer to standard deviation variabled

    Returns: N/A
*****************************************************************************/
void ComputeMeasStats ( float *dataArray, int count, float *average,
    float *stdDev ) {
    float sum = 0;
    float sum_sq = 0;
    float var;
    int i;

    if ( count == 1 ) {
        *average = dataArray[0];
        *stdDev = 0;
        return;
    }

    for ( i=0; i<count; i++ ) {
        sum += dataArray[i];
        sum_sq += dataArray[i]*dataArray[i];
    }

    *average = sum / count;
    var = sum_sq/count - ( *average )*( *average );
    *stdDev = sqrt( var );

    return;
}
```

```
/*****************************************************************
    Description: Data regression in minimum squared error sense

    Params:
        *xdata - pointer to independent measurement array
        *ydata - pointer to dependent measurement array
        cnt - number of measurements
        *coeff1 - pointer to channel coefficient 1
        *coeff2 - pointer to channel coefficient 2
        *coeff3 - pointer to channel coefficient 3

    Returns: N/A
*****************************************************************/
void LeastSquareEst ( float *xdata, float *ydata, int cnt, float
    *coeff1, float *coeff2, float *coeff3 ) {

    float SUMx = 0, SUMy = 0, SUMxy = 0, SUMxx = 0;
    float m, c, est, res[cnt], avg, std;
    int i;

    for ( i = 0; i < cnt; i++ ) {
        SUMx += xdata[i];
        SUMy += ydata[i];
        SUMxy += xdata[i]*ydata[i];
        SUMxx += xdata[i]*xdata[i];
    }

    m = ( SUMx*SUMy - cnt*SUMxy ) / ( SUMx*SUMx - cnt*SUMxx );
    c = ( SUMy - m*SUMx ) / cnt;

    for ( i = 0; i < cnt; i++ ) {
        est = m*xdata[i] + c;
        res[i] = ydata[i] - est;
    }
    ComputeMeasStats( res,cnt,&avg,&std );

    *coeff1 = c;
    *coeff2 = m/-10;
    *coeff3 = std;

    return;
}

/*****************************************************************
    Description: Compute the logarithmic distance between two points

    Params:
        x0/y0/z0 - coordinates of the first point
        x1/y1/z1 - coordinates of the second point
        *logDist - pointer to log distance variable

    Returns: 0 -> success; -1 -> error
*****************************************************************/
```

```
int ComputeLogDist ( int x0, int y0, int z0, int x1, int y1, int z1,
    float *logDist ) {

    float dist;

    dist = sqrt( ( ( x0-x1 )*( x0-x1 ) ) + ( ( y0-y1 )*( y0-y1 ) ) + (
( z0-z1 )*( z0-z1 ) ) );
    if ( dist < 1 ) {
        return -1;
    }

    *logDist = log10( dist );
    return 0;
}

/*********************************************************************
    Description: Compare the current and previous channel model
      parameters. If absolute difference is more than diffPct, update
      previous channel parameters with current values.

    Params:
        c1/c2/c3 - current channel parameters
        c1_prev/c2_prev/c3_prev - previous channel parameters
        diffPct - diff between current and previous measurements in %

    Returns: true -> if more than diffPct, false -> otherwise
*********************************************************************/
bool tooBig ( float param1, float param2, float diffPct ) {

    float diff;

    diff = abs( param1-param2 );
    if ( abs( param1*diffPct ) < diff ) {
        return true;
    }

    return false;
}

bool CompareChanModel ( float c1, float c2, float c3, float *c1_prev,
    float *c2_prev, float *c3_prev ) {

    bool result = false, c1Res = false, c2Res = false, c3Res = false;

    /* we set the difference percentage to 5% */
    c1Res = tooBig( c1, *c1_prev, 0.05 );
    c2Res = tooBig( c2, *c2_prev, 0.05 );
    c3Res = tooBig( c3, *c3_prev, 0.05 );

    /* if any current parameter is too big, update prev values */
    if ( c1Res || c2Res || c3Res ) {
        result = true;
        *c1_prev = c1;
        *c2_prev = c2;
```

```
        *c3_prev = c3;
    }

    return result;
}

/*****************************************************************
    Description: Create new wireless scan result

    Params:
        drone - Result is drone parameters
        model - Result is channel model measurements
        mac - MAC address of neighbouring APs
        x/y/z - Coordinates of neighbouring APs
        val1/2/3 - Result values

    Returns: struct node
*****************************************************************/
struct node * NewWifiScan ( bool drone, bool model, char *mac, int x,
    int y, int z, float val1, float val2, float val3 ) {

    struct node *tmp;

    tmp = ( struct node * ) malloc( sizeof( struct node ) );
    if ( tmp == NULL ) {
        syslog( LOG_ERR, "Error malloc" );
        return NULL;
    }
    tmp->isDrone = drone;
    tmp->isModel = model;
    strcpy( tmp->bssid,mac );
    tmp->coordx = x;
    tmp->coordy = y;
    tmp->coordz = z;
    tmp->data1 = val1;
    tmp->data2 = val2;
    tmp->data3 = val3;
    tmp->next = NULL;
    return tmp;
}

/*****************************************************************
    Description: Add wireless scan result into linked list

    Params:
        *startScan - pointer to beginning of scan results
        *newScan - pointer to new scan result
    Returns: N/A
*****************************************************************/
void AddWifiScan ( struct node *startScan, struct node *newScan ) {
    struct node *tmpScan;

    tmpScan = startScan;
    while ( tmpScan->next != NULL ) {
```

```
        tmpScan = tmpScan->next;
    }
    tmpScan->next = newScan;
}

/*****************************************************************
    Description: Delete one wireless scan result from linked list

    Params:
        **startScan - address of pointer to beginning of scan results

    Returns: N/A
*****************************************************************/
void DeleteWifiScan ( struct node **startScan ) {
    struct node *tmpScan = *startScan;

    if ( tmpScan == NULL ) {
        return;
    }

    *startScan = tmpScan->next;
    free( tmpScan );
}

/*****************************************************************
    Description: Delete all wireless scan result from a linked list

    Params:
        *startScan - pointer to beginning of scan results

    Returns: N/A
*****************************************************************/
void FreeWifiScan ( struct node **startScan ) {

    struct node *tmpScan;

    if ( *startScan == NULL ) {
        return;
    } else {
        while ( *startScan != NULL ) {
            tmpScan = *startScan;
            *startScan = tmpScan->next;
            free( tmpScan );
        }
        *startScan = NULL;
    }
}

/*****************************************************************
    Description: Compare origin of two wireless scan results

    Params:
        *firstScan - pointer to first scan result
        *secondScan - pointer to second scan result
```

```
    Returns: 0 - same origin, 1 - different origin
*****************************************************************/
int CompareWifiScan ( struct node *firstScan, struct node *secondScan )
{

    if ( ( firstScan->coordx == secondScan->coordx ) &&
         ( firstScan->coordy == secondScan->coordy ) &&
         ( firstScan->coordz == secondScan->coordz ) ) {
        return 0;
    } else {
        return 1;
    }
}

/*****************************************************************
    Description: Float to byte converter

    Params:
        value - data of type float
        *eid - address of byte

    Returns: address of updated byte
*****************************************************************/
u8 * flt2byte ( float value, u8 *byteAddr ) {

    int i;
    union {
        u8 bytes[4];
        float f;
    } u;

    u.f = value;
    for ( i=0; i<4; i++ ) {
        *byteAddr++ = u.bytes[3-i];
    }

    return byteAddr;
}

/*****************************************************************
    Description: Add new vendor information elements

    Params:
        mac - MAC address
        type - type of IE
        val1/2/3 - values of IE
        *eid - pointer to existing IE

    Returns: pointer to updated IE
*****************************************************************/
u8 * AddVendorInfo ( bool type, char *mac, float val1, float val2,
    float val3, u8 *eid ) {
```

```
    u8 tmpMac[ETH_ALEN];
    int i;

    hwaddr_compact_aton( mac,tmpMac );

    // vendor eid i.e. 221
    *eid++ = 0xdd;
    // if AP is a drone
    if ( type ) {
        // 22 bytes ie 3 OUI, 1 control, 6 MAC, 4 val1, 4 val2, 4 val3
        *eid++ = 0x16;
    } else {
        // 18 bytes i.e. 3 OUI, 1 control, 6 MAC, 4 val1, 4 val2
        *eid++ = 0x12;
    }
    *eid++ = 0xff;      // OUI #1
    *eid++ = 0xff;      // OUI #2
    *eid++ = 0xff;      // OUI #3
    if ( type ) {       // Control byte
        *eid++ = 0x11;
    } else {
        *eid++ = 0x01;
    }
    for ( i=0; i<ETH_ALEN;i++ ) {
        *eid++ = tmpMac[i];
    }
    eid = flt2byte ( val1, eid );
    eid = flt2byte ( val2, eid );
    if ( type ) {
        eid = flt2byte ( val3, eid );
    }

    return eid;
}

/*********************************************************************
    Description: Parse vendor information elements as string

    Params:
        *parseData - pointer to vendor IE result
        *viedata - raw vendor IE data
        startByte - start point of vendor IE byte
        lenByte - length of vendor IE data

    Returns: N/A
*********************************************************************/
static void ParseVendorInfo ( char *parseData, unsigned char *viedata,
    int startByte, int lenByte ) {

    int l, i;
    l = 0;
    for ( i=startByte; i<(startByte+lenByte); i++ ) {
        sprintf( parseData+l,"%02x",viedata[i] );
        l += 2;
```

```
    }
}

/*****************************************************************
    Description: Parse vendor information elements as float

    Params:
        *parseData - pointer to vendor IE result
        *viedata - raw vendor IE data
        startByte - start point of vendor IE byte

    Returns: N/A
*****************************************************************/
static void ParseVendorInfoFloat ( float *parseData, unsigned char
    *viedata, int startByte ) {

    int i, j;
    union {
        u8 bytes[4];
        float ival;
    } u;

    j = 0;
    for ( i=startByte; i<(startByte+4); i++ ) {
        u.bytes[3-j] = viedata[i];
        j++;
    }

    *parseData = u.ival;
}

/*****************************************************************
    Description: Parsing results returned from Wi-Fi scan

    Params:
        *mac - pointer to netlink message
        *arg - void pointer

    Returns: 0 - success, 1 - error
*****************************************************************/
int ParseWifiScan ( struct nl_msg *msg, void *arg ) {

    struct genlmsghdr *gnlh = nlmsg_data( nlmsg_hdr( msg ) );
    struct nlattr *tb[NL80211_ATTR_MAX+1];
    struct nlattr *bss[NL80211_BSS_MAX+1];
    int freq, sdbm = 0, ielen;
    unsigned char s, vielen;
    unsigned char *ie, *viedata;
    struct node *newWifiScan;
    char mac_bssid[20], dev[20], tmpie[128], bssid_drone[13];
    int xcoord = 0, ycoord = 0, zcoord = 0;
    float mean_rss = 0, std_rss = 0, coeff1=0, coeff2=0, coeff3=0;
    bool isModel = false, isDrone = false;
```

```
    static struct nla_policy bss_policy[NL80211_BSS_MAX + 1] = {
        [NL80211_BSS_BSSID] = { .type = NLA_UNSPEC },
        [NL80211_BSS_FREQUENCY] = { .type = NLA_U32 },
        [NL80211_BSS_TSF] = { .type = NLA_U64 },
        [NL80211_BSS_BEACON_INTERVAL] = { .type = NLA_U16 },
        [NL80211_BSS_CAPABILITY] = { .type = NLA_U16 },
        [NL80211_BSS_INFORMATION_ELEMENTS] = { .type = NLA_UNSPEC },
        [NL80211_BSS_SIGNAL_MBM] = { .type = NLA_U32 },
        [NL80211_BSS_SIGNAL_UNSPEC] = { .type = NLA_U8 },
        [NL80211_BSS_STATUS] = { .type = NLA_U32 },
        [NL80211_BSS_SEEN_MS_AGO] = { .type = NLA_U32 },
        [NL80211_BSS_BEACON_IES] = { .type = NLA_UNSPEC },
    };

    nla_parse( tb, NL80211_ATTR_MAX, genlmsg_attrdata( gnlh, 0 ),
genlmsg_attrlen( gnlh, 0 ), NULL );
    if ( !tb[NL80211_ATTR_BSS] )
        return NL_SKIP;

    if ( nla_parse_nested( bss, NL80211_BSS_MAX, tb[NL80211_ATTR_BSS],
bss_policy ) )
        return NL_SKIP;

    if ( !bss[NL80211_BSS_BSSID] )
        return NL_SKIP;

    mac_addr_n2a( mac_bssid, nla_data( bss[NL80211_BSS_BSSID] ) );
    if_indextoname( nla_get_u32( tb[NL80211_ATTR_IFINDEX] ), dev );
    FormatString( mac_bssid,':' );

    if ( bss[NL80211_BSS_FREQUENCY] ) {
        freq = nla_get_u32( bss[NL80211_BSS_FREQUENCY] );
    }

    if ( bss[NL80211_BSS_TSF] ) {
        unsigned long long tsf;
        tsf = (unsigned long long)nla_get_u64( bss[NL80211_BSS_TSF] );
    }

    if ( bss[NL80211_BSS_SIGNAL_MBM] ) {
        sdbm = nla_get_u32( bss[NL80211_BSS_SIGNAL_MBM] );
    }

    if ( bss[NL80211_BSS_SIGNAL_UNSPEC] ) {
        s = nla_get_u8( bss[NL80211_BSS_SIGNAL_UNSPEC] );
    }

    if ( bss[NL80211_BSS_INFORMATION_ELEMENTS] ) {
        ie = nla_data( bss[NL80211_BSS_INFORMATION_ELEMENTS] );
        ielen = nla_len( bss[NL80211_BSS_INFORMATION_ELEMENTS] );

        while ( ielen >= 2 && ielen >= ie[1] ) {
            if ( ie[0] == 221 ) {   //vendor ie
                vielen = ie[1];
```

```
                viedata = ie + 2;
                ParseVendorInfo( tmpie, viedata, 0, 3 );

                if ( ( strcmp( tmpie,"ffffff" )==0 ) ) {
                    switch ( viedata[3] ) {
                        case 0x00:
                            ParseVendorInfo( tmpie,viedata,4,1 );
                            xcoord = ( int )hex2int( tmpie );
                            ParseVendorInfo( tmpie,viedata,5,1 );
                            ycoord = ( int )hex2int( tmpie );
                            ParseVendorInfo( tmpie,viedata,6,1 );
                            zcoord = ( int )hex2int( tmpie );
                            break;
                        case 0x01:
                            ParseVendorInfo( tmpie,viedata,4,6 );
                            if ( strcmp( own_bssid,tmpie ) == 0 ) {
                                isModel = true;
                                ParseVendorInfoFloat( &mean_rss,
                                    viedata, 10);
                                ParseVendorInfoFloat( &std_rss,
                                    viedata, 14);
                            }
                            break;
                        case 0x11:
                            isDrone = true;
                            ParseVendorInfo( tmpie,viedata,4,6 );
                            strcpy( bssid_drone,tmpie );
                            ParseVendorInfoFloat( &coeff1,viedata,10 );
                            ParseVendorInfoFloat( &coeff2,viedata,14 );
                            ParseVendorInfoFloat( &coeff3,viedata,18 );
                            break;
                    }
                }
            }
            ielen -= ie[1]+2;
            ie += ie[1]+2;
        }
    }

    if ( isDrone ) {
        newWifiScan = NewWifiScan( 1, 1, bssid_drone, xcoord, ycoord,
            zcoord, coeff1, coeff2, coeff3 );
        if ( droneNode == NULL ) {
            droneNode = newWifiScan;
        } else {
            AddWifiScan( droneNode, newWifiScan );
        }
        isDrone = false;
    }

    if ( isModel ) {
        newWifiScan = NewWifiScan( 0, 1, own_bssid, xcoord, ycoord,
            zcoord, mean_rss, std_rss, 0 );
        if ( modelNode == NULL ) {
```

```
            modelNode = newWifiScan;
        } else {
            AddWifiScan( modelNode, newWifiScan );
        }
        isModel = false;
    } else {
        newWifiScan = NewWifiScan(0, 0, mac_bssid, 0, 0, 0, (float)
            sdbm/100,0,0 );
        if ( scanNode == NULL ) {
            scanNode = newWifiScan;
        } else {
            AddWifiScan( scanNode, newWifiScan );
        }
    }

    return NL_SKIP;
}

/***************************************************************************
    Description: Configure the wireless card to scan for Wifi networks

    Params:
        *sock - pointer to netlink socket
        genl_family - generic netlink family
        *interface - pointer to wireless interface

    Returns: 0 -> success, -ve -> error
***************************************************************************/
int StartWifiScan ( struct nl_sock *sock, int genl_family, char
    *interface ) {

    struct nl_msg *msg = NULL, *ssids = NULL;
    int err = 0;

    msg = nlmsg_alloc();
    ssids = nlmsg_alloc();
    if ( !msg || !ssids ) {
        syslog( LOG_ERR, "Unable to allocate netlink message" );
        nlmsg_free( msg );
        nlmsg_free( ssids );
        err = -ENOMEM;
        goto exit;
    }

    genlmsg_put(msg,0,0,genl_family,0,0,NL80211_CMD_TRIGGER_SCAN,0);
    nla_put_u32(msg,NL80211_ATTR_IFINDEX,if_nametoindex(interface));
    nla_put( ssids, 1, 0, "" );
    nla_put_nested( msg, NL80211_ATTR_SCAN_SSIDS, ssids );

    err = nl_send_auto_complete( sock, msg );
    nlmsg_free( msg );
    nlmsg_free( ssids );
    if ( err < 0 ) {
        syslog(LOG_ERR,"Error sending trigger scan netlink message");
```

```
        goto exit;
    }

    err = nl_recvmsgs_default( sock );
    if ( err < 0 ) {
        syslog(LOG_ERR,"Error receiving trigger scan netlink message");
        goto exit;
    }

    sleep( 2 );

    msg = nlmsg_alloc();
    if ( !msg ) {
        syslog( LOG_ERR, "Unable to allocate netlink message" );
        err = -ENOMEM;
        goto exit;
    }
    genlmsg_put(msg,0,0,genl_family,0,NLM_F_DUMP,NL80211_CMD_GET_SCAN,0
      );
    nla_put_u32(msg,NL80211_ATTR_IFINDEX,if_nametoindex(interface));

    err = nl_send_auto_complete( sock,msg );
    nlmsg_free( msg );
    if ( err < 0 ) {
        syslog( LOG_ERR, "Error sending get scan netlink message" );
        goto exit;
    }

    err = nl_socket_modify_cb( sock, NL_CB_VALID, NL_CB_CUSTOM,
      ParseWifiScan, NULL );
    if ( err < 0 ) {
        syslog( LOG_ERR, "Error receiving get scan netlink message" );
        goto exit;
    }

    err = nl_recvmsgs_default( sock );

exit:
    return err;
}

/*****************************************************************
    Description: Configure the wireless card to transmit wifi beacon
      frames

    Params:
        *sock - pointer to netlink socket
        genl_family - generic netlink family
        *interface - pointer to wireless interface
        head_len - length of beacon header
        *beacon - pointer to beacon header frame
        tail_len - length of beacon tail
        *tail - pointer to beacon tail frame
```

```
    Returns: 0 -> success, -ve -> error
**********************************************************************/
int SendWifiBeacon ( struct nl_sock *sock, int genl_family, char
    *interface, size_t head_len, struct mgmt_beacon_head *beacon,
    size_t tail_len, u8 *tail ) {

    struct nl_msg *msg = NULL;
    int err = 0;

    msg = nlmsg_alloc();
    if ( !msg ) {
        syslog( LOG_ERR, "Error allocating netlink message" );
        nlmsg_free( msg );
        err = -ENOMEM;
        goto exit;
    }

    genlmsg_put( msg,0,0,genl_family,0,0,NL80211_CMD_SET_BEACON,0 );
    nla_put( msg, NL80211_ATTR_BEACON_HEAD, head_len, beacon );
    nla_put( msg, NL80211_ATTR_BEACON_TAIL, tail_len, tail );
    nla_put_u32( msg,NL80211_ATTR_IFINDEX,if_nametoindex(interface) );
    nla_put_u32( msg, NL80211_ATTR_BEACON_INTERVAL, 100 );
    nla_put_u32( msg, NL80211_ATTR_DTIM_PERIOD, 0 );

    err = nl_send_auto_complete( sock, msg );
    nlmsg_free( msg );
    if ( err < 0 ) {
        syslog( LOG_ERR, "Error sending set beacon netlink message" );
        goto exit;
    }

    err = nl_recvmsgs_default( sock );

exit:
    return err;
}

/**********************************************************************
    Description: Daemon child signal handler

    Params:
        signum - signal error

    Returns: N/A
**********************************************************************/
static void child_handler ( int signum ) {
    switch ( signum ) {
        case SIGALRM: exit( EXIT_FAILURE ); break;
        case SIGUSR1: exit( EXIT_SUCCESS ); break;
        case SIGCHLD: exit( EXIT_FAILURE ); break;
    }
}

/**********************************************************************
```

```
    Description: Daemon initialization

    Params:
        *lockfile - pointer to lock file

    Returns: N/A
**********************************************************************/
static void daemonize ( const char *lockfile ) {
    pid_t pid, sid;
    int lfp = -1;

    /* already a daemon */
    if ( getppid() == 1 ) return;

    /* Create the lock file as the current user */
    if ( lockfile && lockfile[0] ) {
        lfp = open( lockfile,O_RDWR|O_CREAT,0640 );
        if ( lfp < 0 ) {
            syslog( LOG_ERR, "Unable to create lock file %s, code=%d
                  (%s)",lockfile, errno, strerror( errno ) );
            exit( EXIT_FAILURE );
        }
    }

    /* Trap signals that we expect to recieve */
    signal( SIGCHLD,child_handler );
    signal( SIGUSR1,child_handler );
    signal( SIGALRM,child_handler );

    /* Fork off the parent process */
    pid = fork();
    if ( pid < 0 ) {
        syslog( LOG_ERR, "Unable to fork daemon, code=%d (%s)",errno,
            strerror( errno ) );
        exit( EXIT_FAILURE );
    }

    /* If we got a good PID, then we can exit the parent process. */
    if ( pid > 0 ) {

        /* Wait for confirmation from the child via SIGTERM or SIGCHLD,
        or for two seconds to elapse (SIGALRM).  pause() should not
        return. */
        alarm( 2 );
        pause();
        exit( EXIT_FAILURE );
    }

    /* At this point we are executing as the child process */
    /* Cancel certain signals */
    signal( SIGCHLD,SIG_DFL ); /* A child process dies */
    signal( SIGTSTP,SIG_IGN ); /* Various TTY signals */
    signal( SIGTTOU,SIG_IGN );
    signal( SIGTTIN,SIG_IGN );
```

```
    signal( SIGHUP, SIG_IGN ); /* Ignore hangup signal */
    signal( SIGTERM,SIG_DFL ); /* Die on SIGTERM */

    /* Change the file mode mask */
    umask( 0 );

    /* Create a new SID for the child process */
    sid = setsid();
    if ( sid < 0 ) {
        syslog( LOG_ERR, "Unable to create a new session, code %d
            (%s)",errno, strerror( errno ) );
        exit( EXIT_FAILURE );
    }

    /* Change the current working directory.  This prevents the current
    directory from being locked; hence not being able to remove it. */
    if ( (chdir( "/" )) < 0 ) {
        syslog( LOG_ERR, "Unable to change directory to %s, code %d
            (%s)","/",errno, strerror( errno ) );
        exit( EXIT_FAILURE );
    }

    /* Close standard file descriptors */
    close( STDIN_FILENO );
    close( STDOUT_FILENO );
    close( STDERR_FILENO );
}

/************************************************************************
    Description: Netlink socket initialization

    Params:
        *family - pointer to generic netlink family

    Returns: netlink socket structure
************************************************************************/
struct nl_sock * init_socket ( int *family ) {

    struct nl_sock *sock = NULL;
    int genl_family = 0;

    sock = nl_socket_alloc();
    if ( !sock ) {
        syslog( LOG_ERR, "Error allocating netlink socket" );
        return NULL;
    }

    if ( genl_connect( sock ) ) {
        syslog( LOG_ERR, "Error connecting to generic netlink" );
        nl_socket_free( sock );
        return NULL;
    }

    genl_family = genl_ctrl_resolve( sock, "nl80211" );
```

```
    if ( genl_family < 0 ) {
        syslog( LOG_ERR, "Unable to resolve generic netlink family" );
        nl_socket_free( sock );
        return NULL;
    }
    *family = genl_family;

    return sock;
}

/*************************************************************************
    Description: Display help screen and terminate

    Params:
        extval - Error value

    Returns: N/A
*************************************************************************/
void print_help ( int extval ) {
    fprintf( stdout,"%s -x XCOORD -y YCOORD -z ZCOORD -i IFACE [-r] [-
        h] [-l] [-d YYYYMMDD -t HHMM] [-s SLEEP] [-n SCANCOUNT] [-p
        PIDFILE] [-b HOSTAPDCONF]\n\n",DAEMON_NAME );

    fprintf( stdout,"\t-x XCOORD\t\tSet the AP x-coordinate\n" );
    fprintf( stdout,"\t-y YCOORD\t\tSet the AP y-coordinate\n" );
    fprintf( stdout,"\t-z ZCOORD\t\tSet the AP z-coordinate\n" );
    fprintf( stdout,"\t-i IFACE\t\tName of the wireless interface\n" );
    fprintf( stdout,"\t-r\t\t\tSet this AP as remote drone i.e. not
        connected to network\n" );
    fprintf( stdout,"\t-h\t\t\tPrint this help and exit\n" );
    fprintf( stdout,"\t-l\t\t\tEnable logging\n" );
    fprintf( stdout,"\t-d YYYYMMDD -t HHMM\tSet the system date and
        time in 24-hour format\n" );
    fprintf( stdout,"\t-s SLEEP\t\tSet the daemon sleep period in
        minute (default is 10 minutes)\n" );
    fprintf( stdout,"\t-n SCANCOUNT\t\tSet the number of consecutive
        scan (default is 1 scan)\n" );
    fprintf( stdout,"\t-p PID\t\t\tAbsolute path to Hostapd PID file\n"
);

    fprintf( stdout,"\t-b CONF\t\t\tAbsolute path to Hostapd
        configuration file\n" );
    fprintf( stdout,"\n" );
    exit( extval );
}

/*************************************************************************
    Description: Main minavd function

    Params:
        command line arguments specified in help screen

    Returns: N/A
*************************************************************************/
```

```
int main ( int argc, char *argv[] ) {

    /* Process command line arguments */
    char cmd[128];
    char *interface = NULL, *pidfile = NULL, *confile = NULL;
    int scan_count = DEFAULT_SCANCOUNT;
    int sleep_in_minute = DEFAULT_SLEEPTIME;
    int txpower = DEFAULT_TXPOWER;
    int opt, date = 0, time = 0;
    bool isXcoord = false, isYcoord = false, isZcoord = false;
    bool isDrone = false, isLog = false, isDate = false;
    bool isTime = false;
    u8 own_xcoord = 0, own_ycoord = 0, own_zcoord = 0;

    if ( argc == 1 ) {
        fprintf(stderr,"Incorrect usage of %s program\n", DAEMON_NAME);
        print_help( EXIT_FAILURE );
    }

    while ( (opt=getopt(argc,argv,"x:y:z:i:e:rhld:t:s:n:p:b:"))!=-1 ) {
        switch ( opt ) {
        case 'x':
            isXcoord = true;
            own_xcoord = ( u8 ) ( atoi( optarg ) & 0xff );
            break;
        case 'y':
            isYcoord = true;
            own_ycoord = ( u8 ) ( atoi( optarg ) & 0xff );
            break;
        case 'z':
            isZcoord = true;
            own_zcoord = ( u8 ) ( atoi( optarg ) & 0xff );
            break;
        case 'i':
            interface = optarg;
            break;
        case 'r':
            isDrone = true;
            break;
        case 'h':
            print_help( EXIT_SUCCESS );
            break;
        case 'l':
            isLog = true;
            openlog( DAEMON_NAME, LOG_PID|LOG_CONS, LOG_LOCAL5 );
            break;
        case 'd':
            if ( strlen( optarg ) != 8 ) {
                fprintf( stderr,"%s: Wrong format for date specified,
                    %s\n", DAEMON_NAME, optarg );
                print_help( EXIT_FAILURE );
            } else {
                isDate = true;
                date = atoi( optarg );
```

```
                break;
            }
        case 't':
            if ( strlen( optarg ) != 4 ) {
                fprintf( stderr,"%s: Wrong format for time specified,
                    %s\n", DAEMON_NAME, optarg );
                print_help( EXIT_FAILURE );
            } else {
                isTime = true;
                time = atoi( optarg );
                break;
            }
        case 's':
            sleep_in_minute = atoi( optarg );
            break;
        case 'n':
            scan_count = atoi( optarg );
            break;
        case 'p':
            pidfile = optarg;
            break;
        case 'b':
            confile = optarg;
            break;
        case ':':
            print_help( EXIT_FAILURE );
        case '?':
            print_help( EXIT_FAILURE );
        }
    }

    if ( !isXcoord || !isYcoord || !isZcoord ) {
        fprintf( stderr,"%s: Missing mandatory AP location
            coordinate(s)\n", DAEMON_NAME );
        print_help( EXIT_FAILURE );
    }

    if ( interface == NULL ) {
        fprintf( stderr, "%s: Missing mandatory interface option i.e. -
            i IFACE\n", DAEMON_NAME );
        print_help( EXIT_FAILURE );
    }

    if ( isDate^isTime ) {
        fprintf( stderr,"%s: Both date and time must be specified\n",
            DAEMON_NAME );
        print_help( EXIT_FAILURE );
    }

    if ( isDate&isTime ) {
        sprintf( cmd,"date %d%d",date,time );
        if ( ShellCmd( cmd ) < 0 ) {
            fprintf( stdout,"%s: Unable to set system date & time.
                Using default values\n", DAEMON_NAME );
```

```
    }
}

if ( pidfile == NULL ) {
    pidfile = "/var/run/wifi-phy0.pid";
}

if ( confile == NULL ) {
    confile = "/var/run/hostapd-phy0.conf";
}

/* Program variables */
FILE *file;
bool runOnce = false, updateServer = false;
char line[256];
char *filepos;
char *ap_ssid = NULL, *new_bssid = NULL;
int err = 0, i, j, mCnt = 0;
struct mgmt_beacon_head *beacon = NULL;
struct nl_sock *sock = NULL;
int genl_family = 0;
struct node *mCurrPtr = NULL, *mNextPtr, *mMarkPtr = NULL;
struct node *dCurrPtr = NULL, *dNextPtr, *dMarkPtr = NULL;
struct node *sCurrPtr = NULL, *sNextPtr;
u8 *tail = NULL, *headpos, *tailpos;
size_t head_len = 0, tail_len = 0;
struct tm tm;
float *xdata, *ydata, *rdata;
float coeff1, coeff2, coeff3;
float coeff1_prev, coeff2_prev, coeff3_prev;
float ldist, meanRss, stdRss;
CURL *curl;

if ( isLog ) {
    syslog( LOG_INFO, "Starting daemon" );
}

/* Daemonize */
daemonize( "/var/lock/" DAEMON_NAME );

/* The big loop */
while ( true ) {
    /* run this part once to initialize certain variables. can be
    moved out of loop */
    if ( !runOnce ) {
        runOnce = true;

        /* get the access point identification parameters i.e. SSID
        and BSSID */
        /* for WRT54G router, we get these parameters from the
        hostapd configuration file */
        /* for other routers, may need to find from somewhere */
        if ( ( file = fopen( confile,"r" ) ) == NULL ) {
            if ( isLog ) {
```

```
            syslog( LOG_ERR, "Unable open hostapd configuration
                file" );
        }
        exit( EXIT_FAILURE );
    }

    while ( fgets( line, sizeof( line ), file ) ) {

        if ( ( line[0] == '#' ) || ( line[0] == '\0' ) ) {
            continue;
        }

        filepos = line;
        while ( *filepos != '\0' ) {
            if ( *filepos == '\n' ) {
                *filepos = '\0';
                break;
            }
            filepos++;
        }

        filepos = strchr( line, '=' );
        if ( filepos == NULL ) {
            continue;
        }
        *filepos = '\0';
        filepos++;

        if ( strcmp( line, "ssid" ) == 0 ) {
            ap_ssid = malloc( strlen( filepos )+1 );
            if ( ap_ssid == NULL ) {
                if ( isLog ) {
                    syslog( LOG_ERR, "Unable to allocate memory
                        for AP ssid" );
                }
                err = -ENOMEM;
                break;
            }
            strcpy( ap_ssid, filepos );
        } else if ( strcmp( line, "bssid" ) == 0 ) {
            own_bssid = malloc( strlen( filepos )+1 );
            if ( own_bssid == NULL ) {
                if ( isLog ) {
                    syslog( LOG_ERR, "Unable to allocate memory
                        for own bssid" );
                }
                err = -ENOMEM;
                break;
            }
            strcpy( own_bssid, filepos );
            FormatString( own_bssid,':' );

            new_bssid = malloc( sizeof( own_bssid ) );
            if ( new_bssid == NULL ) {
```

```
            if ( isLog ) {
                syslog( LOG_ERR, "Unable to allocate memory
                    for new bssid" );
            }
            err = -ENOMEM;
            break;
        }
        strcpy( new_bssid,own_bssid );
    }
}
fclose( file );

if ( err < 0 ) {
    free( ap_ssid );
    free( own_bssid );
    free( new_bssid );
    exit( EXIT_FAILURE );
}

if ( isLog ) {
    syslog( LOG_INFO, "AP identification SSID = %s, BSSID =
        %s", ap_ssid, own_bssid );
}

/* get the access point transmit power settting */
/* for WRT54G router, we get it from uci commands */
/* for other routers, may need to find from somewhere */
if (!(file=popen( "uci get wireless.radio0.txpower","r"))){
    syslog( LOG_ERR, "Error opening pipe to shell" );
    free( ap_ssid );
    free( own_bssid );
    free( new_bssid );
    exit( EXIT_FAILURE );
}

while ( fgets( line, sizeof( line ), file ) ) {
    txpower = atoi( line );
}
pclose( file );

if ( isLog ) {
    syslog( LOG_INFO, "AP transmit power setting = %d dBm",
        txpower );
}

/* modify BSSID because driver doesn't allow more than one
beacon frame having same BSSID. Here, we change the last
octet i.e. -1 if last octet if 0xf, else add 1 */
if ( new_bssid[strlen( new_bssid )-1] >= 15 ) {
    new_bssid[strlen( new_bssid )-1]--;
} else {
    new_bssid[strlen( new_bssid )-1]++;
}
```

```
/* configure header of beacon management frame */
beacon = ( struct mgmt_beacon_head * ) malloc( sizeof(
    struct mgmt_beacon_head ) );
if ( beacon == NULL ) {
    if ( isLog ) {
        syslog( LOG_ERR, "Unable to allocate beacon frame
            header buffer" );
    }
    free( ap_ssid );
    free( own_bssid );
    free( new_bssid );
    exit( EXIT_FAILURE );
}

beacon->frame_control = 0x0080;
beacon->duration = 0x0000;
memset( beacon->da,0xff,ETH_ALEN );
hwaddr_compact_aton( new_bssid,beacon->sa );
hwaddr_compact_aton( new_bssid,beacon->bssid );
beacon->beacon_int = 0x0064;    //100ms
beacon->capab_info = 0x0000 | 0x0001;

headpos = &beacon->tags[0];
*headpos++ = 0x00;
*headpos++ = ( u8 ) strlen( ap_ssid );
for ( i=0; i<strlen( ap_ssid ); i++ ) {
    *headpos++ = ( u8 ) ap_ssid[i];
}

/* these should depend on each AP and obtained from
hardware configuration somewhere. Here we hardcode them */
*headpos++ = 0x01;  /* Supported rates Element ID */
*headpos++ = 0x08;  /* Number of supported rates = 8 */
*headpos++ = 0x82;  /* Rate = 2*0.5Mbps = 1Mbps */
*headpos++ = 0x84;  /* Rate = 4*0.5Mbps = 4Mbps */
*headpos++ = 0x8B;  /* Rate = 11*0.5Mbps = 5.5Mbps */
*headpos++ = 0x96;  /* Rate = 22*0.5Mbps = 11Mbps */
*headpos++ = 0x24;  /* Rate = 36*0.5Mbps = 18Mbps */
*headpos++ = 0x30;  /* Rate = 48*0.5Mbps = 24Mbps */
*headpos++ = 0x48;  /* Rate = 72*0.5Mbps = 36Mbps */
*headpos++ = 0x6C;  /* Rate = 108*0.5Mbps = 54Mbps */
*headpos++ = 0x03;  /* DS Parameter Element ID */
*headpos++ = 0x01;  /* DS Parameter length */
*headpos++ = 0x01;
head_len = headpos - ( u8 * )beacon;

if ( isLog ) {
    syslog( LOG_INFO, "Beacon frame header configured" );
}

/* Set default channel parameters */
coeff1 = DEFAULT_COEFF1;
coeff2 = DEFAULT_COEFF2;
coeff3 = DEFAULT_COEFF3;
```

```
    coeff1_prev = DEFAULT_COEFF1;
    coeff2_prev = DEFAULT_COEFF2;
    coeff3_prev = DEFAULT_COEFF3;
}

tm = disp_time();

/* setup netlink socket and resolve genl family */
sock = init_socket( &genl_family );
if ( !sock ) {
    if ( isLog ) {
        syslog( LOG_ERR, "Cannot allocate netlink socket" );
    }
    free( ap_ssid );
    free( own_bssid );
    free( new_bssid );
    free( beacon );
    exit( EXIT_FAILURE );
}

if ( isLog ) {
    syslog( LOG_INFO, "Netlink socket initialized" );
}

/* scan the wireless channel */
/* first we have to stop hostapd, otherwise wireless card
cannot scan because it is in master mode */
err = StartStopHostapd( 0, interface, pidfile, confile );
if (err < 0) {
    if ( isLog ) {
        syslog( LOG_ERR, "Unable to stop hostapd" );
    }
    free( ap_ssid );
    free( own_bssid );
    free( new_bssid );
    free( beacon );
    nl_socket_free( sock );
    exit( EXIT_FAILURE );
}

if ( isLog ) {
    syslog( LOG_INFO, "Hostapd daemon stopped" );
}

/* scan the wireless channel based on scan_count */
i=0;
do {
    err = StartWifiScan( sock, genl_family, interface );
    if ( err < 0 ) {
        if ( isLog ) {
            syslog( LOG_ERR, "Wifi network scan failed" );
        }
        break;
    }
```

```
    i++;
} while ( i<scan_count );

if ( err < 0 ) {
    free( ap_ssid );
    free( own_bssid );
    free( new_bssid );
    free( beacon );
    FreeWifiScan( &scanNode );
    FreeWifiScan( &modelNode );
    FreeWifiScan( &droneNode );
    nl_socket_free( sock );
    exit( EXIT_FAILURE );
}

if ( isLog ) {
    syslog( LOG_INFO, "Network scan finished" );
    syslog( LOG_INFO, "Scan date/time: %d-%d-%d,
        %.2d:%.2d:%.2d", tm.tm_mday, tm.tm_mon+1,
        tm.tm_year+1900, tm.tm_hour, tm.tm_min, tm.tm_sec );
}

/* restart hostapd so AP can function as master mode again */
err = StartStopHostapd( 1, interface, pidfile, confile );
if ( err < 0 ) {
    if ( isLog ) {
        syslog( LOG_ERR, "Unable to restart hostapd daemon" );
    }
    free( ap_ssid );
    free( own_bssid );
    free( new_bssid );
    free( beacon );
    FreeWifiScan( &scanNode );
    FreeWifiScan( &modelNode );
    FreeWifiScan( &droneNode );
    nl_socket_free( sock );
    exit( EXIT_FAILURE );
}

if ( isLog ) {
    syslog( LOG_INFO, "Hostapd daemon restarted" );
}

/* post-process the wi-fi measurement results */
/* but first check if there is wi-fi scan result. If not, go to
sleep */
if ((scanNode==NULL)&&(modelNode==NULL)&&(droneNode==NULL)) {
    if ( isLog ) {
        syslog( LOG_INFO, "Scan did not find any nearby
            network(s)" );
    }
    goto goodnight;
}
```

```
/* if we perform scan more than once, then we might receive
   multiple duplicate reciprocal RSS measurements */
/* so, we first remove the duplicate measurements (if any) */
if ( modelNode != NULL ) {
    mCnt = 0;
    mMarkPtr = modelNode;
    while ( mMarkPtr != NULL ) {
        mCurrPtr = mMarkPtr;
        mNextPtr = mMarkPtr->next;
        while ( mNextPtr != NULL ) {
            if ( CompareWifiScan( mMarkPtr,mNextPtr ) == 0 ) {
                mCurrPtr->next = mNextPtr->next;
                free( mNextPtr );
                mNextPtr = mCurrPtr->next;
            } else {
                mCurrPtr = mNextPtr;
                mNextPtr = mNextPtr->next;
            }
        }
        mCnt++;
        mMarkPtr = mMarkPtr->next;
    }
}

/* Self-calibration of the AP channel models if enough
   reciprocal RSS measurements found */
if ( mCnt > 0 ) {
    xdata = ( float * ) malloc ( sizeof( float )*( mCnt+1 ) );
    ydata = ( float * ) malloc ( sizeof( float )*( mCnt+1 ) );
    if ( ( xdata == NULL ) || ( ydata == NULL ) ) {
        if ( isLog ) {
            syslog( LOG_ERR, "Cannot allocate measurement
                buffers" );
        }
        free( ap_ssid );
        free( own_bssid );
        free( new_bssid );
        free( beacon );
        FreeWifiScan( &scanNode );
        FreeWifiScan( &modelNode );
        FreeWifiScan( &droneNode );
        nl_socket_free( sock );
        exit( EXIT_FAILURE );
    }

    i=0;
    while ( modelNode != NULL ) {
        err = ComputeLogDist( (int)own_xcoord, (int)own_ycoord,
                (int)own_zcoord,modelNode->coordx,modelNode->
                coordy,modelNode->coordz,&ldist);

        if ( err < 0 ) {
            if ( isLog ) {
                syslog(LOG_ERR,"Log of distance is negative");
```

```
                }
                break;
            } else {
                if ( isLog ) {
                    syslog(LOG_INFO,"xdata[%d]=%6.4f,
                    ydata[%d]=%6.4f", i,ldist,i,modelNode->data1 );
                }
            }

            xdata[i] = ldist;
            ydata[i] = modelNode->data1;
            mCurrPtr = modelNode;
            modelNode = modelNode->next;
            free( mCurrPtr );
            i++;
        }
        modelNode = NULL;
        xdata[i] = 0;
        ydata[i] = ( float ) ( txpower - REF_PATHLOSS );
        if ( isLog ) {
            syslog(LOG_INFO, "xdata[%d]=%6.4f, ydata[%d]=%6.4f",
                i,xdata[i],i,ydata[i] );
        }
        i++;

        if ( err < 0 ) {
            free( xdata );
            free( ydata );
            free( ap_ssid );
            free( own_bssid );
            free( new_bssid );
            free( beacon );
            FreeWifiScan( &scanNode );
            FreeWifiScan( &modelNode );
            FreeWifiScan( &droneNode );
            nl_socket_free( sock );
            exit( EXIT_FAILURE );
        }

        /* Fit the measurement data to the radio propagation model
           in a least squared sense */
        LeastSquareEst( xdata,ydata,i,&coeff1,&coeff2,&coeff3 );
        free( xdata );
        free( ydata );

        if ( isLog ) {
            syslog( LOG_INFO, "AP propagation channel modelled
                (%6.4f,%6.4f,%6.4f)",coeff1,coeff2,coeff3 );
        }
    } else {
        FreeWifiScan( &modelNode );

        if ( isLog ) {
            syslog( LOG_INFO, "Not enough data to model channel" );
```

```
    }
}

/* Check if new channel models differ significantly from
   previous ones. If yes, udpdate server with new values */
updateServer = CompareChanModel( coeff1, coeff2, coeff3,
    &coeff1_prev, &coeff2_prev, &coeff3_prev );

/* compute neighbour AP's RSS and broadcast in beacon frames */
if ( scanNode != NULL ) {
    if ( isDrone ) {
        tailpos = tail = (u8 *)malloc(VENDORIE_SIZE_APINFO
          +(MAX_BSSID*VENDORIE_SIZE_SCAN)+VENDORIE_SIZE_DRONE);
    } else {
        tailpos = tail = (u8 *)malloc(VENDORIE_SIZE_APINFO +
          (MAX_BSSID*VENDORIE_SIZE_SCAN));
    }

    if ( tail == NULL ) {
        if ( isLog ) {
            syslog( LOG_ERR, "Unable to allocate beacon frame
                tail buffer");
        }
        free( ap_ssid );
        free( own_bssid );
        free( new_bssid );
        free( beacon );
        FreeWifiScan( &scanNode );
        FreeWifiScan( &droneNode );
        nl_socket_free( sock );
        exit( EXIT_FAILURE );
    }

    /* add AP coordinate into vendor IE of beacon frame */
    *tailpos++ = 0xdd; /* vendor ie */
    *tailpos++ = 0x07; /* ie length */
    *tailpos++ = 0xff; /* OUI #1 */
    *tailpos++ = 0xff; /* OUI #2 */
    *tailpos++ = 0xff; /* OUI #3 */
    *tailpos++ = 0x00; /* control */
    *tailpos++ = own_xcoord;
    *tailpos++ = own_ycoord;
    *tailpos++ = own_zcoord;

    i=0;
    while ( scanNode != NULL ) {
        /* count number of RSS measurements for each
           neighbouring AP */
        j=0;
        sCurrPtr = scanNode;
        while ( sCurrPtr != NULL ) {
            if (strcmp(sCurrPtr->bssid,scanNode->bssid) == 0) {
                j++;
            }
```

```
            sCurrPtr = sCurrPtr->next;
        }

        /* alloc sufficient buffer to store all RSS values */
        rdata = ( float * ) malloc( sizeof( float )*j );
        if ( rdata == NULL ) {
            if ( isLog ) {
                syslog(LOG_INFO, "Unable to allocate memory for
                    RSS measurements");
            }
            err = -ENOMEM;
            break;
        }

        // store RSS values into buffer
        j=0;
        sCurrPtr = scanNode;
        sNextPtr = scanNode->next;
        rdata[j++] = sCurrPtr->data1;
        while ( sNextPtr != NULL ) {
            if (strcmp(sNextPtr->bssid,scanNode->bssid)==0) {
                rdata[j++] = sNextPtr->data1;
                sCurrPtr->next = sNextPtr->next;
                free( sNextPtr );
                sNextPtr = sCurrPtr->next;
            } else {
                sCurrPtr = sNextPtr;
                sNextPtr = sNextPtr->next;
            }
        }

        /* compute mean and standard deviation of RSS
           measurements */
        ComputeMeasStats( rdata, j, &meanRss, &stdRss );

        if ( isLog ) {
            syslog( LOG_INFO, "MAC = %s, Mean = %6.4f dBm, Std
                = %6.4f dBm",scanNode->bssid,meanRss,stdRss );
        }

        /* add mac address, average and std rss into beacon
           tail. We limit to MAX_BSSID because that is how many
           our beacon frame buffer can hold */
        if ( i < MAX_BSSID ) {
            tailpos = AddVendorInfo( false, scanNode->bssid,
                meanRss, stdRss, 0, tailpos );
        }
        DeleteWifiScan( &scanNode );
        free( rdata );
        i++;
    }

    if ( err < 0 ) {
        free( ap_ssid );
```

```
            free( own_bssid );
            free( new_bssid );
            free( beacon );
            free( tail );
            FreeWifiScan( &scanNode );
            FreeWifiScan( &droneNode );
            nl_socket_free( sock );
            exit( EXIT_FAILURE );
        }

        // Broadcast channel model if AP is a drone
        if ( isDrone ) {
            if ( updateServer ) {
                tailpos = AddVendorInfo( true, own_bssid, coeff1,
                    coeff2, coeff3, tailpos );
            } else {
                tailpos = AddVendorInfo( true, own_bssid,
                    coeff1_prev,coeff2_prev,coeff3_prev,tailpos);
            }
        }

        // configure beacon frame for transmission
        tail_len = tailpos-tail;
        err = SendWifiBeacon( sock, genl_family, interface,
                  head_len, beacon, tail_len, tail );
        if ( err < 0 ) {
            if ( isLog ) {
                syslog(LOG_ERR,"Unable to transmit beacon frame");
            }
            free( ap_ssid );
            free( own_bssid );
            free( new_bssid );
            free( beacon );
            free( tail );
            FreeWifiScan( &droneNode );
            nl_socket_free( sock );
            exit( EXIT_FAILURE );
        }

        free( tail );

        if ( isLog ) {
            syslog(LOG_INFO, "Beacon frame configured for
                transmission");
        }
    }

    if ( isDrone ) {
        /* free the drone list (if any) since drone cannot update
           location server */
        FreeWifiScan( &droneNode );
    } else {
        /* remove duplicate drone nodes (if any) */
        dMarkPtr = droneNode;
```

```
    while ( dMarkPtr != NULL ) {
        dCurrPtr = dMarkPtr;
        dNextPtr = dMarkPtr->next;
        while ( dNextPtr != NULL ) {
            if ( CompareWifiScan( dMarkPtr,dNextPtr ) == 0 ) {
                dCurrPtr->next = dNextPtr->next;
                free( dNextPtr );
                dNextPtr = dCurrPtr->next;
            } else {
                dCurrPtr = dNextPtr;
                dNextPtr = dNextPtr->next;
            }
        }
        dMarkPtr = dMarkPtr->next;
    }
}

/* send channel parameters to location server if AP is not a
   remote drone */
if ( !isDrone ) {
    curl = curl_easy_init();
    if ( curl ) {
        /* set the URL that is to receive our POST */
        curl_easy_setopt( curl, CURLOPT_URL,
            "http://www.satsis.com/roslee_LSP/data.php" );

        if ( updateServer ) {
            err = UpdateLocationServer( curl, own_bssid,
                      (int)own_xcoord,(int)own_ycoord,coeff1,
                      coeff2,coeff3 );
            if ( err < 0 ) {
                if ( isLog ) {
                    syslog( LOG_INFO, "Server update failed for
                        AP %s",own_bssid);
                }
            } else {
                if ( isLog ) {
                    syslog( LOG_INFO, "Server update
                        successfull for AP %s",own_bssid);
                }
            }
        } else {
            if ( isLog ) {
                syslog( LOG_INFO, "Propagation channel is
                  static. Server not updated for AP %s",
                  own_bssid );
            }
        }

        if ( droneNode != NULL ) {
            while ( droneNode != NULL ) {
                err = UpdateLocationServer( curl,
                          droneNode->bssid, droneNode->coordx,
                          droneNode->coordy, droneNode->data1,
```

```
                                droneNode->data2, droneNode->data3 );
                        if ( err < 0 ) {
                            if ( isLog ) {
                                syslog( LOG_INFO, "Server update failed
                                    for drone AP %s",
                                    droneNode->bssid);
                            }
                        } else {
                            if ( isLog ) {
                                syslog( LOG_INFO, "Server update
                                    successful for drone AP %s",
                                    droneNode->bssid);
                            }
                        }

                        dCurrPtr = droneNode;
                        droneNode = droneNode->next;
                        free( dCurrPtr );
                    }
                    droneNode = NULL;
                } else {
                    if ( isLog ) {
                        syslog( LOG_INFO, "Nearby remote AP drone(s)
                            not found" );
                    }
                }

                /* always cleanup */
                curl_easy_cleanup(curl);
            } else {
                if ( isLog ) {
                    syslog( LOG_ERR, "Unable to communicate with
                        location server");
                }
                free( ap_ssid );
                free( own_bssid );
                free( new_bssid );
                free( beacon );
                FreeWifiScan( &droneNode );
                nl_socket_free( sock );
                exit( EXIT_FAILURE );
            }
        }

goodnight:
        /* debug device memory */
        if ( isLog ) {
            if ( scanNode == NULL ) {
                syslog( LOG_INFO, "Scan result memory empty" );
            } else {
                syslog( LOG_INFO, "Scan result memory not empty" );
            }

            if ( modelNode == NULL ) {
```

```
                syslog( LOG_INFO, "Channel meas memory empty" );
            } else {
                syslog( LOG_INFO, "Channel meas memory not empty" );
            }

            if ( mMarkPtr == NULL ) {
                syslog( LOG_INFO, "Channel meas mark memory empty" );
            } else {
                syslog(LOG_INFO,"Channel meas mark memory not empty" );
            }

            if ( droneNode == NULL ) {
                syslog( LOG_INFO, "Drone data memory empty" );
            } else {
                syslog( LOG_INFO, "Drone data memory not empty" );
            }

            if ( dMarkPtr == NULL ) {
                syslog( LOG_INFO, "Drone data mark memory empty" );
            } else {
                syslog( LOG_INFO, "Drone data mark memory not empty" );
            }
        }

        // Free netlink socket
        nl_socket_free( sock );

        // go to sleep
        syslog( LOG_INFO, "%s goes to sleep for %d minutes",
            DAEMON_NAME, sleep_in_minute);
        sleep( sleep_in_minute*60 );
    }

    // if daemon is stopped, free all memories
    free( own_bssid );
    free( new_bssid );
    free( ap_ssid );
    free( beacon );
    free( tail );
    FreeWifiScan( &scanNode );
    FreeWifiScan( &modelNode );
    FreeWifiScan( &droneNode );
    if ( isLog ) {
        syslog(LOG_INFO, "Terminating daemon");
        closelog();
    }
    exit( EXIT_SUCCESS );
}
```

B. Matlab Script of the Incremental RF Fingerprint Algorithm

```
% This function implements the iterative fingerprint search algorithm
using dynamically generated radio map

function [location, numofap, numofpts] =
dynamicradiomapiterativesearch(fpCoord,apCoord,apPathlossModel,mobRss,
NMeas, NSize)

% Sort online fingerprint in descending order
[mobRssSort, mobRssIndex] = sort(mobRss,2,'descend');

% Iterates the unknown fingerprint, computes the RSS of the AP and
% select only those points that matches the unknown value
fpCoordIter = fpCoord;
for iterCnt=1:length(mobRssSort)
    fpCoordIterIdx = zeros(size(fpCoordIter,1),1);
    for fpCnt=1:size(fpCoordIter,1)
        fpDist = sqrt(sum(abs(fpCoordIter(fpCnt,:)-
apCoord(mobRssIndex(iterCnt),:)).^2,2));
        fpDist(fpDist<1) = 1;
        fpDistLow = fpDist-NSize;
        fpDistLow(fpDistLow < 1) = 1;
        fpDistHigh = fpDist+NSize;
        apRssLow =
mean(repmat(apPathlossModel(mobRssIndex(iterCnt),1),[length(fpDist)
NMeas]) -
repmat(10.*apPathlossModel(mobRssIndex(iterCnt),2).*log10(fpDistHigh),[
1 NMeas]) +
normrnd(0,apPathlossModel(mobRssIndex(iterCnt),3),length(fpDist),NMeas)
,2);
        apRssHigh =
mean(repmat(apPathlossModel(mobRssIndex(iterCnt),1),[length(fpDist)
NMeas]) -
repmat(10.*apPathlossModel(mobRssIndex(iterCnt),2).*log10(fpDistLow),[1
NMeas]) +
normrnd(0,apPathlossModel(mobRssIndex(iterCnt),3),length(fpDist),NMeas)
,2);
        if((mobRssSort(iterCnt)>=apRssLow) &&
(mobRssSort(iterCnt)<=apRssHigh))
            fpCoordIterIdx(fpCnt) = 1;
        end
    end
    fpCoordIterPrev = fpCoordIter;
    fpCoordIter = fpCoordIter(fpCoordIterIdx==1,:);
    if (sum(fpCoordIterIdx)==0 || sum(fpCoordIterIdx)==1)
        if (sum(fpCoordIterIdx)==0)
            fpCoordIter = fpCoordIterPrev;
        end
        break;
    end
end

% Solve for location by averaging remaining locations
```

```
if(size(fpCoordIter,1) == 1)
    location = fpCoordIter;
else
    location = mean(fpCoordIter);
end
numofap = iterCnt;
numofpts = size(fpCoordIter,1);
```