University of Alberta

Location-based Routing and Indoor Location Estimation in Mobile Ad Hoc Networks

by

Israat Tanzeena Haque

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

Spring 2011

Edmonton, Alberta

Examining committee:

Dr. Pawel Gburzynski, Dept. of Computing Science, University of Alberta

Dr. Ioanis Nikolaidis, Dept. of Computing Science, University of Alberta

Dr. Janelle Harms, Dept. of Computing Science, University of Alberta

Dr. Piotr Rudnicki, Dept. of Computing Science, University of Alberta

Dr. H. Vicky Zhao, Dept. of Electrical and Computer Engineering, University of Alberta

Dr. Michel Barbeau, Computer Science, Carleton University

Abstract

Location-based Routing and Indoor Location Estimation in Mobile Ad Hoc Networks

Israat Tanzeena Haque

In Mobile Ad Hoc NETworks (MANETs) autonomous nodes act both as traffic originators and forwarders to form a multi-hop network. Out-of-range nodes are reachable through a process called routing, which is a challenging task due to the constraints of bandwidth and battery power. Stateless location-based routing schemes have been proposed to avoid complex route discovery and maintenance, whereby nodes make routing decisions based solely on the knowledge of their location, the location of their neighbors, and the location of the destination. Natural routing schemes based on these prerequisites suffer from problems like local maxima or loops. We mitigate those problems by proposing randomized routing algorithms, which outperform others in terms of the packet delivery ratio and throughput.

The prerequisite for location-based routing is knowing the location of a node. Location information is more widely useful anyway for location-aware applications like security, health care, robotics, navigation etc. Locating a node indoors remains a challenging problem due to the unavailability of GPS signals under the roof. For this goal we choose the RSS (Received Signal Strength) as the relevant attribute of the signal due to its minimal requirements on the RF technology of the requisite modules. Then profiling based localization is considered that does not rely on any channel model (range-based) or the connectivity information (range-free), but rather exploits the context of a node to infer that information into the estimation.

We propose a RSS profiling based indoor localization system, dubbed LEMON, based on low-cost low-power wireless devices that offers better accuracy than other RSS-based schemes. We then propose a simple RSS scaling trick to further improve the accuracy of LEMON. Furthermore, we study the effect of the node orientation, the number and the arrangement of the infrastructure nodes and the profiled samples, leading us to further insights about what can be effective node placement and profiling. We also consider alternate formulations of the localization problem, as a Bayesian network model as well as formulated in a combinatorial fashion. Then performance of different localization methods is compared and again LEMON ensures better accuracy. An effective room localization algorithm is developed, and both single and multiple channels are used to test its performance. Furthermore, a set of two-step localization algorithms is designed to make the LEMON robust in the presence of noisy RSS and faulty device behavior.

Acknowledgments

It is a long journey to accomplish the research goal in a PhD program. Students need strong motivation and dedication towards the achievement of this goal. It is very common that they become frustrated along the way. Here comes the role of the supervisors, who mentor graduate students not only to guide them in research but also to provide mental support to recover from all sorts of frustrations and difficulties. Their support, encouragement, and guidance make it possible for a graduate student to successfully reach the goal. The scenario is not exceptional in my case and my supervisors Dr. Ioanis Nikolaidis and Dr. Pawel Gburzynski, University of Alberta, were always beside me during the long journey through my PhD program. I would like to express my gratitude to my supervisors for their continuous concern about my work. Their invaluable suggestions and guidance helped me to formulate the challenging problems addressed in this thesis and constructively work towards their solution. Indeed, their support and encouragement helped me to recover from a difficult situation after coming back from a maternity leave and working while having a young child. Also, I would like to thank Dr. Janelle Harms, Dr. Martha Steenstrup, Dr. Mike MacGregor, and Dr. Ehab Elmallah from Communication Networks group, Department of Computing Science, for their valuable suggestions. Also thanks go to Dr. Mario Nascimento, Database Systems research group from the same department, for his advice.

Secondly, I would like to thank my family for their encouragement and support. My father, Azizul Haque, was always there to boost me up from any difficult situations. One

person, without whose support and sacrifice I could not even have thought about embarking on this work is my husband, Wahedunnabi. The tiredness of a long-day work would disappear in a second after hugging my little son, Sudeep Nabi. I could start another long day full of energy after enjoying a big smile from him. This dissertation is thus dedicated to following people.

1. My son, Sudeep Nabi, whose presence is the secret of my energy.
2. My husband, Wahedunnabi, for his love, encouragement, and support to make my professional life successful.
3. My mother, Jasmina Begum, who is always there, especially in tough time of my life.
4. My father, Azizul Haque, for his encouragement.

Also, I would like to thank my colleagues Yuxi Li, Bauchun Bai, Abdullah-al Mahmood, Baljeet Malhotra, Nicholas Boers, Chen Liu, Benyamin Shimony, and Ryan Vogt, Computing Science, University of Alberta, for all the stimulating discussions in Communication Networks group. I want to thank my friends Rocio Sotomayer, Georgia State University and Dongwook Cho, Concordia University for kindly sharing with me their wisdom. A special thanks go to my friend Sheehan Khan, Computing Science, University of Alberta, for the stimulating discussions about both the wireless networks and machine learning. And last, but not the least, thanks go to all my friends who made me feel that I am not alone in this journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The infrastructureless wireless communication paradigm has gained in popularity in the past decade, primarily because of its deployment flexibility and the range of applications it can support. There are many niche applications of such wireless communications, e.g., disaster recovery, battlefield communications, social networking interactions at gatherings etc. and these are classified under the umbrella of Mobile Ad hoc NETworks (MANETs). A MANET consists of wireless nodes that can communicate without any fixed infrastructure or central control. Thus MANETs are a perfect choice in situations where infrastructure is not present or costly to deploy. For instance, an existing infrastructure can be damaged after a disaster like earthquake, tsunami, terrorist attack, etc. Rescue teams can communicate using MANET that can be deployed quickly without waiting to reestablish the destroyed infrastructure.

The devices in MANETs are equipped with omnidirectional (broadcast), directional (point-to-point), steerable, or combination of these types of antennas [42]. The radio range of the nodes defines the neighborhood where devices can use simple broadcast to establish communication with their neighbors. However, it is not possible to have very large radio range due to interference and energy consumption. Thus, the nodes have a moderate range and transmission power to form a multi-hop network, which changes in time because of

the node mobility. Owing to the limited range of communication over a radio channel, nodes located farther than that range have to rely on intermediate nodes acting as routers. Finally, the mobile and infrastructureless nature of the nodes imposes resource constraints (bandwidth, battery power), which make the routing problem in such networks challenging.

1.1 Routing in MANETs

The standard taxonomy of routing protocols for MANETs identifies *proactive* protocols, i.e., ones that try to maintain up-to-date routing information at every node in anticipation of demand [11, 44, 54], and *reactive* ones, which collect the necessary routing information only when it becomes explicitly needed to sustain an actual session [22, 30, 39, 51, 53, 67]. With few exceptions (e.g., [55]), routing schemes assume point-to-point communication, whereby each node forwarding the packet on its way to the destination sends it to a *specific* neighbor. If the identity and/or location of the neighboring nodes is unknown or uncertain, flooding is used as a means of route discovery.

One of the fundamental problems of routing, not necessarily wireless routing, is the scalability of the requisite knowledge, including its acquisition and storage, to the network size. This problem is particularly significant in wireless networks because 1) the acquisition component scales poorly due to the inherent bandwidth limitations of the wireless environment, 2) many applications of wireless networks (notably, wireless sensor networks—WSN) are based on nodes with limited resources, and this limitation affects both acquisition and storage. To forego the cost of discovering and maintaining detailed routing information at the nodes, a family of *memoryless or stateless location-based* routing schemes [7, 16, 33, 35] has been proposed. In these protocols, nodes rely solely on the knowledge of their location (which can be established, e.g., via GPS [15]), the location of their neighbors, and the location of the destination. We can also distinguish between deterministic

and randomized solutions to the routing problem. A routing algorithm is said to be *deterministic* if the routing rules never rely on a "coin toss" to decide a packet's fate, i.e., every decision is arrived at via a deterministic set of rules that produce the same output under identical inputs. Routing algorithms that are not deterministic are called *randomized*.

In this work, we consider location-based routing protocols, which might be a good choice for a network with large number of nodes. Two classic location-based routing protocols are *Greedy* [16] and *Compass* [35]. The former one selects the next neighbor on its way to the destination such that the remaining distance is minimized at every step. The next neighbor in Compass routing is the one that minimizes the direction (angle) towards the destination. These two simple protocols are effective in a network with dense collection of nodes; however, in a sparse network Greedy suffers from *local maximum* problem (if a node does not have a neighbor closer to the destination than itself) and Compass may face routing loops (when a packet is trapped between two neighboring nodes due to the next neighbor selection criteria) [21]. A network with low load may use deterministic approach as it is unlikely to create bottleneck in some nodes. At high loads, some nodes (hot spots) may become congested and start dropping packets, thus limiting the throughput. Randomization comes to the scene as by definition it may follow different routing paths even for the same pair of source-destination. The packet may traverse a suboptimal route, but the protocol has the potentiality to avoid congested nodes. In addition, randomized protocols could be an attractive choice to avoid routing loops and local maxima.

We propose location-based randomized routing protocols *Directional Location-based Randomized (DLR)* and *Forward with Random selection out of Two (FRT)* [23, 24] in ad hoc networks. These protocols could avoid local maxima and loops and improve the packet delivery ratio compared to the state-of-the-art. Indeed they perform impressively in various load conditions (low to high and uniform to biased traffic) to offer high throughput.

1.2 Indoor Localization

The prerequisite for location-based routing is knowing the location of a node. Indeed it is an interesting and challenging problem in many areas of practical applications, including security, health care, behavioral pattern recognition, robotics etc. While the problem is well addressed by GPS [9] outdoors, locating a device indoors still remains a challenging problem due to the extremely poor characteristics of GPS signals under the roof. Formally an indoor *localization* problem can be defined as a procedure of determining the Cartesian coordinates of a wireless device within some monitored area. The problem can be tackled via several different technological approaches (e.g., infrared or acoustic sensing [17, 31], as well as various pressure/vibration sensors [75]), but the generic RF-based approach has the promise of simplicity, ubiquity, low cost, and unobtrusiveness, if implemented properly. Hence, in this study we have selected RF-based techniques for indoor localization.

Once we decide to follow the RF class of techniques, we have the options to either rely on some existing wireless infrastructure (most prominently, WiFi Access Points (APs)) or use special wireless nodes dedicated to the specific task of localization. Within the framework of the first option, the coordinates of the APs are assumed to be known and act as the basis by which the location of other nodes is gauged. Many indoor areas (where the localization problem may be of relevance) are already outfitted with Local Area Networks (LANs) of APs providing wireless connectivity to the Internet. It appears to be a natural extension of their role if, in addition to acting as APs, one piggybacks on them the secondary task of localization, if possible. A relevant, often unstated, property of such systems is that the placement of the APs is constrained by the tradeoffs of WiFi connectivity (quality of service) against the cost (the number of APs, transmission power, collisions, RF pollution), which need not result in a good design for a localization network. Indeed, our experience indicates that the placement of the static nodes in such a network is a significant factor, pretty much unrelated to the concept of wireless coverage, as understood by the clients of a typical WiFi system. In other words, an AP deployment that can support localization

should be designed with the purpose of aiding localization in mind.

Based on the attributes of the wireless signal used for localization, one can identify techniques based on measuring RSS (the Received Signal Strength), TOA (the Time Of Arrival), or AOA (the Angle Of Arrival) of the signal between the transmitter and receiver (details will be given in Chapter 3). The first of those categories of techniques is most attractive from the practical point of view, as it poses minimalistic requirements on the RF technology of the requisite modules, which translates into low cost and off-the-shelf availability.

Within its realm, RSS-based localization can be carried out using methods that are *range-based*, *range-free*, or based on *fingerprinting (profiling)*. Schemes in the first category use channel models to transform the received signal strength into a distance estimate, and then apply lateration [36] to estimate the location. Due to the complicated nature of signal propagation indoors, range-based schemes tend to produce large errors, even when driven by sophisticated channel models, e.g., ones taking into account the presence of walls [3]. The range-free class of solutions exploits the connectivity (direct neighborhood) information among the wireless nodes distributed in the monitored area. For example, the infrastructure nodes (anchors) may broadcast their location along with the average hop distance to other nodes into the network, with the tracked nodes using that information to infer their distances to the anchors. Lateration may then be applied to estimate the precise location of the tracked nodes. Such a scheme might be an attractive choice in a network with a large number of nodes (preferably of very limited range); however, its performance is highly sensitive to the network structure [46]. This is because the geographical extent of a single hop may vary depending on the distribution of nodes and also on the capricious propagation characteristics of the indoor environment. Consequently, the quality of approximation of distance via hops may vary quite drastically and, quite likely, more so in those environments that are also difficult for RSS-based schemes [46]. The fact that some (nominally) range-free schemes try to mitigate those problems by resorting to range-based

adjustments (like assessing the hop range from a channel propagation model) is not reassuring, because of the fundamental shortcomings of such models, which often go to great lengths distinguishing between the Line-Of-Sight (LOS) and Non-Line-Of-Sight (NLOS) signal arrival.

The unifying idea behind the profiling-based methods is the constatation that the only hope at fighting the inherent and fundamental flaws of blanket propagation models lies in embracing those flaws as features. Thus, as any attempts to derive a meaningful notion of distance directly from RSS are bound to fail in an unknown indoor environment, such an environment has to be studied (profiled) in order to create a customized model for accurate location estimation in the particular area. The perceived values of RSS are viewed as some property of the received radio signal whose association with the actual distance from the sender need not be explicit or straightforward. Instead, those values contribute to a *radio map* of the monitored area. The procedure of localization becomes a two-stage process with *profiling* preceding the proper "operation," i.e., actual *estimation*. Profiling means collecting signal strength samples, through infrastructure nodes, from known locations within the monitored area. During the localization stage, the tracked nodes send signals whose RSS footprints are matched against the set of profiled values stored in a database of the central server. It has been demonstrated that this kind of approach offers consistently a better location estimation accuracy than other categories of schemes [3, 26, 74].

Examples of profiling-based schemes include RADAR [3] and LANDMARC [45] (details will be provided in Chapter 3). The former relies on desktop computers as infrastructure nodes and laptop as the tracked (sending) device, whereas the latter uses RFID (Radio Frequency IDentification) readers and tags as infrastructure nodes and sending devices, respectively. However, both approaches try to keep the number of nodes small due to cost and hence suffer from poor estimation accuracy. We propose a profiling-based localization scheme, dubbed *LEMON (Location Estimation by Mining Oversampled Neighborhoods)* [26], using low-cost low-power devices. LEMON has been implemented on a collection

of inexpensive (microcontroller-based) nodes amounting to a massive ad-hoc wireless sensor network (WSN). Consequently, it can deal with a large number of infrastructure nodes whose transmission/reception areas can largely overlap. The infrastructure nodes (called *pegs*) can be inconspicuously placed with practically any requisite density and practically anywhere, including furniture, outlets, or even inside walls to generate an accurate radio map. Given a particular distribution of pegs, the profiling stage consists in collecting sample readings from prospective tracked nodes (called *tags*) and storing them in a database. Actual estimation of a tag's location is then done by reporting the RSS readings of the tag's signal from the nearby pegs and mining the database for "close" samples (in sample space). The coordinates of these samples contributing to the "best" matches are then averaged (in some weighted fashion) to produce the location estimate.

RSS measurements using micro-controller based devices were collected at various indoor spaces at the University of Alberta and were used as input to various localization algorithms. The localization results demonstrate that LEMON offers better accuracy compared to the state-of-the-art. In particular, we present a series of experimental results, whose purpose is to learn the effect of various parameters such as the number of nearest neighbors, the number and density of profile data, the number and placement of infrastructure nodes etc.. We also show that RSS-scaling can significantly improve the localization accuracy, where scaling could be done by giving high/low RSS values different weight when deriving the location estimates. Most of these initial experiments were carried out in a single room. We further have considered multiple rooms and investigated the performance of LEMON with and without partitioning the profile data into room level.

We further study and compare profiling-based LEMON with other methods including *lateration, Bayesian Networks, Maximum Likelihood Estimation (MLE), Gaussian Process (GP),* and *Support Vector Machine (SVM)*. In addition, we propose a new localization method dubbed *combinatorial localization*. LEMON, a much simpler localization approach outperforms the state-of-the-art or performs just as well as much more complicated

schemes. In addition to the above mentioned localization problem, i.e., computing the absolute Cartesian coordinates of an object, we also consider the room localization problem, i.e., identifying a room from which the query came from. For this purpose we choose K closest neighbors and count their room labels, the majority of the labels is reported as the room label of the query tag. This room localization is tested under multiple rooms and channels, where we obtain adequate performance in terms of identification accuracy. Multiple channels are also used for location estimation to improve the accuracy.

LEMON is used indoors where RSS is affected by significant multi-path propagation due to the presence of obstacles (walls, furniture etc.). As the first step of mitigating such effect of multi-path, we have used RSS profiling to obtain similar RSS reading from similar environment. We have seen that this idea helps to obtain good estimation accuracy. However, RSS is still contaminated by noise. The disregard by LEMON for any explicit representation of the propagation environment or of any of the noise and interference processes around the nodes suggest that there may be cases where LEMON will fail to produce acceptable results. Yet, as we will see, LEMON is robust even when completely artificial measurements are introduced on purpose, and certainly is more robust than a number of existing techniques. Even when the overall performance of all schemes deteriorates, LEMON maintains an edge over the rest. Finally, we introduce a set of two-step localization schemes along this path of achieving robustness. In *midrange* localization, we reconstruct the RSS range for each peg based on their *cross correlation coefficient (c)* and using profiled reference points. These new RSS values then find their way into the localization of tags. In *peg reliability* based localization we use midrange RSS values to compute c for each peg. We use it as the weighting factor during the RSS discrepancy measurement.

1.3 Thesis Contribution

In a nutshell, the contribution of this thesis could be summarized as follows:

- We propose location-based randomized routing to overcome the local maxima and routing loop problems and to improve the packet delivery ratio (Chapter 2 and [23]).
- Randomized routing is also proposed to avoid node-congestion (created by the deterministic routing schemes) and better distribute the load among nodes to obtain high throughput (Chapter 2 and [23, 24]).
- A simple and robust indoor localization system, dubbed LEMON, is proposed based on low-power low-cost wireless devices while offering good indoor localization accuracy. Its performance is evaluated, including its sensitivity to RSS reading adjustments such as scaling (Chapter 3 and [26]).
- The tradeoffs of infrastructure node and profiling point placement for profiling-based localization schemes is examined in depth (Chapter 3 and [25]).
- We examine how LEMON and other profiling-based schemes compare to well known estimation and classification techniques, e.g., Bayesian Network, as well as variations of the basic LEMON scheme (Chapter 4).
- Finally, we consider the imperfect nature of RSS readings and examine ways in which the robustness of RSS measurements, or individual pegs can be assessed and mitigated. We study how the mitigation schemes impact profile-based localization schemes, including LEMON (Chapter 5).

1.4 Thesis Organization

Chapter 2 starts with the routing issues and an overview of the routing schemes in MANETs. We then define our proposed randomized location-based routing schemes. A detailed performance analysis of the proposed solution is then presented and compared with the state-of-the-art. The performance of the deterministic and randomized routing schemes are then

analyzed under different traffic conditions and using various network topologies to assess the throughput of these schemes. Our proposed randomized solution again performs adequately under all the traffic conditions and could successfully avoid the hot spots, which is an issue for deterministic approaches [24].

Chapter 3 starts with the motivation behind choosing the indoor localization problem and a description of the hardware that was used in our experiments. Then we present the literature review of indoor localization. Our proposed localization scheme, LEMON, is then defined. A series of experiments is carried out in various locations of the University of Alberta campus to test the performance of LEMON in different environments. We then present the performance results of LEMON and compare and contrast them to other profiling-based localization schemes. We examine the RSS scaling effect, the effect of the node orientation, the number and arrangement of the infrastructure nodes and the profiled samples on the accuracy of LEMON [25]. Other tests are also performed such as the RSS discrepancy measurement, the averaging techniques, and the impact of different types of obstacles.

In Chapter 4 alternative formulations of the localization problem, including a Bayesian Network based model, and a combinatorial localization technique are proposed and evaluated. The performance comparison of different localization methods is performed using both unscaled and scaled RSS, and LEMON offers better accuracy compared to the others. A simple but effective room localization scheme is proposed and tested with different setups. Both single and multiple channels are used for this purpose. The latter option is also tested for localization and an improvement is observed.

Chapter 5 investigates the robustness issues of profiling-based localization, including LEMON. For this purpose we conduct further experiments and observe the impact of the RSS dimension expansion and the noisy RSS on the localization performance. LEMON is also quite capable of maintaining good performance in the presence of a node producing faulty RSS readings. We also observe that LEMON, as well as other localization schemes,

can benefit from relatively simply two-step schemes that restrict the use of RSS values that are deemed to be outliers or put less weight on measurements from pegs that are deemed unreliable.

In Chapter 6 we briefly discuss our contributions to the research on MANETs. We have discovered several interesting and challenging open problems during the research for this thesis, which are also described in the end of this final chapter.

Chapter 2

Location-based Routing in Mobile Ad Hoc Networks

2.1 Introduction

In general, the wireless environment is quite susceptible to interference and vulnerable to environmental changes. The propagation characteristics of wireless nodes are difficult to characterize because of their unpredictability and dependence on many difficult to pinpoint factors [6]. Here is the list of fundamental issues that must be fought by effective communication techniques in such an environment:

The hidden and exposed terminal problem: The hidden terminal problem [5] is illustrated in Figure 2.1(a), where node *A* is communicating with *B*. Node *C* is unaware of this ongoing communication as it is outside A's radio range, thus, *C* suspects the medium is free to use and sends message to its neighbor *B*, resulting in a collision. All the hidden nodes for *A* are located in $C-(A\cap C)$. The exposed terminal problem [5] (illustrated in Figure 2.1(b)) prevents nodes from transmitting packets in some situations where safe communication is possible. For example, *A* wants to communicate with *B* and *C* has packets for *D*. Upon hearing the communicating session between *A* and *B*, *C* will stay silent. Thus the hidden

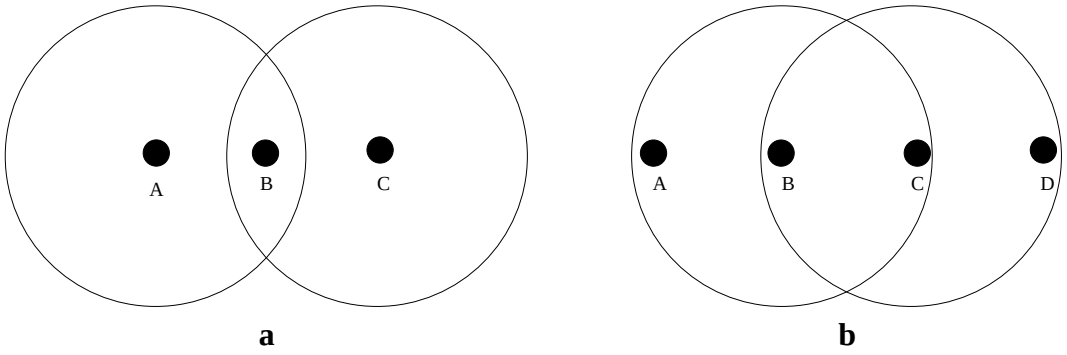


Figure 2.1: Hidden and exposed terminal phenomena.

terminal problems wastes the resources by generating more collisions whereas the exposed terminal problem lowers the throughput because of unused resources.

The absence of infrastructure: As there is no preexisting infrastructure or central control, the nodes in the ad hoc network need to serve both as the traffic originators and routers (to relay the packets for other nodes). This imposes extra burden to the nodes and makes the network management more challenging.

Dynamic network topologies: As the nodes move frequently and unpredictably, the multi-hop network topology changes often. The network partition, link-asymmetry, route changes, and packets drops come at a price [13, 42].

Constrained resources: Nodes operate within a limited resource budget (mostly battery power). Thus a clever mechanism is required for better throughput. As nodes serve both as routers and originators of packets, energy utilization at the nodes is crucial.

Heterogeneous nodes and links: Nodes usually have different hardware/software configuration with different capabilities. Indeed, nodes may be equipped with multiple transceivers with varying capabilities (transmission power, frequency). In turn, asymmetric links may appear in the network topology. All these issues, make routing a tedious job and require adaptation to changing conditions [13].

Scalability: It is an issue when considering a large number of nodes (e.g., in sensor networks). Routing, location management etc., become non trivial to manage in resource-constrained large networks.

2.1.1 Routing issues of MANETs

In the past few decades, a large number of routing protocols for MANETs have been proposed, having different objectives, i.e., addressing different performance metrics to be optimized. We need a general set of metrics - to comprehensively and meaningfully compare diverse protocols. These metrics are independent of any routing protocols to judge a protocol in MANETs. Corson *et al.* in [42] define both qualitative and quantitative metrics for MANETs.

The qualitative properties include *distributed operation*, *loop-freedom*, *reactive and proactive operation*, *adaptation to asymmetric link*, *support for constrained resources*, *scalability*, and *robustness* [21, 42]. Routing decisions are made individually at each node, instead of relying on any central control. It is not desirable to allow packets to survive in the network for a long time because of unexpected routing loops. This may lead to wastage of bandwidth and degradation of throughput. Thus, protocol design should take account of loops without relying on TTL (Time To Live). Proactive protocols maintain up-to-date routing information at every node ahead of demand. Thus they suffer from high control overhead and usage of scarce resources (energy and bandwidth). Reactive or on-demand operation helps to minimize these overheads with the cost of route discovery latency. Most of proposed routing protocols assume symmetric links between nodes. As we mentioned before, link-asymmetry may arise due to the dynamic nature of the networks. Thus an elegant routing scheme would consider such asymmetric links. In ad hoc networks of sensors, nodes have limited battery power, thus to prolong the network lifetime, it is desirable that the routing protocols consider energy utilization at the nodes as well as the communication

energy (the smaller the distance the lower the necessary energy dissipation). Indeed, scalability is an issue in such large networks as routing protocols are not supposed to degrade in their performance with increasing number of nodes. Finally, robust routing is needed to cope with the dynamic behavior of the ad hoc networks.

The quantitative metrics include throughput, end-to-end delay, route acquisition time (end-to-end delay for the route discovery in case of on-demand routing), and percentage of out-of-order delivery [42].

2.2 The Problem Definition and Network Model

Formally a MANET is described as a set V of N nodes placed in 2- or 3-dimensional Euclidean space. In location–based schemes, it is assumed that each node is aware of its location, expressed as Cartesian coordinates (x,y) or (x,y,z). We assume that the transmission range of all nodes is the same and equal to R. Two nodes can communicate with each other if and only if their Euclidean distance is at most R. The ability to communicate is represented by an edge between the corresponding nodes. The resulting graph, *G=(V,E)*, is the topology of the network. G varies over time due to the presence or absence of the links among nodes. Given G and a pair of nodes (i, j), $i, j \in V$, the problem of routing is to find a path from i to j while minimizing some objective functions.

2.2.1 Channel model

In simulations we assume the shadowing propagation model [56] with the path loss at distance d being

$$PL(d)[dB] = PL(d_0) + 10\alpha \log(\frac{d}{d_0}) + X_\sigma$$

where $PL(d_0)$ is the path loss at the reference distance d_0, α is the path loss exponent, and X_σ is a zero-mean Gaussian distributed random variable with standard deviation σ. The

antenna gain is included in $PL(d)[dB]$,

$$PL(d_0) = 20\log(\frac{4\pi d_0}{\lambda})$$

where λ is the wavelength. The received power is expressed as

$$P_r[dBm] = P_t[dBm] - PL(d)[dB]$$

where P_t is the transmission power. If the received power is less than the threshold power P_{th}, the signal is not correctly received [66]. In our simulations, α and σ are 3 and 8, respectively. The reference distance is 1m, the transmit power is 25dBm, and the threshold is -95dBm and λ corresponds to 2.4 GHz. Note that these parameters are chosen based on simulation results such that network is connected and nodes communicate with each other with moderate power and range. Also, these parameters are consistant with the standard values used in the related work.

2.2.2 Energy model

The energy model is the same as in [58], which coincides with the one used in ns-2 [66]. A node loses $P_{xmit} \times t_{transmit}$ amount of energy, where $t_{transmit}$ is the transmission time. Also, when receiving a packet, the energy loss is $P_{recv} \times t_{receive}$. Note that we need to update the remaining energy for the performance analysis of the network lifetime.

2.2.3 Antenna model

A protocol can use a smart switched-beam antenna [14] with multiple predefined directional beams. There are two modes of operation: omni-directional and directional, with one mode being active at a time. The size of the main lobe in *Directional Location-based Randomized (DLR)* [23] is $\pi/3$; the side lobes (deemed insignificant) are approximated into

a (single) sphere. The *Probabilistic Geographic Routing (PGR)* [58] protocol in particular also starts with this main lobe, but increases it up to π.

2.2.4 Mobility model

When mobility is simulated, we use the *random waypoint* model, whereby each node chooses a uniformly distributed random location from a rectangular area and moves there at a constant speed selected at random from $[0, V_{max}]$. After reaching the new location, the node stays there for a *pause time.* Then, the node repeats the same process until the end of the simulation run. In our simulation experiment, V_{max} is 10m/s and the pause time is constant 30 sec. These parameters are again chosen based on the simulation results, which are consistant with the related work.

2.2.5 Traffic model

The *uniform traffic* model adopted in our simulations makes sure that the destination of a packet is at least two hops away from the source. The proper way to implement such a model is to generate the destination first (uniformly from all nodes), and then select a random source, uniformly from among all nodes excepting the destination as well as its neighbors. In the case of *biased traffic*, we assumed that the endpoints are located on the edges (specifically the bottom and upper edge of the grid), while the interior nodes act exclusively as routers.

2.2.6 Performance measures

The following performance measures are collected during the experiment.

- **Packet delivery ratio**: the ratio of the total number of packets successfully received by the destination to the total number of packets originated at the source.

- **Path length**: the number of hops taken by a packet to reach the destination in case of a successful packet delivery. The path length is an indicator of delay performance.
- **Network lifetime**: the average number of successful routing tasks before the first node in the network has lost all its energy.
- **Throughput**: the maximum number of bits per unit of time that were successfully received. In all cases, we drove the networks to saturation to see how the schemes perform under extreme loads.

2.3 The State-of-the-art in MANETs Routing

In this section we outline the routing approaches that are considered the state-of-the-art in MANETs.

2.3.1 Proactive routing protocols

In proactive routing protocols, nodes maintain up-to-date routing information ahead of demand. This strategy helps such protocols to follow optimal path. *Destination-Sequenced Distance-Vector (DSDV)* [54] and *Clusterhead Gateway Switched Routing (CGSR)* [12] are examples of proactive protocols.

Destination-Sequenced Distance-Vector (DSDV)

DSDV is one of the early proposed protocols in MANET. It is based on the traditional distance vector (Distributed Bellman-Ford) routing. In distance vector routing, each node maintains a routing table with the best path (say in number of hops) to each destination along with the next-hop neighbor on that path. Tables are updated by periodic exchanges among the neighbors. However, it suffers from loop and count-to-infinity problems in case of link breakage. Consider a network of 4 nodes A, B, C, and D shown in Figure 2.2.

Suppose D goes down, C detects it and sends an update to its neighbor. After receiving this update B realizes that it has an entry for D (corresponding to the now defunct route through C) and passes it to the neighbors. Consequently, C changes its entry for D with the new metric (increased by 2 hops). This process keeps going up to infinity. In DSDV loops are prevented by using a sequence number for each routing entry. Nodes broadcast routing updates at regular intervals or after any significant change. Each entry in the routing table contains the destination, the next-hop neighbor to reach it, the required number of hops, the sequence number generated by that destination, the install time (when the entry is made), and the stable-data. The install time is used to remove stale routes, i.e., the routes for which no update has been made in the last few update periods [54]. The last field (stable-data) is used to determine how stable the route is, i.e., no update is broadcast until the route is considered to be stable. This is useful to damp the fluctuations in the networks.

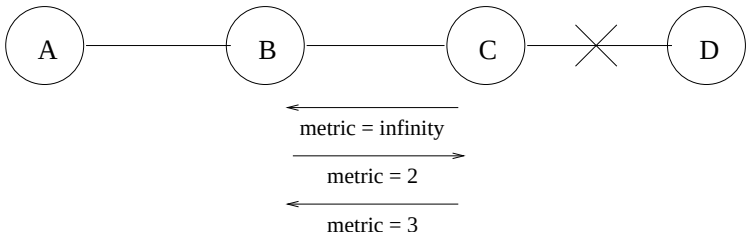


Figure 2.2: The presence of count-to-infinity in distance vector routing.

A node will update an entry, if it sees a ”recent” update, i.e., one with an increased sequence number (compared to the last one). In case of same sequence number, the smaller cost metric is used. As nodes move, links may go up and down dynamically. When a node detects that a link to one of its neighbors has gone down, it will generate (on behalf of the neighbor) an odd-numbered update (a link-down event) with the metric value of ∞. This is the only case where another node generates the sequence number instead of the node responsible for the metric. The nodes generate for themselves only even numbers. Link down events signal significant changes in the topology. Thus, there are two ways of

broadcasting the route update; one is *full dump* containing all routing information and the other one is *incremental* that includes only changed information after the last full dump. In the limited capacity wireless medium, it is desirable to keep the full dumps infrequent, especially in a low-mobility environment.

Consider Figure 2.2 again to see how the loop and count-to-infinity problem is eliminated in DSDV. As before, suppose that the link between C and D is broken, C detects it and generates an odd sequence number by incrementing D's sequence number by one and setting the metric to ∞. Now if B broadcasts a route entry for D with a lower metric, C will know it's a stale route as the sequence number is not recent. On the other hand, a broadcast from C may help other nodes to learn about the link breakage.

A node may receive a large metric first followed by the best metric for the same destination. This may happen when the number of nodes is large and the update frequency is low. This can also happen in a not-so-large network using diversified update frequency [54]. If care is not taken, then it could create unnecessary traffic in the network. Nodes need to determine when an entry for a destination becomes stable and ready to trigger an update. For this purpose *settling time* is used, which is the time difference between the first and best (least cost) received updates for a destination. The last stored settling time, t_l, is the most recent measurement. Nodes also keep calculating the *average settling time*, t_a for each destination. t_a can be measured as $t_a^{n+1} = \alpha t_l^n + (1-\alpha)t_a^n$. α is used to assign more weight to the recent measurement. After receiving the first update for a destination, a node waits for a period of $2 \times t_a$ before broadcasting it. This rule is not applicable for the case of a broken link, where immediate broadcast is required.

Clusterhead Gateway Switched Routing (CGSR)

In a hierarchical network architecture, nodes are grouped into clusters and each cluster is controlled by a special node called *clusterhead*. Nodes could belong to multiple clusters, and serve as *gateways* between the clusters. This hierarchical structure may help to achieve

better channel access, bandwidth allocation, and routing [12]. However, it may also lead to following suboptimal routes. Also, a clever clusterhead selection procedure is needed in MANETs. In CGSR, a clusterhead selection algorithm called LCC (Least Cluster Change) is proposed, where initially the lowest ID or highest node degree is used to select the clusterheads. Afterwards, instead of using the reselection procedure of clusterheads after a significant change in the topology, in LCC the clusterheads only change when two of them come into contact, or when a node moves out of contact of all other clusterheads. To route packets, nodes first send them to the corresponding clusterhead, the packets then travel from clusterhead to clusterhead through gateways to reach the destination clusterhead. Each node needs to maintain two tables, *cluster member table* and *routing table*. The first one records the clusterhead of each node (this table is periodically sent to the neighbors), and the second one keeps the next hop node to reach the destination clusterhead.

2.3.2 Reactive or on-demand routing protocols

These protocols differ from the proactive protocols in the way nodes maintain routes. Reactive protocols collect the necessary routing information only when it becomes explicitly needed to sustain an actual session. In general, routing is performed in two steps namely *route discovery* and *route maintenance*. *Dynamic Source Routing (DSR)* [30], *Ad hoc On-Demand Distance Vector (AODV)* [53], *Temporally Ordered Routing Algorithm (TORA)* [50, 51], *Tiny Ad hoc Routing Protocol (TARP)* [48], and *Associativity Based Routing (ABR)* [68] are the examples of on-demand routing protocols. In the following subsections we will briefly present three of them.

Ad hoc On-Demand Distance Vector (AODV)

AODV is a reactive version of the DSDV protocol, which minimizes the required number of broadcasts by creating on-demand routes instead of maintaining a complete list of routes as in DSDV. It uses the destination sequence number to maintain up-to-date routes and the DV

logic in calculating the best path. The route discovery process is initiated by a node when it is required to deliver packets to a destination for which it does not have a route in the cache. The control packets for this purpose are **route request (RREQ)** and **route reply (RREP)**. The former is of the form $< S_{add}, S_{seq-num}, broadcast_{id}, D_{add}, D_{seq-num}, HC >$. The first two attributes represent the source address and the sequence number, the next one along with the source address uniquely identify a RREQ, the following two attributes are for the destination, and HC is the hop-count. After receiving a RREQ message, an intermediate node may react by a) sending a unicast reply message to the source, if it is the destination or has a route entry for the destination in its cache with a destination sequence number greater or equal to the one in RREQ or b) dropping the RREQ, if it has already seen this request. Otherwise, intermediate nodes simply increment the hop-count and rebroadcast the RREQ.

The RREP message is of the form $< S, D, D_{seq-num}, HC, Lifetime >$. The destination sequence number is higher than that of the one in RREQ, and HC is initialized to 0 at D. The last attribute provides the expiration time of the route setup. The unicast RREP is sent back by D to S along the path of RREQ. An intermediate node after receiving a RREP, generates a reverse path pointer to the neighbor from which the RREP came. Also, it updates the $D_{seq-num}$ and the timeout information. The RREP is then sent back to the neighbor that propagated the RREQ, if this is the first time RREP is received or $D_{seq-num}$ is greater than the previous one. Otherwise, subsequent RREPs are dropped to suppress multiple paths unless bring in better hop counts for D. Thus, intermediate nodes learn about the routes to both S and D as a side effect of this route discovery process. Finally, S receives RREP which might not be optimal the first time around, but eventually it will receive the optimal path. The *route request expiration timer* is used to purge the reverse path entries from the nodes that are not on the path between S and D.

Nodes in AODV maintain route table entries for the set of active destinations. Those entries are of the form $< D, D_{seq-num}, Next_{hop}, HC, Active_{nei}, time-out >$. $Next_{hop}$ is the node to be used for the given destination D, HC is the number of hops required from S to D,

and time-out is used to purge stale routes. Also, a set of active neighbors is maintained for a particular *D*. The *active neighbors* of *S* for *D* are the ones that originated or relayed at least one packet for *D* in the most recent active-timeout-period [53]. The active neighbors are the ones to be notified if the route entry for *D* becomes invalid (note the difference with respect to the periodic broadcasts in DSDV). When a node detects an unreachable next-hop node for a given route, it informs the active neighbors with a RREP having a higher destination sequence-number and the hop-count of ∞. These active neighbors are then propagating the message to their upstream active neighbors until all the active source nodes are informed.

Dynamic Source Routing (DSR)

In addition to RREQ and RREP, DSR uses yet another control packet called RERR. It retains the two routing phases of AODV. The route discovery in DSR is similar to that in the AODV except that every time a node rebroadcasts a RREQ, it also includes its own address in this request packet to keep track of the entire route from source to destination. To prevent loops, an intermediate node drops the received RREQ if it is already in the route shown in the received RREQ. The RREP could be sent by the destination using the cached route (if it is available) for the source. In the case of bidirectional links the destination can also use the route available in the RREQ. If *D* does not have any cached route for *S*, it needs to initiate a RREQ for *S* and may piggyback the RREP in the RREQ packet. To prevent the loss of piggybacked data, an intermediate node, after replying a RREQ from its cache, must construct another new packet with the piggybacked data.

After receiving the RREP, the source node, caches it and sends data packets where the entire route is added in the header (hence the name source routing). As part of the route maintenance, nodes monitor for link breakage using either hop-by-hop or end-to-end acknowledgments. In either case, RERR is sent back to the source to force a proper action. The former method allows to detect a particular hop in trouble and the latter may give an impression that the last hop along the path to the destination is down [53]. The error

handling scheme is further enhanced by a) restricting the rate of route discovery for the same destination, as useless RREQ packets may occupy the medium in case of partitioned networks, b) allowing nodes to operate in promiscuous mode to overhear the error message to update their cache, if the broken link is present in any of their cached routes, and c) retransmitting the error message up to the point of error to inform all the nodes along this path about the link breakage (this may happen if intermediate nodes need to discover route to the source to send back the RERR).

DSR further reduces the control overhead by using cached routes, i.e., nodes may consult their cache to answer a RREQ to make the discovery faster. If an intermediate node needs to forward a packet for which the next-hop link is broken, it may use its cache to use an alternative route to the same destination instead of dropping the packet. Also, nodes may stay in promiscuous mode to overhear all the packets originating at the neighbors. Multiple intermediate nodes may send RREP for the same request at the same time and collisions may occur. Thus a node waits for a period of $d = H \times (h - 1 + r)$, where H is a constant, h is the length of the route for this RREP in hops, and r is a random number between 0 and 1. Node can also stay in promiscuous mode during this waiting period and stop sending the RREP if a RREP with a shorter path has been received during this period. A source node may also add hop limit to RREQ to reduce the overhead. For instance, initially a RREQ with the hop limit of 0 may be used to check if any neighbor has a cached route for the requested destination. Otherwise, the node may use another non-zero value to limit the propagation of RREQ. Finally, formation of loops can be stopped by allowing an intermediate node to respond to a RREQ using its cache if it is located at the beginning of the cached route and in the end of the route in RREQ. The promiscuous mode also helps a node determine if any nodes could be reached directly instead of going through an intermediate neighbor. This helps to reduce the number of hops to the destination.

Tiny Ad hoc Routing Protocol (TARP)

TARP is an on-demand routing scheme intended for networks with low-cost and constrained-resource devices [48]. Controlled flooding is used to force the packets to follow routes with a limited number of hops. The objective is accomplished through three rules namely *Duplicate Discard (DD)*, *Sub-optimal Path Discard (SPD)*, and *Load Balancing (LB)*. The primary goal of DD is to discard duplicate packets by checking the cached packet signature. The performance of TARP could be further enhanced by a) discarding the cached signatures quicker as the packets approach the destination and b) setting the signature expiration timer as $T_r = F_c \times (t_{avg} \times (r - h))$. F_c is the flooding control constant, t_{avg} is the average transmission time (the time between a packet is queued and first received by a neighbor) , r and h are the number of hops allowed to traverse for a packet and already traversed, respectively. The SPD rule is implemented by using a tuple of $< S, D, h_{DK}, h_{SK}, C_{DS}, C_{SD} >$, where S, D, K are the source, destination, and an intermediate node. h_{DK} and h_{SK} are the number of hops from D to K and S to K for the last-seen packet at K, respectively. Finally, C_{DS} and C_{SD} are the *discard counters* for the two directions. These parameters are set by K as $C_{DS} = \bar{m} \times [(h_{SK} + h_{DK}) - \bar{h}]$ and $C_{SD} = \bar{m} \times [(h_{SK} + h_{DK}) - \bar{h}]$, where $\bar{m}$ and $\bar{h}$ are the mobility factor and number of hops on the reverse path. Thus K checks the packet header traveling through it and discard it if the packet is going to traverse a path shorter than what K is expecting based on the above calculations. If both C_{DS} and C_{SD} are positive, the route through K is suboptimal and better to avoid it. Finally, the LB rule is used to allow packets to traverse less congested nodes and might be suboptimal route to the destination. TARP does not have any route discovery phase as is DSR and AODV, rather it initiates flooding and control it using three rules to allow packets to traverse reasonably good paths.

2.3.3 Location-based or geographic routing protocols

All the protocols described in the previous sections are routing-table based solutions, where these tables need to be updated to cope with the topological changes. The communication

overhead is quadratic in the network size. This makes the proactive and reactive routing protocols progressively more complex and less attractive as the number of nodes in the network grows. Controlled flooding, as in TARP, can simplify the routing and bring about a potentially significant reduction in the routing cost, but the control overhead is still there. Location-based or geographic routing protocols are proposed to eliminate the need of flooding. In these routing schemes, the next-hop node along the route to the destination is chosen solely based on the, possibly approximate, knowledge of the geographic coordinates of the source, its neighbors, and the destination. They do not require route establishment and maintenance, thus they can efficiently utilize the scarce resources in the wireless environment. Numerous location-based routing schemes have been proposed in the last decades. In the following discussion, we will present the most significant ones. We classify these protocols roughly as a) *limited flooding-based routing* [4, 34], b) *guaranteed-delivery routing* [7, 33], and c) *progress-based routing* [16, 35]. A detailed description of location-based routing protocols can be found in [21, 43, 62].

Limited flooding-based routing

Limited flooding is a class of location-based routing schemes where nodes forward the packets to all neighbors that are located in the direction of the destination. *Location-Aided Routing (LAR)* [34] and DREAM [4] are two routing algorithms that apply this principle. LAR uses directed flooding only for route discovery, while DREAM applies a restricted flooding for packet delivery.

Location-Aided Routing (LAR) reduces control overhead due to route discovery (based on flooding) through a controlled flooding by using location information of the nodes. This is accomplished by defining two geographical regions called *expected zone* and *request zone*. The expected zone is the region where the destination node d is expected to be located. The zone is defined as a circular region centered at the destination's last known location at time t_0 with the radius of $v(t_1 - t_0)$, where v is the velocity of the destination

and t_1 is the current time. Note that if no previous location information is available, the entire network is the expected zone of the destination. The request zone is the region that contains the source and the expected zone and some other nodes from the network. The idea is to confine the route request packet to a relevant subset of the network as to reduce the flooding overhead. The source may increase the size of the request zone if the route discovery stage fails. However, a large request zone also means a high overhead.

The source node can define the request zone in two different ways. In the first scheme the request zone is the smallest rectangle containing the source and the expected zone. The source calculates the four corners of the request zone for a route discovery and appends this information to the route request packet. An intermediate node discards the request packet if it stays outside the request zone. Destinations may use the same approach as in AODV to reply to route requests or they perform the same (but reversed) procedure that the source used for route discovery. In the reply packet the destination appends its current location, time, and the speed. In the second variant of the scheme, the source calculates the distance between itself and the destination, $dist(s,d)$, and includes it in the RREQ packet along with the destination's location. An intermediate node forwards the RREQ if $dist(s,d) \geq dist(i,d)$, where $dist(i,d)$ is the distance between the intermediate node and the destination.

Unlike LAR, DREAM is a proactive approach that uses location information to forward packets. The location information is dispersed across the network based on two criteria: 1) *distance effect* and 2) *mobility rate* to limit the scope of the propagation. The former implies that the closer the nodes are the more important are their locations to each other. Thus nodes broadcast their locations periodically along with a lifetime to restrict the travel of the control packet up to the specified distance from the originator. The broadcasting frequency also depends on the mobility rate: frequently moving nodes may need to trigger control packets more often. The packet forwarding method in DREAM is similar to that of LAR, i.e., the sender defines a sector towards the destination that contains the expected

zone of the destination. All nodes inside this sector are the forwarders of the packets along the path from the source to the destination.

Guaranteed-delivery routing

Greedy Perimeter Stateless Routing (GPSR) [33] and *Greedy-Face-Greedy (GFG)* [7] have been proposed to address the local maxima problem of the original Greedy scheme. In both schemes, routing is performed in two modes *greedy* and *face or perimeter*. The protocol starts with the greedy mode and at every step packets are forwarded to the neighbor closest to the destination. Upon hitting a local maximum, the protocol switches to face mode. The face routing is based on the *right-hand-rule* for exploring a maze. The rule says that a maze explorer could successfully traverse it by placing the right hand on the maze-wall and keep walking. It will eventually take the explorer out of the maze. Thus if we consider a network topology as a maze, which is a connected graph, we may use the above rule to explore it. However, the connected graph needs to be *planar* (a graph without any crossing edges). This ensures the explorer is not trapped into a loop. Thus a connected and planar graph could be traversed using right-hand-rule. GFG and GPSR use *Gabriel Graph (GG)* [18] and *Relative Neighborhood Graph (RNG)* [69] as the planar graphs. The planar graph could be constructed in a distributed fashion at each node while routing a packet.

Greedy mode : The current node forwards the packet to the neighbor closest to the destination. In the case of a local maximum, i.e., no further neighbors can be found to be closer to the destination, the routing switches to the face or perimeter mode.

Face or perimeter mode : Let f be the node where this mode has been initiated and j be the destination. In perimeter mode, the packet needs to traverse all the connected faces intersected by the $\overline{fj}$ line. f forwards the packet to the next node x in the current face according to the right-hand-rule and the process keeps going until an intersection between the lines $\overline{cx}$ and $\overline{fj}$ is found, where c is the current node. In this situation, a face change is required to ensure progress towards the destination, i.e., c instead of forwarding the packet

to *x*, needs to choose the next node from the next adjacent face towards the destination according the right-hand-rule. At every step of face mode, the current node also checks whether the next node is strictly closer to the destination than the node which was the local maximum. The protocol then reverts to the greedy mode at this point.

The above two protocols seem attractive in terms of packet delivery ratio as they always guarantee packet delivery if there is a path between a given source-destination pair. However, the assumption of unit disk graph and exact location of nodes are not valid in real life. In particular, the constructed planar graph may have crossing edges due to irregular radio range of the nodes, which may lead to routing loops. Indeed routing nodes are forced to take on the extra burden of planar graph construction.

Progress-based routing

There are numerous protocols proposed under this subclass of routing schemes. One of the simplest progress-based routing schemes is *Greedy routing* [16], whereby the routing node selects the next-hop neighbor as the one with the shortest distance to the destination. Although the scheme works statistically well in dense networks, it suffers from the problem of *local maxima*, i.e., nodes with no neighbors in the transmission range located closer to the destination than themselves. Another position-based approach, called *Compass*, was proposed in [35]. With Compass, a routing node *i* forwards the packet to the neighbor *j* that minimizes the angle formed between *j*, *i* and the destination. The protocols, whose performance is similar to that of *Greedy*, suffers from loops. Also, in resource constraint environments Greedy and Compass perform poorly because of the tendency to exhibit hot spots. Figure 2.3 shows routing examples using Greedy and Compass protocols. Say *I* and *J* are the source and destination, respectively. *I* has three neighbors, but *A* is closest to *J*, whereas *B* minimizes the angle towards it. Thus Greedy selects *A* and Compass chooses *B* as the next node to route the packet to *J*. However, at *A* packet enters at local maximum problem as it does not have a neighbor closer to *J* than itself. Greedy routing fails at this

point. On the other hand, packet traps in a loop between B and A in case of Compass routing. Even though both protocols fail to route the packet, there is a clear path between I and J, which could be made by clever protocol design. Randomization could be a smart choice in such situations as by definition packets may follow different routes even for the same source-destination pair.

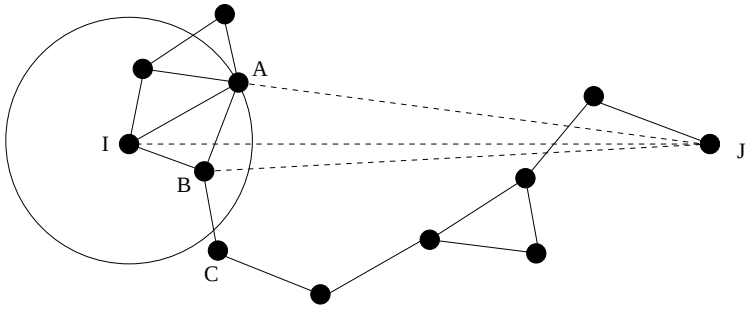


Figure 2.3: An example of location-based routing.

A straight forward randomized routing scheme is *random walk*, where routing node i randomly chooses a neighbor to forward packets. However, this may lead the packets to follow unnecessarily a long route. Thus, in this thesis we introduce two variants of random walk: namely *RW-90* and *RW-180*, where i randomly chooses a neighbor from inside a sector of size $\pi/2$ and π towards the destination, respectively.

In *Geographic Random Forwarding (GeRaF)* [76], a routing node i first divides the forward transmitting region towards the destination into A_k sub-regions. Region A_1 is the one closest to the destination and can be defined by taking an arc centered at the destination with the radius $r = (d_{ij} - R) + a$, where d_{ij} is the Euclidean distance between nodes i and j and $0 < a <= R$. i then randomly chooses a neighbor from the region closest to the destination and relays the packet to it. If this region is empty, subsequent regions are sequentially searched until a relay neighbor is found. If all regions are empty, the packet is dropped. Note that performance of GeRaf is highly dependent on the size of the subdivided regions and may create hot spots in region A_1. For instance, in the above example, node A

might be the only choice from region A_1 and could be congested in a highly loaded network. Thus it is a good idea to consider more neighbours as the potential next-hop nodes and assign weight to them to control the congestion and routing path (e.g., the weight could be assigned based on the remaining distance or minimum angle so that the resultant path length could be smaller.)

As the above schemes do not take into account the energy budget, they do not perform very well in an energy constrained environment [58]. As pointed out in [61], these algorithms will generally perform poorly in such environments because of the tendency to exhibit hot spots. Note that the random walk may avoid hot spots, but with high probability it may follow a long route that also consumes more resources. In the following we will present location-based routing that considers energy budget while routing.

The proper definition of local progress as the routing goal is a prerequisite for efficient routing in MANETs. Such definitions are proposed in [37], where the *power progress* objective is to minimize $(\frac{d_{ix}^{\alpha}+r}{d_{ij}-d_{xj}})$ where i is the forwarding node, x is its neighbor, and j is the destination. r is the constant component of a single forwarding step and α transforms the transmission distance into the requisite transmission power. The denominator captures the proportion of the contribution of a single hop to the packet delivery task. To account for the remaining lifetime of a node, the *cost progress* is defined as minimizing $(\frac{f(x)}{d_{ij}-d_{xj}})$ where $f(x)$ is the inverse of the node's remaining lifetime interpreted as its reluctance to forward. The two components can be combined into a single metric by straightforward multiplication and the resulting approach is named as *power-cost progress*. The *projection power progress*, *projection cost progress*, and *projection power-cost progress* can be defined by replacing the denominators from the above progress-based definitions with the dot product over the vectors d_{ij} and d_{ix}. However, as these are deterministic approach, they may suffer from the same drawbacks of Greedy and Compass protocols.

Another energy-aware routing scheme, dubbed PGR [58], operates in two phases. In the *discovery phase*, each node updates geographic location, remaining battery power, and

reliability of neighbors through the periodic exchange of HELLO messages. The *reliability* is defined as the ratio of the number of HELLO packets successfully received at their destinations within some time Δ to the total number of packets that would have been received within the same period, if every transmitted HELLO packet had always made it to the neighbor [58]. The inverse of reliability is the expected number of retransmissions required to deliver a packet. Once the discovery phase is over, each node selects a certain subset of its neighbors with the highest reliability as the prospective next-hop nodes. A routing node *i* first defines an angular sector centered at itself and around the direction of the destination. The initial size of this sector is chosen arbitrarily and then increased up to at most π, if it contains less than two neighbors. If the enlarged sector still has less than two neighbors the packet is dropped. Otherwise, *i* assigns every neighbor *x* falling into the sector the rank $P(x) = \frac{E_{res}(x)}{TR(x)}$, where $E_{res}(x)$ is the residual power of *x*, and $TR(x)$ is the inverse reliability of *x*, i.e., the expected number of retransmission required over the link. The next-hop node is then chosen at random with the probability directly proportional to its rank. PGR may suffer from node congestions as most reliable links get highest rank. Also, it may follow a long route in case of empty initial sectors.

Our proposed routing protocols, *Directional Location-based Randomized (DLR)*, *Forward First with Ranking (FFR)*, and *Forward with Random selection out of Two (FRT)* [23, 24], could avoid local maxima and loops by exploiting the random next neighbor selection. Unlike random walk, GeRaF, or PGR they could also control their path length by introducing weight to the randomized neighbor selection, where the weight could be determined based on various criteria such as remaining distances, angles etc. Indeed, they could also balance load by avoiding congested nodes on their way to the destination.

2.4 Protocol Definitions

In this section we present our proposed protocols. We first define our notation. Suppose that the current routing node, the next-hop node, the destination, and the number of neighbors of i are i, x, j, and n, respectively. The Euclidean distance between two nodes i and j is denoted by d_{ij}. Let $\theta_{x_l} = \angle j i x_l$ be the angle formed between i, j, and one of the neighbors of i, x_l. $E_{res_{x_l}}$ represents the residual energy of a neighbor x_l.

Algorithm 1 Directional Location-based Randomized (DLR) (i, j, Θ)

1: **for** $l \leftarrow 1$ to n **do**
2: Assign rank $P(x_l) \leftarrow E_{res_{x_l}}$ to the neighbor x_l of the routing node i.
3: Assign weight $W(x_l) \leftarrow \frac{1}{(E_{res_{x_l}} + c_0(\frac{n-1}{2\pi})\theta_{x_l})}$ to the neighbor x_l of the routing node i.
4: **end for**
5: Define a sector S of size Θ around the routing node i towards the destination j.
6: Select the *candidate next nodes*, *nc*, from inside the sector S.
7: **if** $nc\ != NULL$ **then**
8: Choose next node x out of *nc* proportional to the rank $P(x)$.
9: **else**
10: Choose next node x out of n proportional to the weight $W(x)$.
11: **end if**

In DLR, the sector size Θ is chosen as $\pi/3$ so that with high probability i may have neighbors inside the sector, which will also be close to the direction of j compared to the remaining neighbors. The closer the direction the better the chance that the protocol follows a shorter route to the destination. In case of empty sector, weight is assigned to the neighbors according to their angles so that a neighbor with smaller angle may get higher priority to be chosen as the next node x. Assuming that the traffic load and the mobility of nodes are uniform across the network, then we may expect that the energy depletion is approximately the same across all nodes. Therefore, if nodes start with the same energy reserves, they may approximately have equal reserves at a later point. Hence, even though the weight biases in favor of nodes with less energy, a suitable choice of constant c_0 can amplify the impact of the angle to be dominant over the smaller differences we expect in terms of the energy across nodes.

Algorithm 2 Forward with Random selection out of Two (FRT) (i, j, Θ)

for $l \leftarrow 1$ to n **do**
    Assign weight $W(x_l) \leftarrow d_{ix_l} \times |\cos\theta_{x_l}|$ to the neighbor x_l of the routing node i.
end for
Define a sector S of size Θ around the routing node i towards the destination j.
Select the *candidate next nodes*, *nc*, from inside the sector S.
if $nc != NULL$ **then**
    Rank neighbors from inside S in terms of maximizing $W(x_l)$.
    Choose first two such neighbors, x_1 and x_2, with highest rank.
    Choose next node x between x_1 and x_2 uniformly at random.
else
    Drop the packet.
end if

The sector size Θ is chosen as $\pi/2$ and π in case of FRT-90 and FRT-180, respectively. FRT initially considers two neighbors that ensures highest progress towards the destination. Then select one of them uniformly at random as the next node to balance the load and to avoid creating congestion to the best candidate on the way to a destination.

Algorithm 3 Forward-First with Rank (FFR) (i, j, Θ)

for $l \leftarrow 1$ to n **do**
    Assign weight $FW(x_l) \leftarrow d_{ix_l} \times |\cos\theta_{x_l}|$ to the neighbor x_l of the routing node i.
    Assign weight $BW(x_l) \leftarrow d_{x_l j} \times |\theta_{x_l}|$ to the neighbor x_l of the routing node i.
end for
Define a sector S of size Θ around the routing node i towards the destination j.
Select the *candidate next nodes*, *nc*, from inside the sector S.
if $nc != NULL$ **then**
    Choose next node x out of *nc* such that $FW(x)$ is maximum.
else
    Choose next node x out of n such that $BW(x)$ is minimum.
end if

The sector size Θ is π for FFR. It selects the next node from inside the sector S that maximizes the progress towards the destination. However, instead of dropping the packet in case of empty sector (unlike Greedy, Compass, GeRaF, and PGR do) FFR considers remaining neighbors to forward packets to the destination. Thus it may follow a suboptimal route but may able to reach destination with high success rate compared to the other schemes.

In a nutshell, we proposed a new set of location-based routing protocols that are designed as a compromise between the packet delivery ratio, the path length, the loop freedom, the network lifetime, and the throughput. Indeed we explore the impact of different load conditions and network topologies on the performance of these proposed protocols and the state-of-the-art.

2.5 Performance Study of DLR

2.5.1 Simulation environment

A set V of N nodes is generated at random and uniformly spread over a 1000m$\times$1000m rectangle (1000m$\times$1000m$\times$1000m in 3D), with $N \in 50, 70, 90, 110, 130$. The nodes are moving using the random waypoint model with the maximum speed of 10m/s. The transmission range of each node is 200m (300m in 3D) and its initial energy is 600J. The transmit and receive power is 25dBm. Each experiment run simulates 500sec of operation. Each data point is the average result from 10 independent replications. In the results reported in this chapter, Confidence Intervals of 95% have been calculated but they are very tight around the averages reported, hence they are not included in plots. However, standard deviation is included when tabular format is used to present the results. In order to maintain up–to–date information about the neighboring nodes, a neighbor discovery process is needed. Upon bootstrap, the initial neighbor discovery phase takes 40 sec, with each node emitting a beacon message every 4 sec. Neighboring nodes respond to the beacon. Following the bootstrap phase, beacons are sent once every 10sec. Routing starts after the neighbor discovery phase has been bootstrapped. Data traffic is generated by randomly selecting source–destination pairs. Each source–destination pair generates a packet to route once every 15 sec. We also consider the node mobility according to the model defined in Section 2.2.

2.5.2 Performance comparison of DLR in 2D

The packet delivery rate of DLR vs PGR and Greedy is shown in Table 2.1. Note that the packet delivery rate of DLR is higher than in the other protocols. This is because DLR increases the choice of alternative paths and thus reduces the packet dropping rate. The performance of Greedy and PGR is very close. PGR can reach the destination as long as it finds eligible forwarding nodes inside the sector. Greedy may drop packets due to local maxima, even though the chance of facing local maxima is less in dense networks.

As the number of nodes increases, all the protocols exhibit better performance due to high node density. For example, the chance of facing a local maximum by Greedy is reduced, which tends to increase the packet delivery rate. In PGR, the chance of having more forwarding nodes inside the sector increases with the increasing node density, which also pushes up the delivery rate. Also in DLR, the delivery rate is slightly improved owing to the reduced probability of reaching the threshold.

Table 2.1: The average packet delivery rate in 2D space.

	$n = 50$		$n = 70$		$n = 90$		$n = 110$		$n = 130$	
Algorithms	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.
GREEDY	69.50	19.86	91.90	11.23	96.80	4.89	98.30	2.71	99.40	1.90
DLR	82.90	16.26	91.90	12.78	99.30	2.21	99.50	0.85	99.70	0.95
PGR	68.30	19.97	90.10	13.09	97.50	3.87	98.00	2.21	99.40	1.58

The performance of the three routing strategies in 2D in terms of the network lifetime is shown in Table 2.2, with PGR being the winner. This is because PGR considers residual energy of nodes and link reliability to balance the energy utilization among the nodes. The next protocol is Greedy, which does not confine routing to a narrow sector, which gives it more flexibility to distribute the energy utilization among the neighbors of a routing node. Finally, DLR has the worst performance in terms of the network lifetime. In DLR, a packet may bounce back and forth, possibly several times, before arriving at the destination, which

may increase overall energy usage by involving more nodes than necessary.

With fewer nodes in the network, Greedy performs better than PGR. The probable explanation is that PGR has fewer choices for next-hop nodes, due to its dependence on the link reliability and the sector size. The Greedy protocol retains a relatively large choice for balancing energy, even within a relatively sparse network. This advantage disappears with increased node density, as the choice for PGR becomes relevant and discriminating. Similarly, the performance of DLR also improves with the increasing node density as the likelihood of "bouncing" a packet is reduced.

Table 2.2: The average network lifetime in 2D space.

	$n = 50$		$n = 70$		$n = 90$		$n = 110$		$n = 130$	
Algorithms	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.
GREEDY	51.20	16.82	62.90	15.43	89.90	29.17	103.40	24.17	134.80	26.73
DLR	15.30	8.41	36.00	24.44	47.90	23.48	76.20	33.63	80.30	44.61
PGR	46.50	15.09	70.10	14.74	90.60	26.97	106.60	33.51	156.80	73.11

Table 2.3 shows the performance of three routing schemes in terms of the average path length. Greedy is the winner here, followed by PGR and then DLR. This result was expected as minimizing the distance towards destination is Greedy's primary objective. Both PGR and DLR may traverse a longer route due to the (biased) randomization. Then, in DLR, packets may occasionally travel backwards, which can never happen in PGR.

As the number of nodes increases, the path length of Greedy decreases slightly. In DLR, the likelihood of a backward "bounce" decreases, and so does the average path length. PGR, however, may need to traverse a few extra hops in such circumstances, as it always prefers shorter links with high reliability. Hence, the path length of PGR tends to increase as the network becomes denser.

In summary, DLR's enhanced packet delivery rates comes at increased energy cost.

Table 2.3: The average number of hops in 2D space.

	$n = 50$		$n = 70$		$n = 90$		$n = 110$		$n = 130$	
Algorithms	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.
GREEDY	3.67	0.32	3.97	0.17	3.85	0.39	3.80	0.29	3.65	0.17
DLR	9.71	1.42	7.97	4.00	6.93	3.36	5.52	1.31	4.71	0.41
PGR	3.83	0.44	4.40	0.20	4.25	0.31	4.51	0.41	4.40	0.39

2.5.3 Performance comparison of DLR in 3D

Table 2.4: The average packet delivery rate in 3D space.

	$n = 50$		$n = 70$		$n = 90$		$n = 110$		$n = 130$	
Algorithms	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.
GREEDY	58.70	8.90	81.10	7.95	87.50	5.08	94.70	5.93	98.10	1.60
DLR	71.40	13.05	91.70	10.12	94.60	4.95	99.60	1.26	99.90	0.32
PGR	65.90	7.19	82.80	6.92	89.40	7.85	94.90	5.45	98.60	1.17

The behavior of the three protocols in 3D is highly consistent with their planar performance. In particular, DLR still offers the highest packet delivery rate, the performance of Greedy and PGR is almost the same and close to that of DLR. The network lifetime of Greedy starts better than PGR, but, as the node density increases, PGR picks up. DLR still yields to the other protocols in terms of network lifetime, improving its performance with increased node density. In terms of the path length, the performance of the three protocols is also similar to their 2D variants.

Table 2.5: The average network lifetime in 3D space.

	$n = 50$		$n = 70$		$n = 90$		$n = 110$		$n = 130$	
Algorithms	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.
GREEDY	55.00	13.66	66.00	20.26	93.10	20.15	114.50	26.85	115.00	31.66
DLR	13.30	7.94	17.20	8.13	22.70	13.73	27.60	16.97	41.30	20.62
PGR	42.90	8.84	62.00	21.07	69.10	34.42	114.20	27.78	164.40	80.76

Table 2.6: The average number of hops in 3D space.

	$n = 50$		$n = 70$		$n = 90$		$n = 110$		$n = 130$	
Algorithms	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.
GREEDY	3.25	0.27	3.57	0.20	3.58	0.23	3.42	0.20	3.42	0.15
DLR	9.24	1.26	11.07	0.84	9.65	1.39	8.87	0.74	7.44	0.99
PGR	3.99	0.25	4.13	0.41	4.52	0.64	4.02	0.58	3.81	0.25

2.6 Performance Study of FFR and FRT

2.6.1 Simulation environment

We have evaluated our protocols under SMURPH (a System for Modeling Unslotted Real-time PHenomena) [19]. For grid network models, we considered a 10×10 perfect grid with the edge (node-to-node link) length of 50m. The random topologies were generated by uniformly distributing 75 nodes within a 500m $\times$ 500m area. The transmission radius was set to 90m in all the experiments. For randomly generated topologies, only connected networks were used. Note that we use static environment for this simulation (unlike in DLR) as our main goal is to observe the behavior of different protocols under variuos load conditions and network topologies without making the environment more complex with node mobility. Also, we use differnt radio range and network size but they could be rescaled to the same size used in Section 2.5.

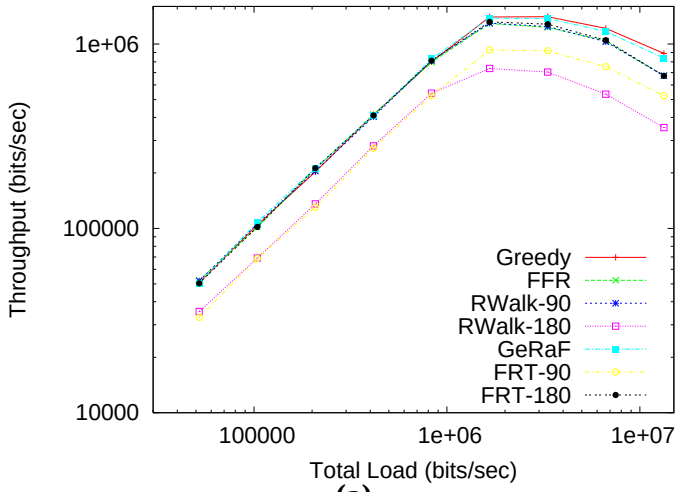
2.6.2 Results

Grid network with uniform traffic

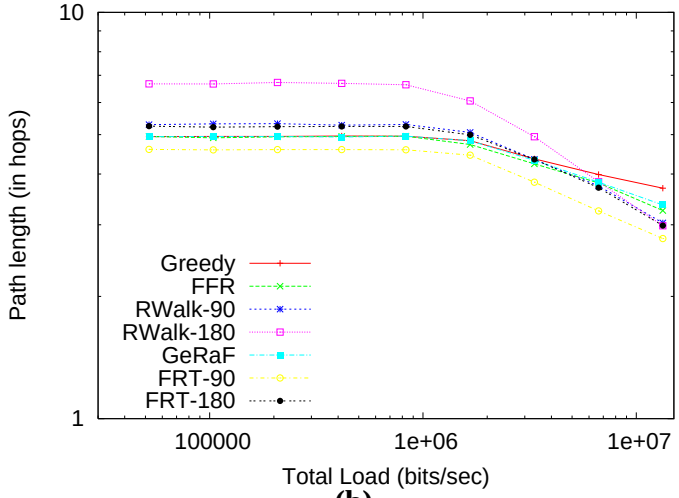
RW-180 has the worst performance under all loads. The shortcomings of RW-180 are obvious: the protocol selects the next node at random from all neighbors inside the sector of size π. Odds are against an optimal choice, even if by restricting the sector to π we can guarantee some "progress" towards the destination. Thus, packets may have to traverse a long route and hence occupy the network for long time resulting in more packets competing for the channel and, eventually, higher dropping rate. However, the throughput of the smaller sector variant of RW-180, i.e., RW-90 is almost best under low load. This performance though degrades with increasing load. Thus by restricting the sector size RW-90 controls the path length and reaches the destination quicker than RW-180. However, at high load it still may leave some packets that eventually do not reach the destination.

At low load both FRT-90 and RW-180 perform poorly though FRT-90 performs better than RW-180 under heavy load. The performance of FRT-180 and FFR is also similar to that of RW-90, which falls in the second best protocols group. Due to the limited sector size of $\pi/2$, biased randomization, and maximized projection, RW-90, FRT-180, and FFR might end up following similar paths. FRT-90 achieves better throughput compared to RW-180 as it may have limited number of eligible neighbors within its sector and reach the destination faster than RW-180. However, restricting the sector size along with the biased randomized selection might not be a good choice while using bursty traffic. It may happen that the candidate nodes of FRT-90 suffer from congestion (it chooses two neighbors first and then randomly selects one). RW-90 on the other hand restrict the sector size but again does not squeeze the set of eligible next nodes, which in turn helps it to balance the load better compared to FRT-90.

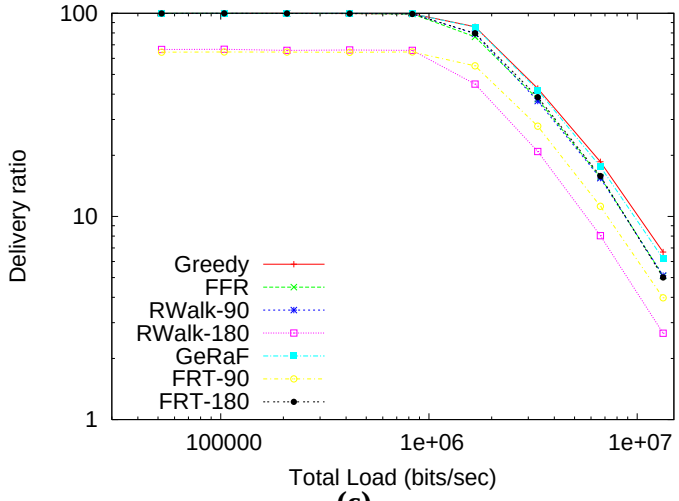
Greedy has the best throughput under uniform burst traffic on grid network. GeRaF can comfortably compete with Greedy across the entire range of traffic loads. GeRaF divides the forward region into sub-regions based on geographic progress towards the destination,



(a)



(b)



(c)

Figure 2.4: (a) average network throughput, (b) path length, and (c) packet delivery ratio, for 10×10 grid.

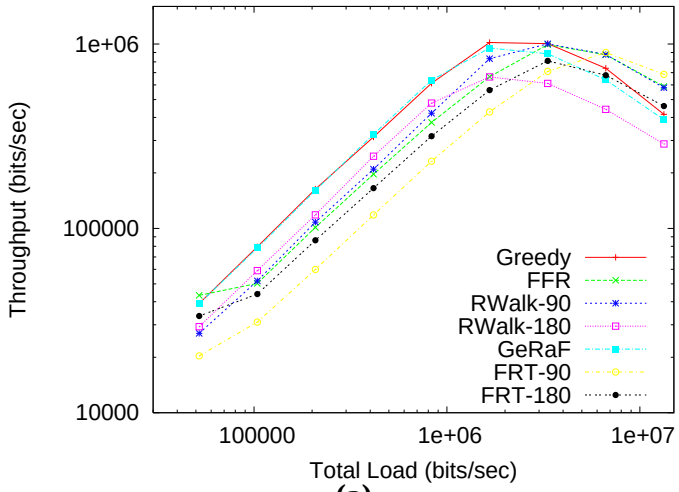
if the size of these regions are reasonable (in grid network it may happen that only one or two neighbors might be available in each regions) and next node is chosen from the closest region towards destination, chance is high that the closest node towards the destination might be selected. This way GeRaF may reach the destination quickly without leaving packets in the network for prolonged periods of time.

As expected, RW-180 has the worst path length; whereas, the best path length is offered by FRT-90. The remaining protocols fall in between. RW-90 and FRT-180 have slightly longer path compared to FRT-90. Greedy has the worst path length under highest load while RW-180 improves its path length under same conditions. At heavy traffic the latter protocol may succeed for relatively close source-destination pairs, which in turn offers short path length. Greedy however may deliver more packets even for longer paths. All other protocols retain their behavior under heavy loads.

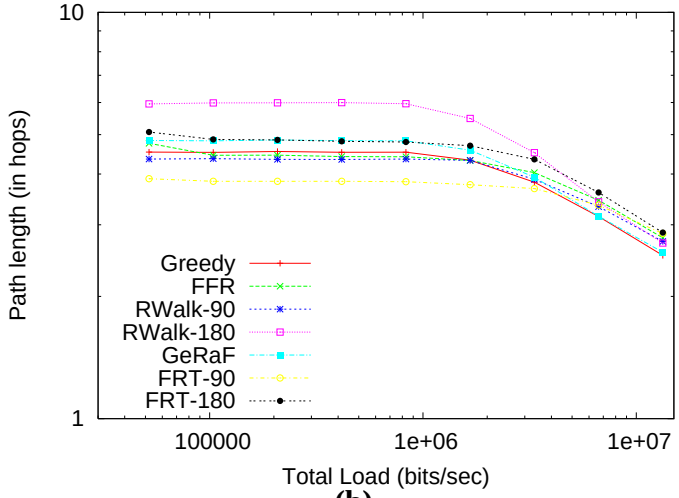
Random network with uniform traffic

Under light traffic, FRT-90 has the lowest throughput, whereas, Greedy and GeRaF have best performance. The remaining protocols perform some place in the middle of those extremes. Interestingly under heavy load the relative performance changes. FRT-90 outperforms all other protocols, but Greedy and GeRaF degrade their performance significantly. They suffer from congestion while load is high and this is expected. FFR performs better than Greedy under high load as it has the option of choosing a neighbor from the backward sector while the forward sector is empty, this might help FFR to follow alternate routes to the destination. Also, again at high loads, RW-90 follows closely the best performance while RW-180 exhibits the worst performance.

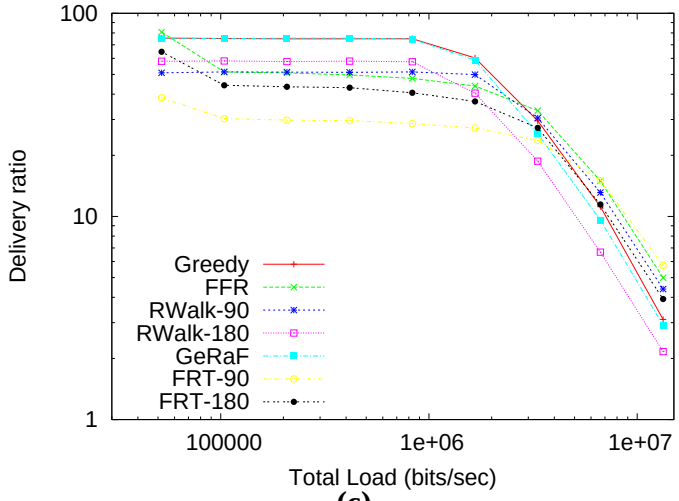
It appears that the sector size has a great impact on the performance of protocols. FRT-90 and RW-90 have a limited number of neighbors, which does not ensure good load balance, but helps to retain a steady performance under all loads because traffic reaches the destination quickly to leave more room for additional packets. Increasing the sector size to



(a)



(b)



(c)

Figure 2.5: (a) average network throughput, (b) path length, and (c) packet delivery ratio, for random topology.

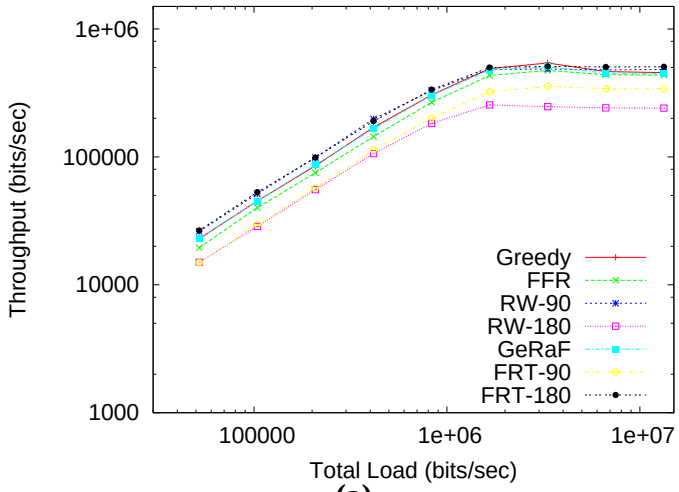
double helps to find more neighbors to balance the load. We observe that under low load RW-180 balances the load nicely, but as load increases due to true randomness (no bias) it leaves more packets in the network and longer compared to those with smaller sector size. Hence, quickly degrades its performance to the worst one. FRT-180 shows just the opposite behavior, i.e., due to less number of congested candidate nodes, packets reach the destination quickly and the result is evident under heavy load.

The lesson here is that under heavy load, a randomized scheme should use selection criteria based on smaller sector sizes. The cost of randomized schemes is the selection of non-optimal next hops, and it needs to be reduced when the network load is high. We could therefore argue that a design criterion for adopting randomized protocols is to (a) make them competitive against Greedy and (b) allow them to "sense" the load of the network in order to reduce the average cost of non-optimal choices when the network load is high.

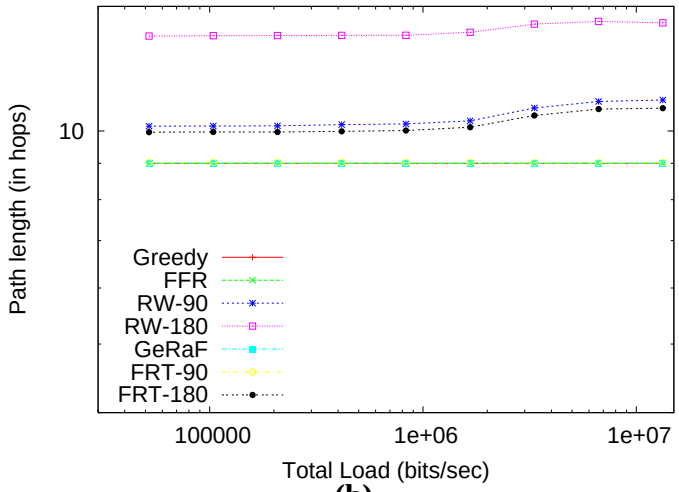
Grid network with biased traffic:

The most interesting behavior is observed while using biased traffic on a grid network. The deterministic protocols Greedy and FFR suffer from congestion and FRT-90 also follows them due to limited sector size. However, they follow the shortest paths compared to other schemes. On the other hand, the best performance is observed for two randomized protocols with notably worse hop count performance, namely RW-90 and FRT-180. RW-90 expresses randomization without any ranks or weights to bias next node choices, but focused within a narrow sector. Opting for a larger sector (like RW-180) can destroy its advantages. On the other hand, FRT-180 expresses randomization with ranking of nodes based on some figure of merit that characterizes good and bad choices, but is given a larger sector to pick for its possible candidates. The two schemes achieve comparably high performance.

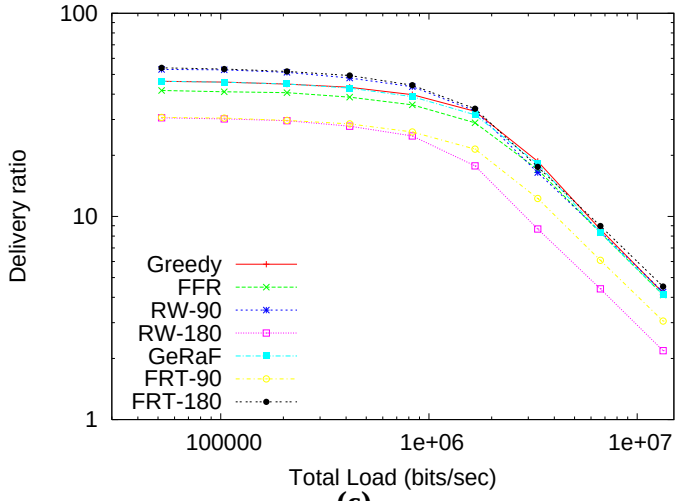
Of course, nothing is for free, and the sub-optimal choices of RW-90 and FRT-180 eventually catch up with them at high loads, where Greedy and FFR converge with the performance of RW-90 and FRT-180. Their differences become small to be statistically



(a)



(b)



(c)

Figure 2.6: (a) average network throughput, (b) path length, and (c) packet delivery ratio, for 10×10 grid with biased traffic.

significant, yet at ultra-high loads, FRT-180 and RW-90 hold their ground quite well as they are able to utilize nodes that are not hot-spots, whereas Greedy and FFR are unable to do so. The explanation of what happens at very high loads is better provided in the context of the number of hops where a slight increase of the number of hops is seen at very high loads for RW-90 and FRT-180. This means that the packets that eventually make it to their destinations have followed inflated paths (possibly by only just one hop) but that allowed them to bypass hosts where the high congestion would have resulted in packets being dropped.

2.7 Conclusions

In this chapter, we consider location-based routing because the nodes do not require to maintain routing tables but use location information for routing. This makes location-based routing suitable for wireless networks with limited resources like MANETs. In location-based routing pure progress-based next node selection seems attractive due to simplicity and effectiveness in a dense network. However, in a sparse network routing protocols like Greedy and Compass suffer from local maxima and loops, respectively. While GFG and GPSR protocols were proposed to handle these issues, they require a planar network topology which might not be possible in real deployment due to irregular radio range of nodes. We propose to consider randomized next node selection to address the above problems and propose DLR routing as the first step.

The main objective of DLR is to avoid local maxima and loops and provide high percentage of packets delivery. DLR does not easily "give up" forwarding packets, in that it is willing to divert packets away from the path to the destination and play the odds that at some later time the packet will be eventually pushed in the right direction. Due to the central importance of energy constraints in routing protocols for mobile ad hoc networks, it is desirable that any other routing objective be seen from an energy consumption perspective.

In the particular case of location–based routing, its synthesis with energy considerations can be achieved by ensuring that the selection of the next–hop node is made on the basis of a combination of energy and direction considerations. In addition to offer high packet delivery ratio, DLR also attempts a synthesis of the two factors.

Other than the local maximum and loop problems, deterministic approaches may also create congestion to some nodes in the network. Again randomization might be considered to avoid such congestion or hot spots. For this purpose we propose additional routing protocols FFR and FRT. We avoided relying on one type of network topology only. Instead we considered regular (grid) as well as random topologies. Likewise, we considered uniform and non-uniform traffic. Some of our intuitions, like "routing randomization works well with random networks" proved to be wrong. In addition, we established that, yes, randomized routing protocols could deal with random environments but one has to be weary of the additional cost that randomization will place on path lengths and therefore congestion. We also noted that the simplest location-based scheme: Greedy, is quite capable to handle a variety of scenarios except for high load non-uniform traffic situations. However, randomized routing has a lot to offer for biased traffic, even in regular topologies.

Chapter 3

LEMON: an RSS-based Indoor Localization Technique

3.1 Introduction

We have proposed a set of location-based routing protocols in the previous chapter that do not require to maintain detail routing tables to route packets. Thus, they could be considered as the routing solutions for resource constraint devices, like sensors. However, locations of the nodes need to be known for such location-based routing schemes. Existing solutions assume that the nodes are equipped with the GPS and thus determining their location is not a problem. However, its applicability on resource constraint devices is questionable because of 1) cost, 2) form factor, 3) accuracy, and 4) unavailability of GPS indoors. These reasons motivate us to come up with an indoor localization system that could be useful not only for location-based routing but also for a wide range of other applications. In this chapter we propose an RF-based indoor localization scheme called LEMON based on low-power and low-cost wireless devices. The experimental results show that the LEMON competes with (and often exceeds) the state-of-the-art in terms of the localization accuracy.

3.2 An Overview of Indoor Localization Schemes

While, generally, GPS handles outdoor localization quite well, its performance indoors is not satisfactory. Even forgetting about the virtual unavailability of the GPS satellite signals under the roof, the limited precision of GPS (aggravated by the unavoidable multi-path effects), practically reduced to about 20m [46], does not meet the expectations of typical indoor applications, e.g., determining the side of the wall on which the subject is located. On top of that, the relatively high cost and power expenditure make it difficult to use GPS on a massive scale and in an inconspicuous (unobtrusive) manner. Thus we need alternative solutions for indoor localization.

As we mentioned earlier our focus is on RF-based techniques, which are by no means the only possible choice. Among the non-RF based localization techniques applicable to indoor environments, one can mention ultrasound technologies [73] and infrared (IR) systems [72]. While the acoustic schemes work reasonably well at close range and without obstacles "getting in the way," in realistic indoor scenarios, they suffer from serious interference problems. Infrared solutions, on the other hand, in addition to limited range and poor accommodation of LOS obstacles, may completely fail in the presence of sunlight (e.g., rooms with big windows) and are rather costly [3]. Consequently, people usually consider RF-based solutions as being most practicable and cost-effective [3].

Theoretically, the propagation properties of RF signals make it possible to deduce the location of tracked objects by measuring some of those properties at the receiver. For example, the tracked object may emit an RF signal (at some known power) whose strength is measured at the (infrastructure) receivers at known locations. Alternatively, the infrastructure nodes may emit signals whose intensity is measured by the tracked receiver. Instead of the signal's received strength, the measured properties may be the *time of arrival* (TOA) [73], *time difference of arrival* (TDOA), or the *angle of arrival* (AOA) [47]. In TOA-based estimation, the velocity, v, of a signal and the propagation time t from the transmitter to the receiver are used to measure the distance $d = vt$ traveled by that signal.

As the precise TOA measurement of a radio signal poses nontrivial synchronization challenges [8], especially indoor, where the receiver has to deal with multiple paths, hybrid solutions are sometimes devised. For example, in Active Bat [73], the RF signal is used for the synchronization between the transmitter and receiver and ultrasound is used for the actual measurement of distance.

With a typical AOA scheme, the receiver consists of a series of antenna arrays: using their adjacent distance and signal propagation time it is possible to measure the angle of arrival. Angulation is then applied to estimate the transmitter's location. Although generally TOA and AOA systems offer better accuracy than RSS-based schemes, they depend on expensive and delicate hardware which incurs extra cost and adds to the energy consumption [8].

These shortcomings can be overcome by measuring solely RSS, which is a very basic indication available in the lowest end (cheapest) transceivers. RSS-based solutions can be further classified into *range-based*, *range-free*, and *profiling-based* schemes. With a typical range-based approach, signal attenuation is assumed to follow a certain distance-dependent formula, e.g., determined by a free-space, two-ray, or shadowing channel model [56]. The last one is considered the best (most general) representation of RF signal propagation, with the received power determined as:

$$P_r(d) = P_r(d_0) - 10\alpha \log \frac{d}{d_0} + X_\sigma$$

The first two terms stand for the received power at some distance d and some close-by reference distance d_0. The factor α is the path loss exponent, and X_σ is a Gaussian random variable (expressed in dB) with zero mean and standard deviation σ. A range-based method may use this model to determine the distance between the transmitter and the receiver. However, in the presence of multiple paths, it is hard to precisely predict the path loss exponent α. In practice, a typical indoor environment hardly exhibits a single value of α

for all signal propagation scenarios. While useful for simple simulation studies, a single-parameter model, like the one above, is not much relevant for any quantitative assessment in a real-life scenario.

Range-based localization schemes attempt to mitigate the poor quality of blanket models by adjusting their parameters empirically. For example, with the approach proposed in [32], the fixed infrastructure nodes know the locations of themselves as well as of their neighbors and use this knowledge to dynamically build a path loss model. The parameters of that model are then fitted to transform the RSS from an unknown location into a distance estimate for lateration. A similar solution is described in [2]. Another model-based indoor localization scheme is presented in [1], where thirteen different channels (separated by 5MHz) are used to gather RSS from the tag node being localized. This approach attempts to mitigate the multi-path effect which tends to depend drastically on the frequency. The collected values are subsequently transformed into distance for triangulation. The authors claim that their scheme derives a close approximation of the actual distance from the theoretical propagation model. However, we feel that those claims have not been convincingly substantiated by experiments involving elaborate configurations of practical scenarios including obstacles, walls, etc. Generally, distance estimation techniques based on extensive application of channel models are met with skepticism, especially in indoor applications [3, 74].

In [74], the authors carried out a comprehensive study of RSS characteristics, both indoor and outdoor, to analyze their impact on localization. They focus on three generic attributes: the noise, the attenuation rate, and the effective range. Different environmental conditions and parameters are considered in many possible combinations for the comprehensive study e.g., elevation, transmission power, packaging, the impact of obstacles. They conclude that, elevation and transmission power are the most influencing factors for RSS which is also highly susceptible to environmental changes. They use the localization method presented in [46] and confess that their range-based approach does not work very

well in indoor environments—suggesting profiling as a more promising idea. The same observation is made in [3].

A range-free approach based on Distance Vector routing is proposed in [46]. In the DV-hop variant, the infrastructure nodes (anchors) know the distances among themselves through coordinate exchange via flooding. All regular (tracked) nodes (tags) also learn their distances to the anchors expressed as the number of hops. Note that a tag maintains a table containing X_i, Y_i, h_i, where the first two represent the coordinates of anchor i and the last one is the number of hops from that anchor to the tag. When an anchor i receives the coordinates of the other anchors, it calculates:

$$H_i = \frac{\sum \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}}{\sum h_i},\ i \neq j \quad .$$

to obtain the average hop-size, H_i, which is subsequently broadcast into the network. Once a regular node receives the average hop-size from an anchor (usually the closest one), it can estimate its distances to the anchors. This method does not depend on any propagation property of the RF signals, but the network topology must be reasonably uniform in all directions for the hop-size estimation to be useful [46]. A hybrid of range-free and range-based schemes, called DV-distance, is also proposed where the hop distance is transformed into a geographic (actual) distance based on a channel propagation model. In yet another (third) variant, a tracked node needs to have at least two neighbors knowing the Euclidean distance to an anchor, as well as the distances between themselves. Such a node can then easily calculate its own distance to the anchor. That scheme is claimed to have a better accuracy than the hop-based method [46]. However, deriving the Euclidean distance from an RF propagation model is still required.

Another simple range-free scheme is proposed in [8] where the idea is to decide on a subset of anchors (pegs) deemed to be close to the tracked node (tag) and then simply average their coordinates $\left(\frac{\sum_1^n x_i}{n}, \frac{\sum_1^n y_i}{n}\right)$, into an estimate of the tag's location. While naive at first sight, the accuracy of this scheme will tend to improve with the increasing density

of peg coverage (also implying reduced RF range of the tag's transmitter). It can be viewed as a generic scheme which, in particular, has found its way (albeit in a weighted variant) into LEMON. A good survey of range-free localization techniques can be found in [63].

RADAR [3] can be considered the pioneer of profiling-based RF schemes for indoor localization. During the profiling phase, RSS samples are gathered from four different directions for the same location (to overcome the orientation effect). For localization, a collected sample is compared to the stored set and the coordinates of the closest point from the signal space are reported as the estimated location. Choosing more nearest neighbors and averaging their locations tends to improve the estimation. However, RADAR still suffers from large errors due to the limited number of infrastructure nodes: three long range APs covering the entire monitored area. One attempt to fix the poor performance of RADAR reported in [3] involved a signal propagation model taking into account the presence of walls between the transmitter and the receiver (which the authors considered the primary source of problems). That attempt did not work (the observed performance was even worse) which should be taken as a strong hint that, generally, propagation models cannot compensate for inadequate coverage with infrastructure nodes. Their model is defined as

$$P(d)[dBm] = P(d_0)[dBm] - 10\alpha \log\left(\frac{d}{d_0}\right) - \begin{cases} \alpha W \times WAF & \text{if } \alpha W < C \\ C \times WAF & \text{otherwise} \end{cases},$$

The model attempts to account for the impact of walls separating the tag from a base station, which were the source of serious discrepancies for the simple estimation algorithm. Thus, αW is the number of walls along the path, WAF stands for the wall attenuation factor, and C is the threshold for αW beyond which adding more walls becomes irrelevant.

Several popular localization approaches rely on RFID technology. Such a system usually consists of a set of RFID readers, comprising the infrastructure, and trackable RFID tags. An RFID reader is able to detect the signal from a tag, if it gets sufficiently close.

For a passive RFID tag, this will happen when the distance to the reader is so small that the scheme becomes range-free: detection by a reader is a sufficient estimation of the tag's location. A localization system like this may not provide a full coverage of the monitored area and be only concerned about detecting the presence of tags in certain "critical" places or regions. With active tags, on the other hand, which act like cheap (low-range) transmitters, the readers may be able to meaningfully assess the received signal strength and use it as a representation of the tag's distance, e.g., quantized into a few coarse discrete levels.

One RFID-based representative of the profiling-based schemes is LANDMARC [45]. The network consists of a set of RFID readers as the infrastructure nodes and RFID tags as the sending (tracked) devices. LANDMARC suffers from the technological limitation of RFID readers (the lack of a direct measurement of RSS by the reader). Also, the large diversity of hardware versions of tags impacts the performance.

In [52] the existing residential powerline network is used for localization purposes, with the infrastructure nodes being attached to the powerline around the perimeter of the household. The system, called PLP (for Power Line Positioning) targets residential applications. The signal transmitted by the infrastructure nodes is received by the tracked tag. Thus, with this approach, tags collect signal samples from the infrastructure nodes, not the other way around, as in RADAR, LANDMARC, and also LEMON. During profiling, signatures of signals from known locations are stored in a database. The estimation stage proceeds in two phases: first the room where the tag appears to be present is identified, and using a respectively trimmed down population of samples, the more exact assessment of the tag's location within that room is carried out. However our experimental results show that the two-phased approach to location estimation in PLP may not be an effective approach. The task of accurately inferring whether a tag is in a particular room is often difficult (especially when the tag is positioned close to the wall), and once that decision is made incorrectly, its subsequent refinement is not useful.

With respect to WiFi-based solutions, [65] investigates the practical performance of

WiFi profiling for different numbers of APs. The profile samples are gathered at every grid point, where the grid consists of 1m$\times$1m cells. The closest neighbor from the signal space, like in RADAR, is then reported. The authors propose a model for ranking the collected RSS samples with respect to their contamination level and selecting less contaminated samples for location estimation. The practical performance results reported in [65] are rather disappointing: in our framework, they would translate into several meters of average error distance, even with five APs. One advantage of LEMON over WiFi-based schemes, worth stressing in this context, is the highly reduced power of RF signals resulting in smaller coverage of a single "AP." As our pegs do not compete in any manner and are cheap, we can easily afford a dense deployment translating into a reduced coverage of a single peg. This automatically implies better resolution and accuracy, even without any further tricks. On top of the reduced energy consumption, one should also note the reduction in the pollution of the RF spectrum caused by the signals periodically emitted by the tracked tags. This reduction is not without significance, considering the ever increasing popularity of various RF-based devices (and WiFi enabled devices in particular) in all populated areas.

Another WiFi-based localization scheme is presented in [28] where the authors propose to use pairwise RSS differences from the APs as the profiled samples. This increases the dimensionality of the sample space and, to some extent, may compensate for the limited number of pegs. For example, with four APs, the number of samples from a single collection is six rather than four. Our experiments indicate that having a larger number of actual pegs is much more beneficial than such tricks.

3.3 LEMON: Transforming Samples into Locations

Localization in LEMON is a two-step process: *profiling* and *estimation*. During the profiling stage, the transmitter (tag) is placed at known locations to send a signal, which is then captured by a set of infrastructure nodes (from now on called *pegs*) to generate a

signal strength signature (also called a sample). The collected samples are then stored in the database along with the tag's location. A kind of radio map of the monitored area is thus constructed. After the profiling phase the actual estimation from a query tag can be performed. In all our experiments we consider static node placement. The following description refers to the version of LEMON as presented in [26].

3.3.1 Profiling

A single RSS profiled sample[1] stored in the database can be viewed as a triplet $< C, \Omega, \tau >$, where C stands for the known coordinates of the profiled sample, Ω is the *association list*, and τ, is the sample's *class*, where class identifies the RF parameters of the transmitter (typically transmission power, bit rate, and channel number). The association list Ω is a set of pairs $< p, r >$, where p identifies a peg (an infrastructure node), and r is the signal strength value observed at that peg. In practice, the scheme operates by having the tag periodically transmit RF packets that include a sequence number used to uniquely identify the transmission. A peg receiving such a packet will forward to the central server a report consisting of its own identifier, the identifier of the tag, the packet number and class. The server constructs then the association list based on all the reported RSS readings for the same packet received by the multiple pegs. τ is used to distinguish between different RF parameters. Samples acquired under specific values of those parameters can only be matched to readings collected under the same values. We hasten to add that depending on the environment and the deployment not all pegs may receive a tag's transmission; hence Ω need not contain all p's in its list.

[1] We interchangeably use the terms *profiled samples* and *reference points* to refer to the measured RSS values from the known locations that are stored in the database for localization. Note that a *sample* is the average of multiple RSS *readings*. In all our experiments each sample is the average of 5 individual readings.

3.3.2 Selecting relevant profiled samples

In the estimation phase, having received a location estimation request, the server will search through the database of profiled samples in order to choose a small number of best matching ones. The input to the estimation process is a list of readings $< p, r >$, denoted by Φ, representing the set of pegs that have received the current packet sent by the tag (at the unknown location) along with their RSS readings. One more element of the query is the configuration of transceiver parameters used by the tag and denoted by τ. In the first step of its search, the server will eliminate all those samples that were collected under different RF parameters (τ) than the present set of readings. Having performed this obvious preliminary step, we can forget about the existence of τ: it plays no further role in the scheme.

One simple heuristic used to narrow down the subsequent search consists in selecting from Φ the pair $< p_h, r_h >$, such that r_h is the highest among all pairs in Φ. Then, the procedure will only consider those profiled samples from the database whose association lists include p_h as one of the pegs. Clearly, as p_h appears to have received the present packet at the highest RSS value among all pegs, it makes no sense to even look at samples whose association lists do not include p_h.

3.3.3 Measuring discrepancy in the sample space

Having pre-selected a subset of relevant profiled samples from the database, the server will order them according to their discrepancy from the query sample (association list) Φ (from the tag). Then, it will select K profiled samples in signal space that are "closest" to the query sample. The location estimate is the weighted average of the real locations that correspond to those K closest profiled samples. Clearly, K is a parameter of the proposed estimation procedure.

One can think of several measures of discrepancy in the sample space. For the most natural one, i.e., Euclidean, suppose that $\Omega = \{\omega_1, \ldots, \omega_m\}$ and $\Psi = \{\psi_1, \ldots, \psi_m\}$ are two

associations lists. The discrepancy between these lists is defined as:

$$D(\Omega,\Psi) = \sqrt{\sum_{j=1}^{m}(R_\Omega(j) - R_\Psi(j))^2} \tag{1}$$

where m is the total number of pegs in the network and $R_\Omega(j)$ is defined as r_j, if the pair $< p_j, r_j >$ occurs in Ω, and 0 otherwise. The same is true for $R_\Psi(j)$.

3.3.4 Estimating the location

In the last step, the coordinates of the K selected profiled samples are averaged to produce the estimated coordinates of the query tag. The averaging formula biases the samples in such a way that the ones with a smaller discrepancy from the query sample contribute proportionally more to the estimate. Let d_{max} be the maximum discrepancy from Φ among the best K selected profiled samples and $D = \sum_i^K d_i$ be the sum of all those discrepancies. The located coordinates are estimated as:

$$(x_e, y_e) = \frac{\sum_{i=1}^{K}(x_i, y_i) \times (d_{max} - d_i)}{K \times d_{max} - D} \tag{2}$$

where (x_i, y_i) are the coordinates associated with profiled sample i. In particular, to answer a query from the tag, we first search the profiled data to choose the candidate samples (the association lists consists of peg p_j perceived highest RSS from the query tag). The RSS distance between these samples and the query is then computed and top K of them is chosen in terms of minimizing the RSS distance. The weighted average of these K coordinates is reported as the estimated location of the tag.

Note that the above approach does *not* link RSS to any measure of geographical distance but treats it as a purely numerical attribute of a sample whose value should be close to the observed value. The averaging formula does factor in the magnitude of discrepancy between RSS values (in terms of distance between points in Euclidean space), but this is

a pure interpolation and not an application of any RF propagation model. LEMON does not impose any prior restriction on the number of pegs or reference points. It relies on the matching rules to locate those profiled samples that best apply to a particular "instance" of localization. If the number of profiled samples is large, the role of the last-step interpolation becomes secondary: we do not assume that the specific RSS values encode some information about the real–world distance. In particular, it may make sense to oversample the area, e.g., collecting multiple samples from the same point (which is, e.g., in contrast to [65]). For example, those multiple samples may correspond to the different orientations of the tag, as in [3] (without mentioning the orientation as an explicit attribute).

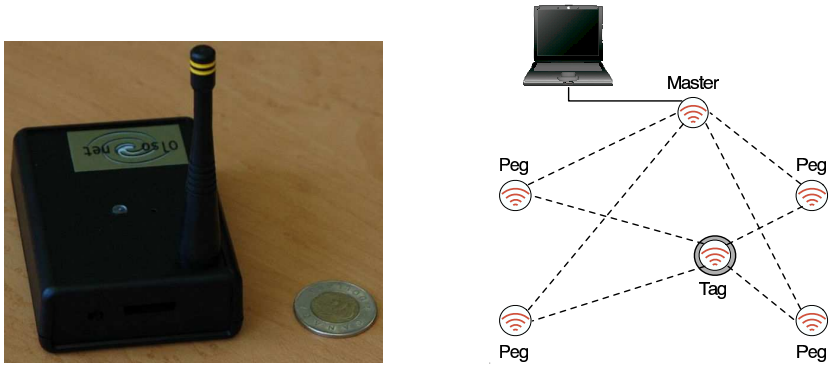
3.4 Experiments

A prototype system was implemented for collecting RSS measurements at various indoor areas on our university campus. The measurements were the basis for evaluating LEMON as well as other localization schemes. This section describes the experimental setup.

3.4.1 The hardware

In our experiments, the wireless device used for both pegs and tags alike is the EMSPCC11[2] from Olsonet Communications shown in Figure 3.1. EMSPCC11 is a low-cost low-power mote for ad-hoc wireless sensor networking programmable in PicOS [20]. The same device, in different casings and with different interfaces, lies at the heart of several WSN applications, including EcoNet and Smart Condo [64]. The mote employs the MSP430F1611 microcontroller and the CC1100 RF module, both from Texas Instruments. The RF module operates within the 916 MHz band. The transmission power is settable from -30 dBm to 10 dBm (in 8 discrete steps), the bit rate options are 5 kbps, 10 kbps, 38 kbps, and 200 kbps,

[2] See http://www.olsonet.com/Documents/emspcc11.pdf for details.



The EMSPCC11 mote **Data collection**

Figure 3.1: The EMSPCC11 mote and the experimental set up to gather signal strength.

and there are 256 different channels (numbered 0 to 255) with 200 kHz spacing. All combinations of options are possible and, in principle, sensible. The experiments reported on in the rest of this work were carried out at the second lowest power setting with 5 kbps transmission rate using channel 0. (Note that lowest power setting helps to reduce interference and saves energy.)

3.4.2 The logistics

As the implementation was intended for research rather than production, all data processing was done off-line, i.e., the system was not used for real-time localization (although there are no fundamental problems barring its true application). The collected data, including the profiled samples as well as specimen tag readings from tracked locations, can be re-interpreted many times within the context of the same experiment. For example, one can try to ignore various subsets of the profiled samples, ignore some pegs, use different metrics for selecting the best-matched set of profiled samples, change the value of K (the number of best-matched profiled samples or nearest neighbors), use different averaging formulas, apply various re-scaling functions to RSS readings, introduce thresholds or clustering (discretizing the RSS values), and so on. Consequently, the separation of data acquisition from

the actual estimation (localization) is the most natural methodology in this type of work.

A typical experiment would start from deploying a number of nodes within the monitored area. Generally, the interpretation of those nodes (tags versus pegs) was left until the off-line analysis stage. The networked program run in the nodes (the *praxis* according to PicOS terminology [20]) allowed us to obtain RSS readings between all pairs of directly reachable nodes and for any selected setting of the transmitter (output power, bit rate, channel number). Thus, the deployment of nodes was usually followed by data collection: the nodes would exchange a massive number of packets, conveying to the central node (dubbed the *master*) their parameters (sender/receiver ID, serial number, transmitter parameters, RSS). The master node was connected (via a USB dongle[3]) to a laptop, where all the data collected by the network were deposited.

During the off-line analysis, some of the collected readings would find their way into the database of profiling samples and some others would be interpreted as measurements of nodes to be localized. As the locations of all nodes were known exactly, their roles as pegs or tags were flexible and depended on whether we pretended we did not know the locations for some of them in order to treat them as the nodes to be localized.

In early single-room experiments, the size of the surveyed areas were: 3m $\times$ 5m (room 1), 7m $\times$ 7m (room 2), and 6m $\times$ 6m (room 3). The rooms were also populated with some obstacles, i.e., furniture and equipment (mostly tables, cabinets, chairs, computers, and related items), to a varying degree, from sparse (a few chairs moved to the walls), to dense (a typical packed office inhabited by graduate students). Subsequent experiments were carried out in a three-room scenario: three adjacent graduate student offices, with each office outfitted with four wooden tables, four chairs, several wooden shelves mounted on the wall, two PC computers, and a steel file cabinet. In one series of experiments, the tag node was moved around by a person: our objective was to determine whether the node's proximity to a body and factors such as, e.g., its orientation, have a significant impact on

[3]EMSPCC11 is equipped with a raw UART interface which can be easily converted to USB.

the localization accuracy.

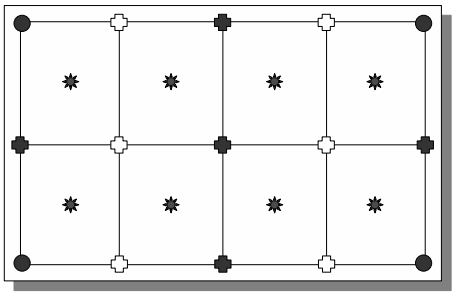


Figure 3.2: A sample distribution of nodes in a room.

Figure 3.1 depicts schematically the data collection phase of a typical experiment, where the master node initiates the exchange of packets and also collects the reported signal readings from the pegs sending them to the server. Figure 3.2 shows a sample distribution of nodes for an experiment carried out in room 1. All marks represent nodes (their total number is 23 in this case). In one interpretation, the 4 solid circles (in the corners) were pegs, while the 11 crosses (both transparent and solid) provided profiled samples. The asterisks acted as tags whose locations were to be determined.

3.5 The Impact of Parameters on the Performance of LEMON

The primary goal of this (initial) series of experiments was to get an insight into the proper settings of the various parameters of LEMON and see how the characteristics of the profiled data (the number of pegs, the number of samples) affect the accuracy of the localization. One important parameter of the location estimation algorithm is K, i.e., the number of best matched points to be used for coordinate averaging. Rooms 1 and 2 were used for this purpose.

3.5.1 The effect of sample density

Consider the configuration of nodes shown in Figure 3.2. The four corner nodes are used as pegs, while the nodes marked with asterisks are interpreted as tags whose locations are to be estimated. In the first crude experiment, we only use five profiled samples provided by the nodes marked with solid crosses. With $K = 4$ (i.e., four best matching samples out of the five), the average error in estimating the location of the eight tags was 0.73 m. With 11 reference points (all crosses) and $K = 5$, the average error was 0.6 m. Other values of K, i.e., $K = 4$ or $K > 5$ gave worse results, as illustrated in Figure 3.3. Note that the error distance is just the Euclidean distance between the estimated and actual locations. The smaller value of K may not be useful as we may lose required information (location of reference points) for the localization. On the other hand, if K is too big we may consider a reference point far from the estimating tag. Thus the primary concern of this profiling-based localization is to tune K first.

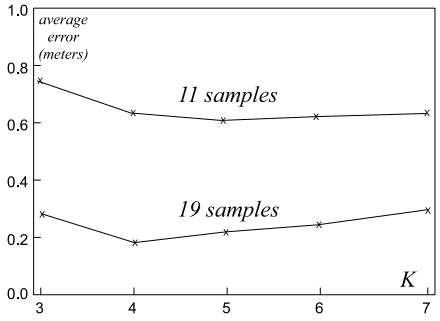


Figure 3.3: Average error distance, room 1.

The set of samples can be augmented by including some readings from the nodes representing tags, which will bring the total number of samples to 19. With those readings included in the database, the quality of location estimation improves rather drastically (see Figure 3.3). This time the best number of K turned out to be four. Owing to the fact that the database contains samples collected from points situated very close (distance zero)

from the estimated locations, using too many points in the averaging formula will have the tendency to pollute the contribution of the true best match, even if that match isn't 100% perfect (because of the statistical fluctuations in RSS). Thus, this simple exercise suggests that dense profiling is likely to improve accuracy and require a lower value of K than a sparse set of samples.

3.5.2 The effect of peg density

In room 2, we set up an 8×8 grid of nodes spaced 1m apart. Initially, we assumed only 4 pegs located in the corners. The remaining 60 nodes of the grid provided signals for the profiled samples. Then we put some nodes in the centers of the grid squares to collect data for estimating their locations. The average error turned out to be around 1.6m, with a not-so-well pronounced minimum (1.56 m) for $K = 6$.

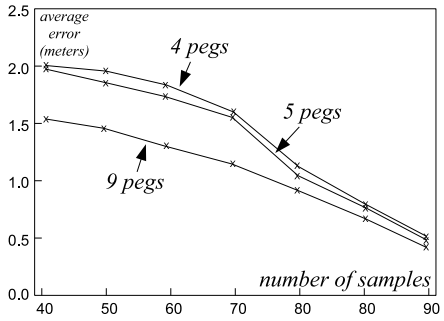


Figure 3.4: Experiment in room 2.

There are two obvious ways to try to reduce that error. The fact that it is significantly higher than in the previous case clearly results from the much larger area being covered by the same (small) number of pegs. Consequently, increasing the number of pegs is one obvious idea, while another approach is to increase the number of profiled samples. Figure 3.4 shows the impact of these two factors on the observed error. Suppose that the grid

points are denoted $< u, v >$, where $u, v = 0, \ldots, 7$. Thus, the original four pegs had coordinates $< 0,0 >$, $< 0,7 >$, $< 7,0 >$, and $< 7,7 >$. The case with 5 pegs includes one extra peg at location $< 3,3 >$, and the additional four pegs for the 9-peg scenario are $< 4,0 >$, $< 7,4 >$, $< 3,7 >$, $< 0,3 >$. The extra samples were collected in such a way as to make their distribution as uniform as possible within the confines of the discrete grid.

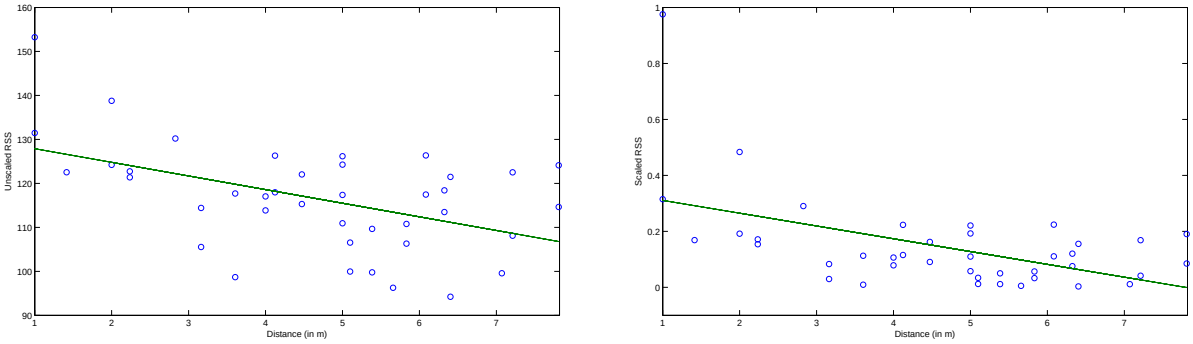
3.5.3 RSS scaling

In one particular experiment, the localized tag was carried by a person (both for the profiling as well as for the localization) to determine the impact of the tag's proximity to a human body (e.g., in people tracking applications). Initially, the results were perceptibly worse than those obtained under the more sterile conditions of the previous environment. The analysis of data revealed that most of the problems resulted from assigning too much relevance to low RSS values, i.e., corresponding to weak reception, which would exhibit large statistical fluctuations. With the linear factoring of the distance (closeness) of the requisite RSS vectors into the coordinate averaging formulas (Section 3.3), such a fluctuation, additionally amplified by the presence of a human body, may cause the coordinates of a distant reference point to affect the estimation in a manner disproportionate to its *true* proximity to the tag.

The numerical RSS readings presented by the RF module of EMSPCC11 are positive numbers, roughly between 84 and 154, representing (numerically) a shifted dBm signal level of the received packet. Considering that the weakest RSS reading is in the ballpark of 84, which is more than half of the entire scale, it made sense to re-scale the RSS values into a range where the weakest readings become zero. The following rescaling formula, in addition to accomplishing the above objective, also allows us to control the impact of RSS values with a different magnitude:

$$R_s = \left(\frac{r - MIN}{MAX - MIN} \right)^{\gamma} ,$$

MIN and *MAX* represent the minimum and maximum obtainable readings (84, 154) across all sets of measurements, and $\gamma > 1$ is the amplification factor. Note that for $\gamma = 1$ we obtain a straightforward linear transformation of the RSS readings into the normalized range $[0, 1]$. Figure 3.5 shows the an example of the RSS scaling effect as measured from a tag at particular distances from peg, where the y axis of the unscaled data spans from 94 to 153 corresponding to power levels from -91 to -61 dBm and the right side of the figure shows how it is transformed with scaling. Note that for a particular experiment we obtain a pair of MIN and MAX that is used for scaling. For instance, the above is the pair for a single room experiments. In the case of three room experiments, our MIN and MAX pair is (58, 192).



Unscaled RSS **Scaled RSS**

Figure 3.5: The effect of RSS scaling.

The best results have been observed for $\gamma = 3$ and $K = 4$. Note that in the remaining single room experiments, the results reported are for those same values of K and γ. In 90% of the cases we were able to estimate the tag location with an error less than 1m. Even though this step alone does not yet maximize the accuracy of LEMON, it already challenges the localization accuracy reported in [45], which was about 2 m using an even finer grid than ours.

3.5.4 The effect of RSS metrics and averaging bias

As explained in Section 3.3, LEMON estimates the location of a tracked tag by first selecting a number of *close* samples from the database, and then averaging the coordinates of the respective reference points into the estimated coordinates of the tag. The notion of closeness assumes some metric in the space of vectors consisting of (transformed) RSS readings. The most natural choice for this metric is straightforward Euclidean distance; however, we have also tested other candidates listed in Table 3.1.

METRIC	FORMULA
Euclidean	$\sqrt{\sum_{j=1}^{m}(R_\Omega(j)-R_\Psi(j))^2}$
Manhattan	$\sum_{j=1}^{m}\lvert R_\Omega(j)-R_\Psi(j)\rvert$
Supreme	$\max_{1\le j\le m}\lvert R_\Omega(j)-R_\Psi(j)\rvert$
Lorentzian	$\sum_{j=1}^{m}\ln(1+\lvert R_\Omega(j)-R_\Psi(j)\rvert)$

Table 3.1: RSS vector proximity metrics.

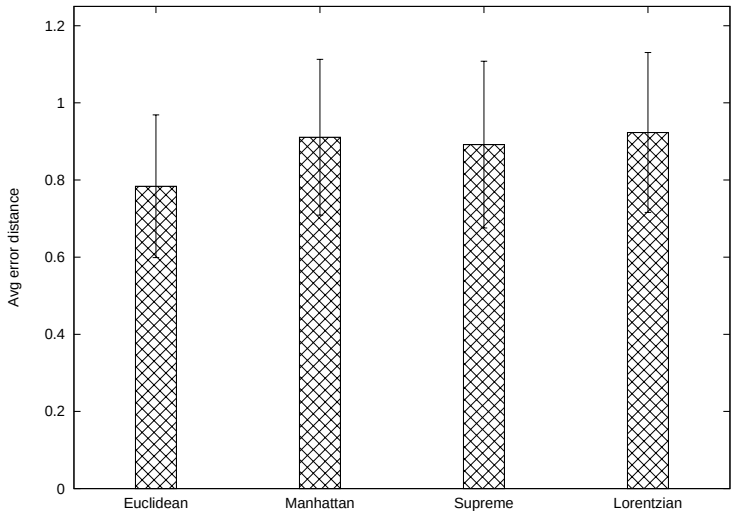


Figure 3.6: Effect of distance metrics on location estimation.

The observed difference in performance was not very pronounced. For example, in

one experiment aimed at assessing the impact of different metrics, we used room 3 (7×7 grid) with 1m×1m cell size. 40 reference points were collected to estimate the locations of 36 tags. The average error observed under different sample selection metrics is shown in Figure 3.6. From now on all the average error results that are presented in this work consider 95% confidence interval. The best performance (the smallest error) was observed for the Euclidean metric.

In the last step of LEMON's location estimation procedure (Section 3.3), the coordinates of the selected samples are averaged to produce the estimated coordinates of the tag. It is possible to define various averaging formulas in addition to the one we proposed in Section 3.3 (2). In the following paragraphs three such averaging techniques are outlined.

$$x_e = \frac{\sum_{i=1}^{K} w_i \times (\sum^* x_j)}{(K-1)},\ y_e = \frac{\sum_{i=1}^{K} w_i \times (\sum^* y_j)}{(K-1)}, \tag{3}$$

where $w_i = \frac{d_i}{D}$ and $(\sum^* x_j)$ is the sum of all K x_j coordinates except x_i. Note the difference between formulas 2 and 3 (we call it LEMON2). Let S be a database sample and (x_i, y_i) be the coordinates of its reference point. Let d_S be the distance of the RSS vector of S from the RSS vector of the tag's readings. The first formula includes (x_i, y_i) in the averaged estimate with a factor directly equal to the difference between d_S and the maximum d over all samples selected for the averaging (d_{max}). In particular, the sample with $d_S = d_{max}$ will be completely ignored from the final result (its factor will be zero). The sole purpose of that sample is to provide the maximum value of distance, i.e., d_{max} to be used for weighting the contribution of reference points from the remaining samples. On the other hand, with formula (3), all the coordinates of all selected samples contribute to the final average. The relative distance of a sample is used as factor weighting the contribution of all other samples, i.e., the further the sample's RSS vector is from the tag's readings, the more all the remaining samples should count in the final estimate.

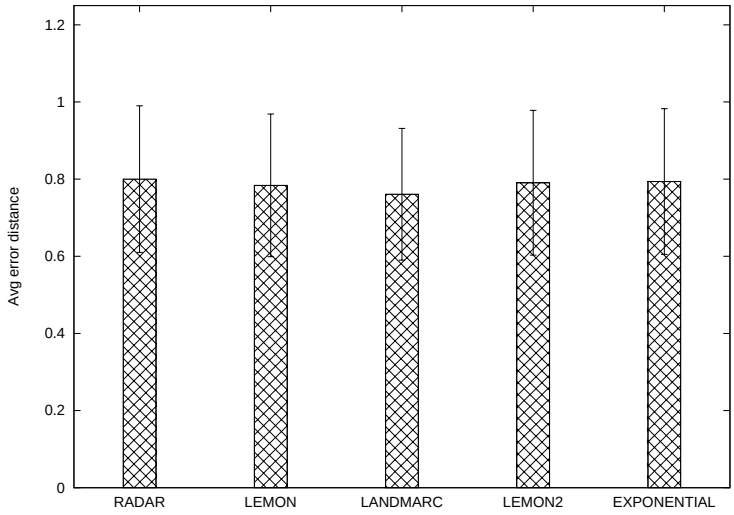


Figure 3.7: Effect of averaging factors.

In the LANDMARC paper [45], the authors suggest this averaging formula:

$$(x_e, y_e) = \sum_{i=1}^{K} w_i (x_i, y_i), where\ w_i = \frac{1/d_i^2}{\sum_{i=1}^{K} 1/d_i^2} \tag{4}$$

where the distance exponent (here, equal to 2), is in principle a parameter. This formula amplifies the impact of closer samples (and reduces the impact of distant ones) with respect to formulas (2) and (3). Of course, one can apply arbitrary functions to the distance, preferably ones preserving the monotonicity of factors (i.e., closer samples having larger contributions to the average). In particular, this formula proposed in [49] (we name it exponential):

$$(x_e, y_e) = \sum_{i=1}^{K} w_i (x_i, y_i), where\ w_i = \frac{e^{-bd_i}}{\sum_{i=1}^{K} e^{-bd_i}} \tag{5}$$

where $b > 0$, amplifies the distance exponentially.

We have tested all the above formulas, and the differences in performance turned out to be marginal. The results are compared in Figure 3.7. Note that RADAR considers simple

averaging formula. The additional *trivial* case corresponds to straightforward averaging, with the same factor of $1/K$ applied to all samples. The best performance (the smallest overall error) was seen for the LANDMARC formula, with the improvement of order $2-4\%$.

3.5.5 The effect of obstacles

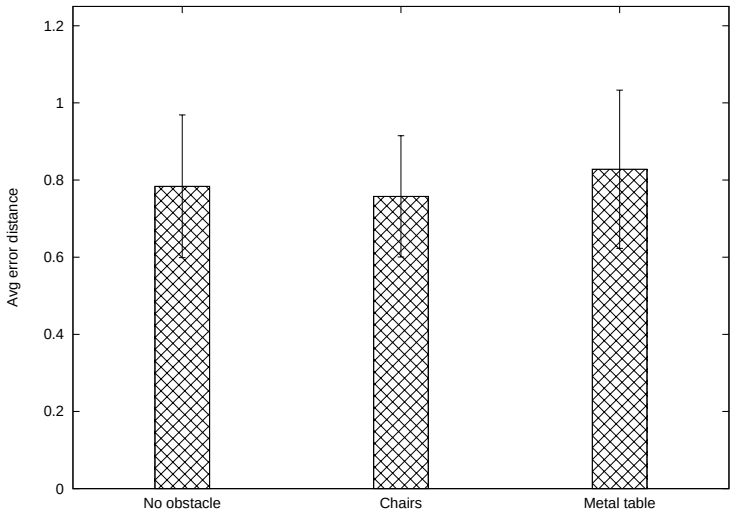


Figure 3.8: Effect of obstacles.

In this series of experiments we studied the impact of obstacles placed in the monitored area. We consider three scenarios deployed in room 3 (7×7 grid):

1. no obstacles
2. four stacks of chairs placed symmetrically in the four centers of the grid quadrants
3. a large table made of wood and steel placed in the center, with a large metal object placed on top of the table

The profiling stage was carried out in an empty room. The idea was to see the performance of our localization scheme in a situation when the layout of the monitored area has been

changed (perhaps quite drastically) after the profiling. In particular, one would like to know in what circumstances the profiling data should be deemed obsolete, calling for a new profiling stage.

Somewhat surprisingly, the disturbance caused by the obstacles turned out to be quite minimal (see Figure 3.8). In particular, the piles of chairs caused absolutely no reduction in accuracy.[4] The impact of the table (including a significant amount of metal) was more perceptible and resulted in a 3.8% increase in the average error.

3.6 Effect of Profiled Samples and Infrastructure Nodes on the Performance of LEMON

Indoors, the presence of multi-path introduces noise to the measured RSS. Thus we would need to collect a large number of reference points to counter the imprecise nature of the measurements. This brings up the issue of determining the smallest number of reference points that are sufficient for a given accuracy of localization. Each reference point measurement could be considered an overhead, since a user (profiler) would have to perform it and associate it with actual coordinates. Whenever the environment changes drastically, e.g., when furniture are rearranged or changed, re-measurement would be necessary. Therefore, the smaller the number of required reference points, the better. In this section we present experimental results to illustrate the impact of the arrangement and number of profiled samples on the localization performance. Indeed, we perform the same analysis for the infrastructure nodes, pegs.

3.6.1 Impact of number and layout of reference points

We compare LEMON against two profiling-based localization schemes, LANDMARC [45] and RADAR [3]. Specifically, we investigate the effect of the number of reference points

[4] We could even see a slight improvement which has to be attributed to a statistical fluctuation.

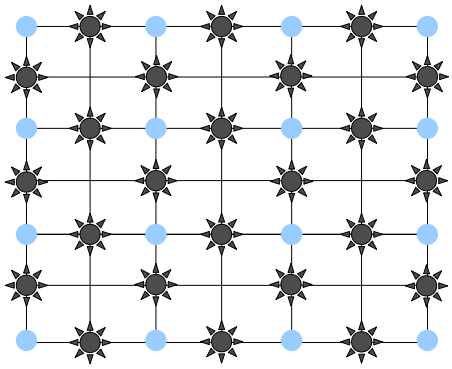


Figure 3.9: Locations of pegs and of 24 reference points (Diamond).

and their arrangement on the localization error. For this purpose, we gathered 49 reference points, i.e., from *every* grid point of a 7×7 grid, where the pegs were positioned at 16 locations (depicted as circles in Figure 3.9). When a reference point measurement was at the same location as a peg, the actual measurement was taken 6 cm away from the peg. We then removed reference points from the database in a regular fashion such that the density of reference points remained roughly the same across the grid. Figure 3.9 shows the layout of reference points after removing 25 points; the resulting setup is dubbed *Diamond* and consists of measurements from 24 reference points. By further removing points in the same fashion, i.e., one at a time, we arrive at two new layouts (see Figure 3.10) called, respectively, *Hexagon* (18 points) and *Nested-Hexagon-Diamond* (12 points). Starting with the Hexagon layout and eliminating more points, we produce two additional layouts dubbed *DoubleD* (14 points) and *Square* (12 points), respectively.

The average error in location estimation for LEMON, LANDMARC, and RADAR in each of the above layouts is shown in Figure 3.11. The average error for LEMON with all the initial 49 reference points was 0.37 m, but we could remove points in a regular fashion and end up with 12 points while the error distance was still less than 1 m. This indicates that it is possible to have a less complex deployment of LEMON without degrading the performance significantly (see, e.g., the case of Dia(24)). Overall, we have observed that the more

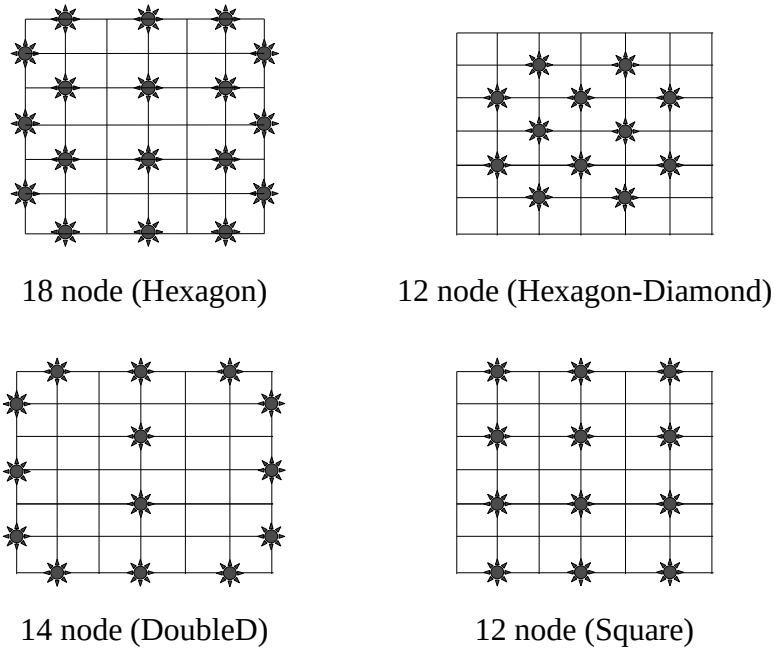


Figure 3.10: Reference point location configurations.

the reference points the better the localization. Yet, the number of reference points alone is not sufficient, as their placement matters as well (see for example the case of DD(14) vs. Square(12)). Moreover, LEMON is less sensitive to the number of reference points and their layout compared to LANDMARC and RADAR. In fact, of all three, RADAR appears to be the one scheme that deteriorates the most when the number of reference points is reduced. This may happen as RADAR uses simple averaging technique without considering the discrepancy relationship between the query and the profiled RSS; hence, sparse collection of profiled samples may affect the estimation accuracy.

3.6.2 Effect of peg placement

The unique feature of LEMON is its flexibility of using as many pegs as needed without imposing any restriction on their number. However, there might exist better or worse ways of placing those pegs that may offer better or worse estimation. Finding the best placement of pegs for a target localization accuracy may pose a challenge. We may not need, say,

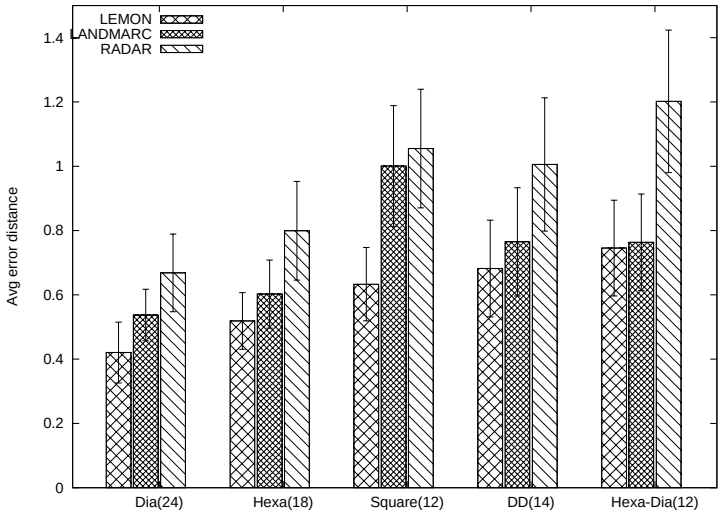
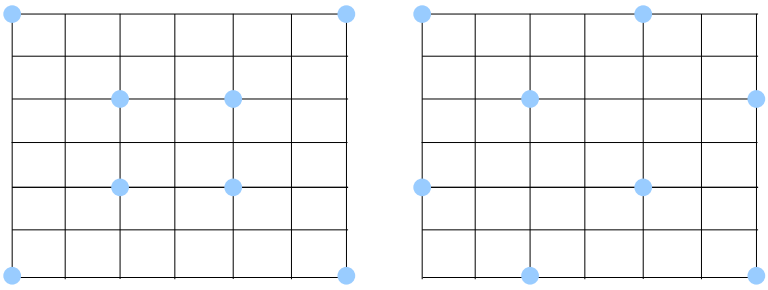


Figure 3.11: Localization error for different reference point configurations.



Nested-Square ZigZag

Figure 3.12: Peg layouts for eight pegs.

16 pegs for 7×7 grid to keep the error distance below 1 m. Thus we conducted another experiment to try to produce configurations with fewer pegs that still provide a localization error less than 1 m. Figure 3.12 shows two such layouts with 8 pegs and the corresponding average localization error is shown in Figure 3.13. As we can see, it is possible to eliminate half of the pegs and still maintain error distance well below 1 m. We also found that removing just four pegs (the "center square" of circles in Figure 3.9) would leave 12 pegs (called "Square" in Figure 3.13) but somewhat higher localization error than the configurations

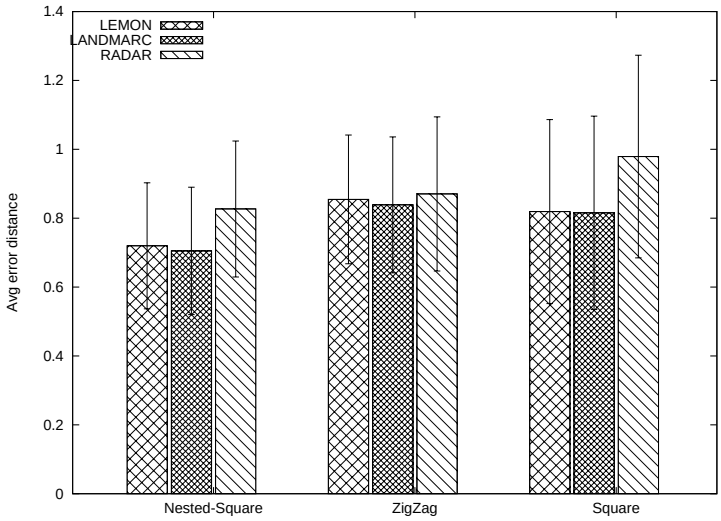


Figure 3.13: Localization error for different peg layouts.

with the fewer but "better placed" pegs. This may suggest that a clever design of placing pegs on the target area may help us reduce the error distance. Our observations indicate that removing pegs from inside the grid may not be a good idea. Yet, what is even more interesting to note is that all three schemes (LEMON, LANDMARC, and RADAR) are relatively insensitive to the peg layout, compared to the impact that the layout (and number) of reference points could have. Additionally, LANDMARC appears to have a slightly better performance than the other two, but the statistical significance of the differences is very small. The lesson is that peg placement is not as critical an issue as the dense "sampling" of the space by many reference points. This is good news, as the placement of pegs is likely to be constrained or even dictated by external factors, e.g., placement of furniture, walls, etc.

3.7 Performance of LEMON in Multiple Rooms

Experiments in this series were carried out in three adjacent rooms, 3m $\times$ 5m each, with almost identical layouts shown in Figure 3.14. Each room included four wooden tables and

chairs, a metal file cabinet, and two desktop computers.

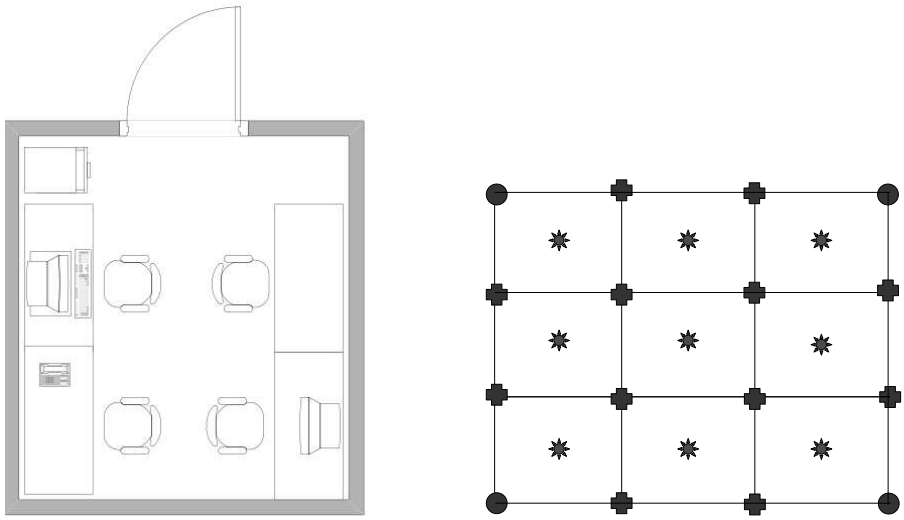


Figure 3.14: Room layout and layout of pegs (circles), reference points (crosses), and localized points (stars).

We placed four pegs at the corners of each room and created a grid with the edge size of 1m. Twelve profile samples were collected in each room at the reference points indicated by the crosses in Figure 3.14. Then, nine locations from each room (indicated by the stars in Figure 3.14) were estimated.

One of our objectives for this series was to study the impact of node orientation, which was reported in [3] to impact localization accuracy. To verify that observation, we collected readings in four different directions (North, South, East, and West) from every profiled reference point, and also noted the tag's orientation during localization. Then we compared the average estimation error for two scenarios:

1. The orientation attribute is ignored. This means that all the profiled samples from the database are applied for the localization of a tag, and the tag's orientation does not affect their selection.

2. The orientation of the tag is reported, and only those samples from the database that

were collected under the same orientation are used for localization.

Despite the fact that the line-of-sight (LOS) of the signal is never affected by the node's orientation (and one would assume that it should be thus irrelevant), we noticed a small, albeit clearly pronounced, improvement when the orientation attribute was taken into account (see Figure 3.15). The observed difference in average error was about 12cm. While not huge, it crisply illustrates how an even minor change of tag placement (namely rotation on the same spot) may influence the delicate procedure of localization. For one thing, the 12cm of difference can be viewed as a lower bound for the resolution of any localization scheme that fails to incorporate orientation. This kind of resolution is likely to be more than acceptable in most practical applications. We should note that in many cases the orientation attribute is not discrete and it cannot be acquired reliably without rather expensive and cumbersome equipment. On the other hand, the 12cm bound is probably not easily reachable, because of other factors.

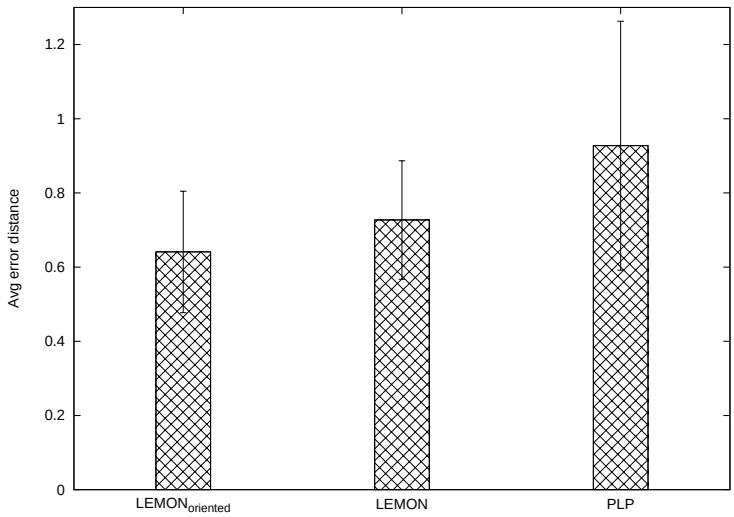


Figure 3.15: Performance of LEMON (w/ & w/o orientation) and k-NN as used by PLP.

Another subject of this study was a comparison of LEMON against the localization scheme reported in [52] and based on the K–Nearest Neighbor algorithm. That approach,

called Power Line Positioning (PLP), also resorts to profiling, but it uses power lines as the means to distribute RF signals throughout a building. Apart from the unconventional use of the powerline as an "antenna," PLP's approach is to apply two different neighbor selection criteria (corresponding to the value of K in LEMON). Specifically, the scheme uses a larger K (a larger number of nearest neighbors) to first identify the room where the tracked tag is located. The reference points from the identified room are then used with a smaller K to estimate the tag's location within the room. As one of our objectives was to investigate the accuracy of room identification in LEMON, we tested the same logic in our scheme. That was easy, as both solutions employ essentially the same concept of K closest neighbors.

Our two values of K were 6 (for the room identification) and 4 for the subsequent localization within the room. The results show that the estimation accuracy of PLP degrades as it subdivides the signal space according to the physical layout, and then proceeds to localize within a selected sub-area. There is an apparent loss of information when we first identify the room and then localize strictly within that room, resulting in the large error distance shown in Figure 3.15, even against LEMON with no orientation information. LEMON clearly outperforms PLP when we consider the three rooms as one signal space and use all the reference points to perform the estimation. This demonstrates that the intuition of restricting the localization to a subset of reference points that might appear "close" results in a loss of data that could be crucial to localization, i.e., reference points in adjacent rooms *do matter*.

Finally, Figure 3.16 illustrates the distribution of errors in the three-room scenario, collected for $K = 4$. The x-coordinate spans all three rooms, y-coordinate is presented by the y-axis, and the z-axis is to present the estimated error values. The red areas are corresponding to the highest values of error, whereas, the blue areas have the lowest error. All the error distributions presented in this thesis follow the same description. It is interesting to see the highly non-random nature of those errors, with clearly visible and structured "bad spots"

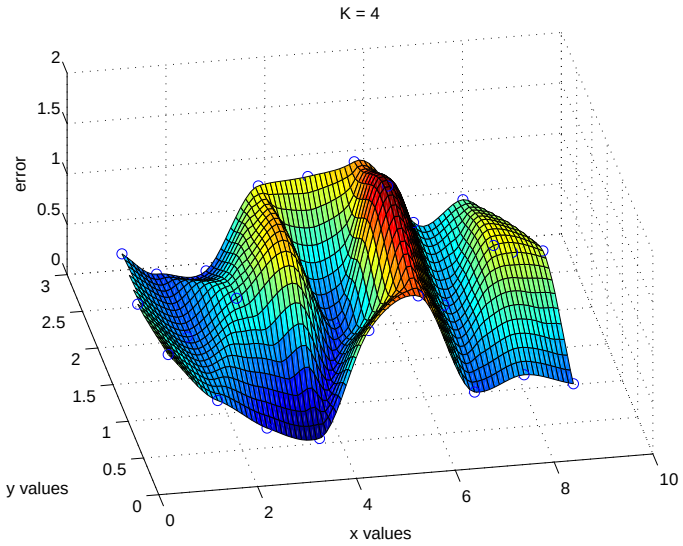


Figure 3.16: Distribution of errors in the three-room experiment.

and gradients (specifically near the walls that are at x=3 and x=6). This suggests that a systematic approach to correcting those errors may be possible, by exploring the correlations so vividly present in the surveyed area.

3.8 Conclusions

We have presented a system for RF-based localization within covered (indoor) areas. In this chapter, we outlined various factors, including the number of nearest neighbors, the number and arrangement of profiled samples and pegs, the RSS scaling, the RSS discrepancy and averaging techniques, the obstacles, the orientation of device, and the partitioning of radio map, on the performance of LEMON. Our experiments indicate that LEMON can easily achieve a practical average localization error of less than a meter in all scenarios examined in this chapter. The results place it as being both more accurate and simpler than competing techniques. The uniformity and low cost of the equipment (even our underfunded lab could afford a few hundred of Olsonet nodes) makes LEMON a highly viable and practical

solution.

It is not easy to meaningfully compare the different localization schemes proposed in the literature, mostly due to the difference in equipment (trading off cost for accuracy), the logistics of testing (e.g., area size, multiple rooms, obstacles), data gathering techniques, and the environment [71]. Nevertheless, we attempted to the extent possible to provide a fair basis of comparison against other schemes. Note that while the average accuracy of $1m$ has to be considered the worst case in our experiments, it is still better than what has been reported for other schemes [3, 10, 29, 45, 65].

One topic for further study is the impact of a possible diversification of the *class* attribute of a sample (see Section 3.3) on the accuracy of localization by LEMON. So far, all our experiments were carried out under the same identical options of the transceiver amounting to a single signal class. The observation that lower values of RSS tend to be confusing to the sample selection algorithm, as well as to the averaging formula, hints at the possibility of using two or more transmission power levels at the tag. A strong signal will confuse nearby pegs (their readings will not be reliable), but some further pegs may then offer assistance in interpreting the reading. This effectively calls for re-scaling the RSS readings at both ends of the range and using multiple matches (within different classes of samples) to arrive at a better approximation of the tag's location.

Another possible way to diversify the signal class would be to use different channels, which could mitigate the problem of multi-path propagation. That, however, is more complicated than diversifying the power level (within the same channel) as it requires the pegs to be able to receive at different frequencies at the same time. One possible solution would be a channel hopping scheme carried out automatically by pegs and tags according to some timed pattern. On the other hand, owing to the low cost of our hardware, it is perfectly conceivable to replicate the peg infrastructure by deploying multiple versions of pegs, tuned to different reception frequencies, at the same place.

The distribution of localization errors depicted in Figure 3.16 provides much food for

thought and hints at the possibility of correcting those errors systematically, e.g., based on some second-level profiling. By accepting reports about errors received during the operation stage, the scheme could build their model (e.g., akin to [65]) and attempt to offset its location estimation in a methodological fashion. There is certainly a lot of potential in LEMON for learning and fine tuning as the system attends to its duties.

Chapter 4

Transforming Samples into Location Estimates: A Performance Comparison

In the previous chapter, a series of experimental results for LEMON are presented. Initial results are quite promising as LEMON guarantees good accuracy in all practically interesting scenarios and also prevails, in terms of localization accuracy, against the state-of-the-art based on profiling. Thus we next focus on analyzing different localization methods based on profiling, including *lateration*, *Bayesian Networks*, *Maximum Likelihood Estimation (MLE)*, *Gaussian Process (GP)*, and *Support Vector Machine (SVM)*. Note that these are well known methods of estimation, thus for the completeness of the comparison we consider all these methods and observe their behavior. In particular, we apply these existing localization methods in our experimental data. Some of the techniques, e.g., the Bayesian Network, require also a proper model definition to fit their particular assumptions. In addition, we propose a new localization method dubbed *combinatorial localization*. The necessary parameters of each of these methods are outlined in the corresponding subsection.

In the previous chapter we mentioned that the lateration based on distance prediction

from a channel model is not a good choice for localization. To verify our claim we perform lateration and compare the results against LEMON. We also examine probabilistic schemes, and Bayesian Networks in particular, which assume that the signal distribution is Gaussian and train the network to learn such distribution from the profiled samples. To estimate the location of a tag, probability (weight) of each profiled sample is determined to generate the weighted average of coordinates of those samples. Finally, we consider an MLE-based scheme where only the most (highest) probable profiled sample is reported as the predicted location, as well as GP and SVM alternatives. These schemes are worse performing (but to varying degree) compared to LEMON.

For the purposes of this chapter, we also propose a combinatorial variety of a localization method, where $\binom{S}{K}$ sets of K nearest neighbors are generated out of S profiled samples. Each one of these sets provides an intermediate estimation. The final estimation is computed as the average of those estimates across all sets. This method can be thought of as an "exhaustive" method of exploring all possible KNN localizations for a given K and hence as yet another means to evaluate the impact of the chosen K. We found that instead of considering all S samples, a subset of them is actually more useful. In particular we may choose the closest $s \subset S$ profiled samples and apply the above trick to estimate the location. Nevertheless, the "exhaustive" exploration is useful to learn about K and its impact on localization.

4.1 Transforming Samples into Locations

Given a database of RSS readings from known locations, one can think of several ways of using those readings to estimate unknown locations of tags based on their dynamic readings. In the following subsection different methods of localization, other than KNN, are presented. Before we start to describe the methods, we would like to define the notation used throughout the description of these methods. Let $S \in \mathbb{R}^{n \times m}$ denotes the collection

of profiled samples, which is a $n \times m$ matrix, i.e., there are n profiled samples each composed of a m dimensional RSS vector produced across m pegs in the monitored area. Also, $s_1 \in \mathbb{R}^{1\times m}$ and $s_2 \in \mathbb{R}^{n\times 1}$ denote a m dimensional row vector and a n dimensional column vector, respectively. Thus the former one represents a profiled sample across all the pegs while the latter one stands for all the profiled samples across a particular peg.

4.1.1 Lateration

The idea of lateration is to estimate the physical distance between the infrastructure nodes (pegs) and the tag by using a path-loss model on the RSS. Once such distances are known a system of non-linear equations is solved to estimate the location of the tag. Thus the localization could be thought of as two-step process : *distance measurement* and *location estimation*. We follow the existing solution of lateration [38, 56] and apply it to our experimental data.

Distance measurement

To compute the distance between pegs and a tag, a path loss model defined in [56] is used, which can be described as

$$L = \frac{P_r}{P_t} \tag{6}$$

$$L[\text{dB}] = C - 10\alpha\log(d) + X_\sigma \tag{7}$$

where P_r and P_t are receive and transmit power, respectively. d is the distance between the transmitter (tag) and receiver (pegs) expressed in units of distance d_0 (e.g. meters). The term C accounts for the frequency dependent component and antenna gains while X_σ is a Gaussian noise component. The α is known as the *path loss exponent*.

Usually, we consider the mean of the Gaussian noise to be zero (trivially, if it is not

zero it can be accounted for in C). What we are generally interested in is the average loss, which is:

$$\bar{L}[\text{dB}] = C - 10\alpha\log(d) \tag{8}$$

thus the path-loss could be described as a straight line equation with respect to $\log(d)$. Note that the variance of the X_σ is significant, so this straight line, while approximating the average behavior could be significantly over/under-estimating the behavior of particular samples. Simple least-square fitting can be applied to determine the path-loss exponent α (as well as C). This can be done based on the profiled samples as we know the actual distance and other parameters for the fitting to determine α. Once α is known, it can be used to estimate the distance between the tag and the pegs using the RSS (for the details of path-loss models please check [56]).

In our localization experiment, we use profiled samples to learn C and α based on equation (8). However, we define two kinds of parameters namely, *local* and *global*. The former approach includes a pair of parameters for every peg in the system, i.e., 12 pairs of parameters are learned for 12 pegs. In particular, for each peg p_i, where $i = 1\ to\ m$, we use an $s_2 \in \mathbb{R}^{n\times 1}$ column vector as the corresponding RSS values to compute the parameters. The distances are calculated using the respective parameters as well. The latter option deals with only one pair of parameters for all the pegs by considering $S \in \mathbb{R}^{n\times m}$ and uses them to compute the distance. Table 4.1 shows all the learned local and global parameters.

Location estimation

This second step of lateration is defined in [38]. Let the predicted distance between the tag and a peg be d_j ($j \in \{1,\ldots,n\}$) and (x_j, y_j) be the coordinates of peg p_i. Let us further assume that (x_e, y_e) is the estimated location of the tag. We may then construct the following system of equations:

Methods	**C**	α
LOCAL	−59.57	3.70
	−63.10	3.54
	−66.66	3.81
	−75.45	2.12
	−74.92	1.46
	−77.49	1.85
	−62.59	3.39
	−79.94	1.80
	−70.55	3.06
	−83.48	1.94
	−66.81	3.13
	−68.59	3.54
GLOBAL	−72.88	2.47

Table 4.1: Local and global parameters.

$$(x_1 - x_e)^2 + (y_1 - y_e)^2 = d_1^2$$

.....

.....

$$(x_n - x_e)^2 + (y_n - y_e)^2 = d_n^2$$

This system of nonlinear equations can be linearized by subtracting the last equation from the first $(n-1)$ equations to produce (see [38] for detail):

$$x_1^2 - x_n^2 - 2(x_1 - x_n)x_e + y_1^2 - y_n^2 - 2(y_1 - y_n)y_e = d_1^2 - d_n^2$$

.....

.....

$$x_{n-1}^2 - x_n^2 - 2(x_{n-1} - x_n)x_e + y_{n-1}^2 - y_n^2 - 2(y_{n-1} - y_n)y_e = d_{n-1}^2 - d_n^2$$

This system of linear equations can be expressed as:

$$\mathbf{Ax} = \mathbf{b}, \tag{9}$$

where

$$\mathbf{A} = \begin{bmatrix} 2(x_1 - x_n) & 2(y_1 - y_n) \\ \cdots & \cdots \\ \cdots & \cdots \\ 2(x_{n-1} - x_n) & 2(y_{n-1} - y_n) \end{bmatrix} \tag{10}$$

$$\mathbf{b} = \begin{bmatrix} x_1^2 - x_n^2 + y_1^2 - y_n^2 + d_n^2 - d_1^2 \\ \cdots \\ \cdots \\ x_{n-1}^2 - x_n^2 + y_{n-1}^2 - y_n^2 + d_n^2 - d_{n-1}^2 \end{bmatrix} \tag{11}$$

The solution x (the estimated location of the tag), can then be written as:

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{12}$$

Let us consider an example to illustrate the localization procedure of lateration. Assume we have 4 pegs p_1, p_2, p_3, p_4 with coordinates $(0,0),(0,4),(3,0),(3,4)$. We know the learned global parameters $(C,\alpha) = (-72.88, 2.47)$. We would like to estimate the location, (x_e, x_e), of the tag T, with the RSS association list $(-54,-74,-81,-84)$.

The first step would be to compute the distances between T and the 4 pegs using equation 8. The corresponding distances are $(0.2, 1.1, 2.2, 3.0)$. We may now generate the following equations:

$$\begin{aligned} (0-x_e)^2 + (0-y_e)^2 &= (0.2)^2 \\ (0-x_e)^2 + (4-y_e)^2 &= (1.1)^2 \\ (3-x_e)^2 + (0-y_e)^2 &= (2.2)^2 \\ (3-x_e)^2 + (4-y_e)^2 &= (3.0)^2 \end{aligned}$$

By following equations 10 and 11, A and b can be computed as:

$$\mathbf{A} = \begin{bmatrix} -6 & -8 \\ -6 & 0 \\ 0 & -8 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} -16.0400 \\ -1.2100 \\ -11.8400 \end{bmatrix}$$

Finally using equation 12 the solution **x** of equation 9 is obtained, which is the estimated location $(x_e, y_e) = (0.3678, 1.6046)$. Note that the procedure of localization using local parameters is the same except that the distance between a peg and the tag is estimated using differently determined corresponding parameters.

4.1.2 The Bayesian Network Approach

A *Bayesian Network* encodes the joint probability distribution of a set of k variables, $X_1, X_2, ..., X_k$, as a directed acyclic graph (DAG) and a set of conditional probability distributions (CPDs) [41, 59]. In this graph, each node corresponds to exactly one of the system variables, while the edges denote correlations. In particular, the CPD associated with a node gives the probability of each state of the variable given every possible combination of states of its parents. The set of parents of X_l, say π_l, consists of those values that are connected with X_l by an arc. The edges are placed in such a way that, given the values of its parents, a node is conditionally independent of all the other variables in the system. The joint distribution of the variables is thus given by

$$P(X_1, ..., X_k) = \Pi_{l=1}^{k} P(X_l | \pi_l) \tag{13}$$

In *naive Bayesian* models, all the variables X_l are assumed to be mutually independent given a variable *CE* [41, 59]. The joint distribution of the variables then can be given by

$$P(CE, X_1, ..., X_k) = P(CE)\Pi_{l=1}^{k} P(X_l|CE) \tag{14}$$

Naive Bayesian models can be considered as Bayesian Networks where each variable X_l has the parent CE, which does not have any parents. We follow the above definitions and define the joint distribution of the variables (corresponding to the location and the RSS across the pegs) as described bellow.

In our localization problem, the variables can be defined as the set $\mathcal{X} = \{X, Y, r_1, r_2, \ldots, r_m\}$. X and Y are random variables associated with the location and r_i are variables corresponding to the RSS readings at the m pegs. We may say that the RSS reading at a peg is a function of its distance to the transmitter, which could be captured with the distribution $P(r_i|X,Y)$. Furthermore, we can say that once we know the position of the transmitter the probability of observing a specific RSS at peg i, is independent of the readings at the other pegs. Note that this is not always true as peg readings can be correlated by some local effects, such as both being next to a metal fence etc. However, this correlation does not invalidate the Bayesian Network methodology. Thus we define the joint distribution as follows;

$$P(\mathcal{X}) = P(X,Y)\prod_{i=1}^{m} P(r_i|X,Y) \tag{15}$$

If we make the further assumption that the X and Y coordinates are independently distributed (if they are correlated we can still use the chain rule to decompose the distribution to $P(X,Y) = P(X|Y)P(Y)$), we can also break down their marginal distribution $P(X,Y) = P(X)P(Y)$. Thus we may define a DAG shown in Figure 4.1. If we dropped the independence assumption of the X and Y coordinates this would force us to add an arc between them.

Given the grid nature of the locations we treat X and Y as discrete random variables in our model, but we model RSS data as Gaussian random variables. In particular, at any

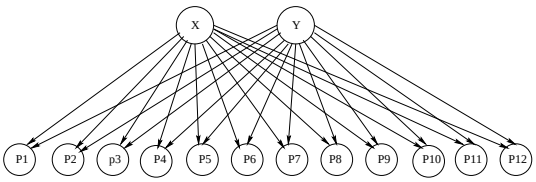


Figure 4.1: DAG representation of the Bayesian network model.

particular node in the DAG we store a table with the probability of the variable taking each of the specific values. To store the Gaussian models we fit a distribution of the form $P(r_i|X,Y) \sim N(\mu_{x,y},\sigma^2)$, which means that for each assignment to its discrete parents we get a different mean. Though the number of readings per sample (5 readings were taken for each profiled and testing samples) might be sufficient to estimate the mean of a distribution, it might not be reasonable to estimate its variance, especially when all 5 data points are pretty much centered on the mean (i.e. almost constant). Thus it would be reasonable to assume that all pegs have the same variance and treat this as a tuning parameter that we can optimize on. In particular, we can learn a global variance based on the profiled samples that is applicable to all distributions.

Once we have our model and distributions, we are ready to predict the location of the tag. The association list of that tag could be used to compute $P(\mathcal{X})$ for every profile samples. The weighted average of all the samples thus could be defined as;

$$(x_e,y_e) = E(P(\mathcal{X})|r_i) \textit{ where } i = 1 \textit{ to } m \tag{16}$$

Suppose that we have 4 profiled samples $(1,1)$, $(2,1)$, $(1,2)$, and $(2,2)$ and using our model above we obtain the probabilities as 0.6, 0.5, 0.8, and 0.4 for the query tag T. Thus the estimated location of T based on our model would be $(1.4,1.5)$.

4.1.3 Maximum Likelihood Estimation

With the MLE approach defined in [59, 60], we estimate the tag's location as the single profiled sample (x_j, y_j) that maximizes $P(\mathcal{X})$. If we assume that all (x_j, y_j) are equally likely and the pegs are independent, we shall obtain this way equation (15). However, instead of taking the weighted average we report here the most likely profile point as the estimated location. Note that when we have no knowledge about prior probability, we may use MLE. Thus if we consider the example from the previous subsection, the estimated location of the tag would be $(1,2)$ as this profile point maximizes $P(\mathcal{X})$.

4.1.4 Gaussian process

The *Gaussian Process (GP)* is a non-linear regression tool, where the non-linear "kernel trick" is used to project the input space to the higher dimensional feature space and perform the linear regression in that feature space. Thus localization could be performed using GP where signal strength and distance have a non-linear relationship. There are two different views of GP; namely *weight-space* and *function-space* and in the following we will present the both views defined in [57].

First we will present the Bayesian analysis of the standard linear regression model with Gaussian noise following the description in [57]:

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{w}, \quad y = f(\mathbf{x}) + \varepsilon \tag{17}$$

where $\mathbf{x} \in \mathbb{R}^m$, $\mathbf{w} \in \mathbb{R}^m$, $f \in \mathbb{R}$, and $y \in \mathbb{R}$ are the input vector, the weight vector, the function value, and the observed target value, respectively. The observed values y are assumed to differ from the function values by an additive noise $\varepsilon \in \mathbb{R}$, which is assumed to be Gaussian distributed with zero mean and variance σ^2, i.e., $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Equation (17) can be rewritten as

$$P(y|\mathbf{X},\mathbf{w}) = \prod_{j=1}^{n} P(y_j|\mathbf{x_j},\mathbf{w}) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y_j - \mathbf{x_j}^T\mathbf{w})}{2\sigma^2})$$
$$= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp(-\frac{1}{2\sigma^2}|\mathbf{y} - \mathbf{X}^T\mathbf{w}|^2) = \mathcal{N}(\mathbf{X}^T\mathbf{w}, \sigma^2\mathbf{I}) \tag{18}$$

where $\mathbf{X} = [X_1, X_2, ...X_n] \in \mathbb{R}^{m\times n}$ is the collection of input vectors, $\mathbf{y} = (y_1, y_2, ..., y_n)^T$, and $\mathbf{I} \in \mathbb{R}^{m\times m}$ is the identity matrix. In Bayesian models, we specify a *prior* over the parameters before looking into the observations; thus, we may assign a $0 \in \mathbb{R}^m$ mean Gaussian prior with the covariance matrix $\Sigma \in \mathbb{R}^{m\times m}$ on the weights, i.e., $\mathbf{w} \sim \mathcal{N}(0, \Sigma)$ [57].

Now, following Bayes' rule, we may say that

$$posterior = \frac{likelihood \times prior}{marginal\ likelihood}, \quad P(\mathbf{w}|\mathbf{y},\mathbf{X}) = \frac{P(\mathbf{y}|\mathbf{X},\mathbf{w})P(\mathbf{w})}{P(\mathbf{y}|\mathbf{X})} \tag{19}$$

the marginal likelihood is independent of the weight and could be considered as a normalizing constant. Thus based on the likelihood and prior and following the calculation in [57], it can be said that

$$P(\mathbf{w}|\mathbf{X},\mathbf{y}) \sim \mathcal{N}(\bar{\mathbf{w}}, \mathbf{A}^{-1}), \quad where$$
$$\bar{\mathbf{w}} = \frac{1}{\sigma^2}\mathbf{A}^{-1}\mathbf{X}\mathbf{y}$$
$$A = \sigma^{-2}\mathbf{X}\mathbf{X}^T + \Sigma^{-1} \tag{20}$$

Thus posterior is Gaussian-distributed with mean $\bar{\mathbf{w}}$ and the covariance matrix $\mathbf{A}^{-1}$.

Now to predict the probability of the function, $\mathbf{f}_* \in \mathbb{R}$, value of a test point $\mathbf{x}_* \in \mathbb{R}^m$, we take the weighted average of all possible parameters values and their corresponding

posterior probabilities, which is

$$\begin{aligned} P(f_*|\mathbf{x}_*,\mathbf{X},\mathbf{y}) &= \int P(f_*|\mathbf{x}_*,\mathbf{w})P(\mathbf{w}|\mathbf{X},\mathbf{y})d\mathbf{w} \\ &= \mathcal{N}(\frac{1}{\sigma^2}\mathbf{x}_*{}^T\mathbf{A}^{-1}\mathbf{X}\mathbf{y}, \mathbf{x}_*{}^T\mathbf{A}^{-1}\mathbf{x}_*) \end{aligned} \tag{21}$$

In the above we use:

$$P(A) = \int P(A,B)dB = \int P(A|B)P(B)dB \tag{22}$$

where $P(A) = P(f_*|\mathbf{x}_*,\mathbf{X},\mathbf{y})$, $P(A|B) = P(f_*|\mathbf{x}_*,\mathbf{w})$, and $P(B) = P(\mathbf{w}|\mathbf{X},\mathbf{y})$. Thus we get the distribution of f_*, the function of a test point $\mathbf{x}_*$, and the mean $(\frac{1}{\sigma^2}\mathbf{x}_*{}^T\mathbf{A}^{-1}\mathbf{X}\mathbf{y})$ of this distribution is the predicted value.

However, a typical problem of the above mentioned regression model, especially in low dimension, is that the assumed linear model may not best fit the data. One possible solution might be to project the, say $m-$dimensional, input space into a high, say $M-$dimensional, *feature space* using a set of basis functions and then apply the above-mentioned linear model on that high dimensional space. Assume that the function $\phi(\mathbf{x})$ maps the input vector to the feature space. Thus the linear model would look like

$$f(\mathbf{x}) = \phi(\mathbf{x})^T\mathbf{w} \tag{23}$$

After some calculation one can notice that the explicit high dimensional feature value of $\phi(\mathbf{x})$ is never used, but rather we only utilize $\phi(\mathbf{x})^T\Sigma\phi(\mathbf{x}') = k(\mathbf{x},\mathbf{x}')$, which is defined as a *covariance matrix* or a *kernel*.

The above description of linear regression considers weight-space, but could also be described (producing the same results) considering distribution over functions, i.e., the

function-space view, which is described bellow.

Definition 1. *A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

Note that in a *Gaussian distribution* the mean and the covariance are a vector and a matrix, respectively, i.e., it is distributed over vectors; whereas, *GP* is distributed over functions.

$$f \sim \mathcal{GP}(mn, k) \tag{24}$$

where the function f is distributed as a *GP* with the *mean function mn* and and the *covariance function k* [57]. It means that in a Gaussian distribution, an individual random variable from a vector is indexed by its position; whereas, in *GP*, a random variable $f(x)$ is indexed by its argument x. Let us consider the example of *GP* in [57], where it is defined by mean $mn(x) = \frac{1}{4}x^2$ and $k(x,x') = \exp(-\frac{1}{2}(x-x')^2)$. Now we may consider n samples from the function f and for the chosen values of x we compute a vector of means and a covariance matrix, these two thus define a regular Gaussian distribution as:

$$\begin{aligned} &\mu_i = mn(x_i) = \frac{1}{4}x_i^2, i = 1,2,....,n \ \ and \\ &\Sigma_{ij} = k(x_i,x_j) = \exp(-\frac{1}{2}(x_i - x_j)^2), i,j = 1,....,n \end{aligned} \tag{25}$$

where mn and k define the process and μ and Σ define the distribution. Now from this distribution we may write

$$\mathbf{f} \sim \mathcal{N}(\mu, \Sigma), \ \ where \ \ \mathbf{f} \in \mathbb{R}^n \tag{26}$$

Now to predict a test point, let $\mathbf{f}$ and $\mathbf{f}_*$ be the function values of the training data and test inputs, $\mathbf{X}_*$, respectively. Their joint distribution can be written as:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} \Sigma & \Sigma_* \\ {\Sigma_*}^T & \Sigma_{**} \end{bmatrix} \right) \tag{27}$$

where μ, μ_*, Σ, Σ_*, and Σ_{**} are the training means, the test means, the training set covariances, the training-test set covariances, and the test set covariances, respectively. The posterior distribution for a test set can be defined as:

$$\mathbf{f}_*|\mathbf{f} \sim \mathcal{N}(\mu_* + {\Sigma_*}^T\Sigma^{-1}(\mathbf{f}-\mu), \Sigma_{**} - {\Sigma_*}^T\Sigma^{-1}\Sigma_*). \tag{28}$$

Note that the above Gaussian distribution considers a finite set of function values. The corresponding posterior process is $\mathcal{GP}(mn_S, k_S)$, where

$$\begin{aligned} mn_S(x) &= mn(x) + \Sigma(\mathbf{X},x)^T\Sigma^{-1}(\mathbf{f}-\mathbf{mn}) \text{ and} \\ k_S(x,x') &= k(x,x') - \Sigma(\mathbf{X},x)^T\Sigma^{-1}\Sigma(\mathbf{X},x') \end{aligned} \tag{29}$$

where $\Sigma(\mathbf{X},x)$ is the vector of covariances between every training case and x. In the presence of noise in the training data, every $f(x)$ has an extra covariance with itself with a magnitude equal to the noise variance [57]. In our localization we follow the above definition of $\mathcal{GP}$ for the localization, where he RBF or Gaussian kernel is defined as $k(x,x') = \exp(-\gamma(x-x')^2)$. We learned two necessary parameters based on the profile data, which are γ and the standard deviation of the Gaussian noise. The values of these parameters are 1.0 and 0.5 in both scaled and unscaled RSS, respectively.

4.1.5 Support Vector Machine (SVM)

SVM, defined in [27, 70], is another choice of non-linear regression, that also maps the input space into a higher dimensional feature space using a kernel function to find the optimal solution. Basically, we use a linear model as follows (being interested in finding **w**):

$$y = f(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + \varepsilon \tag{30}$$

To learn $\mathbf{w}$, we need to solve the following minimization problem

$$min_{\mathbf{w}} \sum_{i=1}^{n} V(y_i - f(x_i)) + \frac{\lambda}{2} \| \mathbf{w} \|^2,$$

where

$$V(u) = \begin{cases} 0, & \text{if} |u| < \varepsilon r \\ |u| - \varepsilon r, & \text{otherwise} \end{cases}$$

Note that the difference between GP and SVM is that in the latter method, we have a loss function $V(u)$ and a penalizing term $\frac{\lambda}{2} \| \mathbf{w} \|^2$. Any measurement with error less than εr is considered loss-free, otherwise, the loss is the difference between the predicted value and the residue εr. The above is a quadratic optimization problem to be solved that may need to consider huge dimensional data as we perform the optimization in the feature space. To solve this computational issue, instead of solving this primal problem, its dual problem is used, which is defined using a pair of Lagrange multipliers, (α_i, α_i^*), as

$$min_{\alpha_i, \alpha_i^*} \varepsilon r \sum_{i=1}^{n} (\alpha_i^* + \alpha_i) - \sum_{i=1}^{n} y_i (\alpha_i^* - \alpha_i) + \frac{1}{2} \sum_{i,j=1}^{n} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) \mathbf{x}_i^T \mathbf{x}_j,$$

subject to,

$$\begin{aligned} \Sigma_{i=1}^{n} \alpha_i^* &= \Sigma_{i=1}^{n} \alpha_i \\ \alpha_i^* &\leq 1/\lambda \\ 0 &\leq \alpha_i \end{aligned}$$

Thus we need to determine the Lagrange multipliers and from equation (30) we may write;

$$\mathbf{w} = \sum_{i=1}^{n} (\alpha_i^* - \alpha_i) \mathbf{x}_i \tag{31}$$

$$\varepsilon = \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \mathbf{w}) \tag{32}$$

After estimating the multipliers, the optimal weight vector $\mathbf{w}$ and bias ε can be determined. Again we follow the above definition of SVM for the localization and learned the parameters γ for the RBF kernel and the insensitive error εr based on the profiled data. Once we know the parameters for the linear model, we are ready to predict a test point using these parameters. The value of γ is 0.40 and 0.10 in unscaled and scaled RSS, respectively. The insensitive error εr is 0.0001 for both unscaled and scaled RSS.

4.1.6 Combinatorial localization

In our proposed combinatorial localization, we first consider all the profiled samples, say n, from the database and sort them in terms of the RSS discrepancy between a tag and the samples. Now instead of choosing the closest K of them as the nearest neighbors, we generate $\binom{n}{K}$ combinations, say I, to get that many intermediate K nearest neighbors. Thus we can compute I intermediate locations as the weighted average of the K members. The final estimated location (x_e, y_e) is simply the weighted average of the I intermediate estimations. Note that the weight in each case could be defined as in LEMON. Also, $n = K$ boils down to the LEMON approach. Following is the algorithm for combinatorial localization, which could also be used to determine the optimal value of K based on the profiled samples. That K could be used to estimate the location of the query tag.

Algorithm 4 Combinatorial Localization (Q, n, K)

1: Find the n nearest neighbors of the query tag Q.
2: Sort the n nearest neighbors in terms of the RSS discrepancy.
3: Generate $I = \binom{n}{K}$ set of K nearest neighbors.
4: **for** $i = 1$ to I **do**
5: $\quad x_i \leftarrow \sum_{j=1}^{K} w_j x_j$
6: $\quad y_i \leftarrow \sum_{j=1}^{K} w_j y_j$
7: **end for**
8: $x_e \leftarrow \sum_{l=1}^{I} w_l x_l$
9: $y_e \leftarrow \sum_{l=1}^{I} w_l y_l$

Assume that we have four profiled samples $(1,1)$, $(2,1)$, $(1,2)$, and $(2,2)$. Let K be 3,

so there will be 4 combinations of 3 nearest neighbors. Those are $\{(1,1),(2,1),(1,2)\}$, $\{(1,1),(2,1),(2,2)\}$, $\{(1,1),(1,2),(2,2)\}$, and $\{(2,1),(1,2),(2,2)\}$. If we consider a simple averaging technique to compute locations out of these nearest neighbors, we will have $(1.33,1.33)$, $(1.67,1.33)$, $(1.33,1.67)$, and $(1.67,1.67)$, respectively. The final estimated location would be the simple average of these 4 intermediate estimations, which is $(1.5,1.5)$.

In our localization experiment, a subset of profiled samples, $s \subset S$, is used for the localization. We consider both the scaled and unscaled RSS for the localization. In the former case, the s and K that offer the best estimations are 7 and 6, respectively. In the latter case, these parameters are 10 and 4 to obtain the best estimations. In both cases the averaging formula is the same as the one in LEMON.

4.2 Discussion of the Experimental Results

To evaluate the performance of the above mentioned localization methods, we have performed another series of experiments using the same three graduate student's office (mentioned in the Section 3.4.2) with a bigger setup, i.e., we gathered 52 reference points as part of the radio map construction and tested 33 tag locations. We have considered both the *scaled* and *unscaled* RSS to evaluate various localization methods. In the following we start with the unscaled RSS and then present the results from the scaled data.

The best estimation of LEMON is obtained with $K = 8$. The corresponding error distribution and the performance comparison with LANDMARC and RADAR are shown in Figure 4.2 and Figure 4.3, respectively. In particular, we compare the RSS profiling based localization schemes LANDMARC and RADAR with LEMON. The latter one outperforms the other schemes. Note that we consider the same experimental data for this comparison even though in real life the other two schemes consider different hardware and size of the monitored area. In the following, we will present the performance of different localization

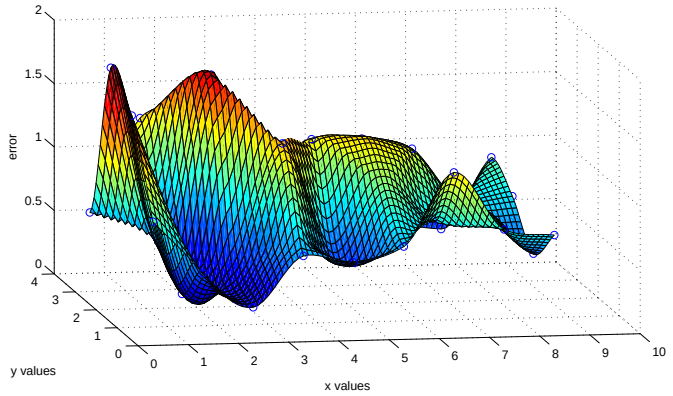


Figure 4.2: The error distribution of LEMON with unscaled RSS.

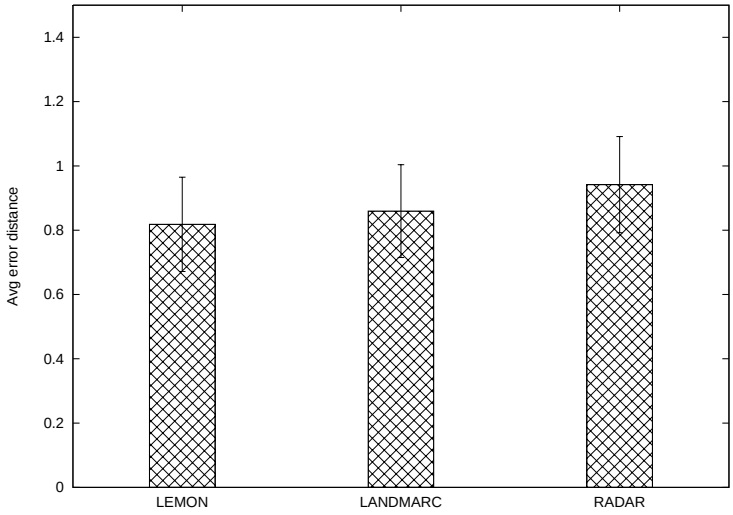


Figure 4.3: The performance comparison of LEMON with LANDMARC and RADAR with unscaled RSS.

methods we have tested using unscaled RSS.

The performance comparison of global and local parameters of lateration is shown in Figure 4.4. The best performance is obtained with 3 pegs and local parameters. The environment with densely deployed furniture may need local parameters to better estimate the path loss exponent to take into account the small scale fading. It also imposes extra parameter estimation which is related to the number of pegs in the system. However, this

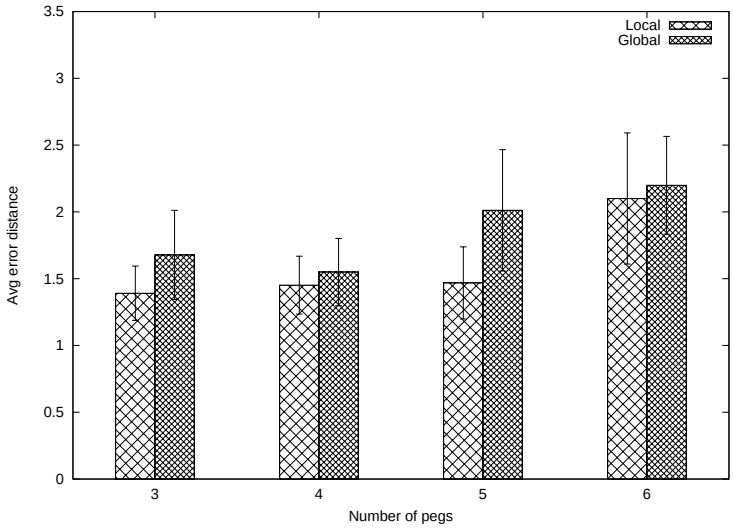


Figure 4.4: The performance of local and global parameters of lateration.

extra estimation is worthwhile in terms of measured accuracy. Thus we may suggest to consider local parameter estimation while using log-distance based lateration to predict the tag's location, especially indoors where the presence of multipath effects is high. However, the performance of lateration is the worst compared to the other methods we have considered. Lateration considers path-loss model that may be approximate. Thus the estimated path-loss exponent, even as a local parameter, may not provide a good distance measurement for lateration. Note that it gives the best results with a small (closest) number of pegs, where chance is high that those pegs have direct LOS with the tag. KNN based LEMON, on the other hand, does not try to separate LOS and NLOS components of signal strength. Indeed, it considers signal strength as a quantity and compares the stored RSS with the query RSS with the expectation that the nearby RSS may experience the same environment thus yielding a similar measurement. A KNN based approach also does not assume that X and Y locations are independent. The performance comparison of different localization methods with unscaled RSS is presented in Figure 4.5.

In combinatorial localization, we try to generate, say I, intermediate estimations using a subset of profiled samples and a suitable K. The idea is to use those I estimations to

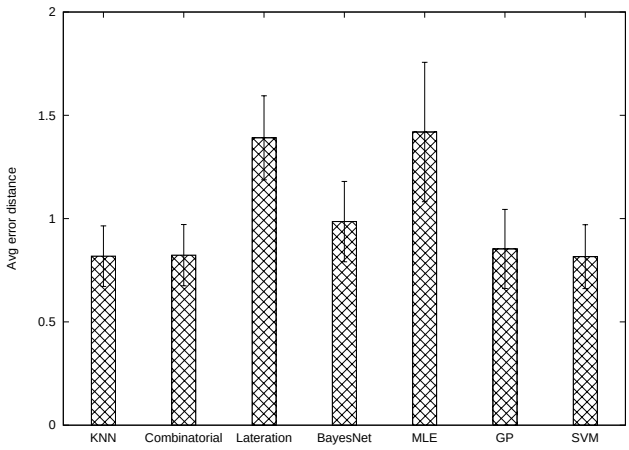


Figure 4.5: The performance comparison of different localization methods without scaled RSS.

produce the final prediction. The performance of this approach is almost the same as of LEMON. However, we may use the combinatorial approach to learn the best value of K using the profiled points. In Bayesian networks, we learn the distribution of profiled points and use that information to learn the probability for a query tag. The final estimation is the weighted average of the profiled points, where the weight is the corresponding probability. The performance of this method is worse compared to LEMON. We use five samples per profiled point to generate the distribution, most of which are very consistent to each other. Thus this may not provide good distributions to be useful for the prediction. Using more samples might help, but by increasing the complexity of the data gathering phase. Without this extra burden we could easily choose LEMON and ensure better accuracy. MLE performs worse compared to the Bayesian Networks as it reports the highest probable profiled point as the estimation. The last two methods GP and SVM both use the kernel trick to project the low-dimensional input space to the high-dimensional feature space. Then they perform linear regression in that feature space. The former learns the posterior distribution of the weight to generate posterior distribution for the query tag. The mean of this distribution is the estimation. The latter one estimates a weight vector that ensures minimum loss

or error during the prediction. Their performance is similar to each other and also similar to LEMON. Thus, a complicated regression after the kernel trick might not be useful for our localization purposes, where simple RSS scaling offers good estimation accuracy, as we will present next. Note that all the results shown in the remainder of the chapter consider scaled RSS.

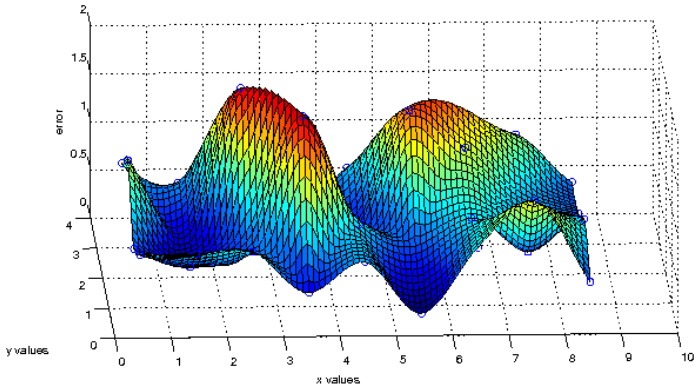


Figure 4.6: The error distribution of LEMON with scaled RSS.

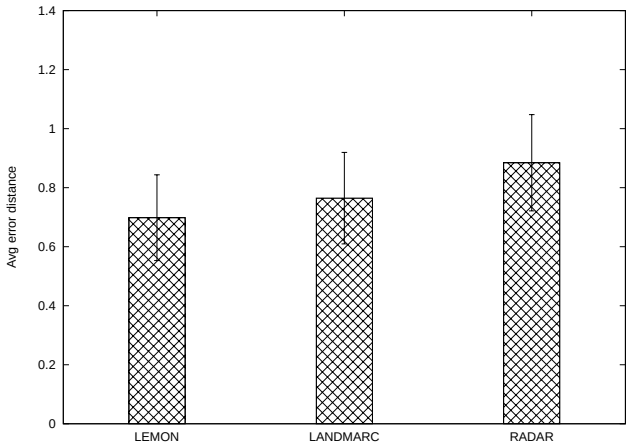


Figure 4.7: The performance comparison of LEMON with LANDMARC and RADAR with scaled RSS.

The best result for LEMON is obtained with $K = 6$ and scaling factor $\gamma = 3.5$ subject to RSS scaling. The error distribution of LEMON is shown in Figure 4.6, where the highest

estimation error is observed near the walls. The performance comparison of LEMON with RADAR and LANDMARC is given in Figure 4.7.

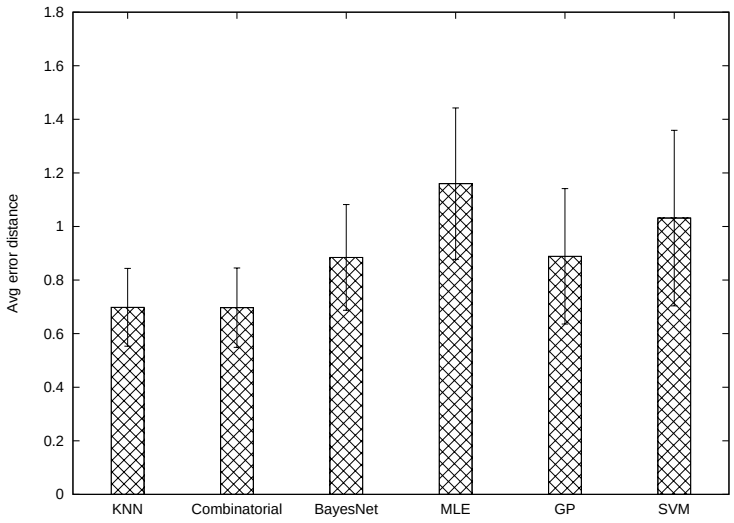


Figure 4.8: The performance comparison of different localization methods with scaled RSS.

The performance of different localization methods using scaled RSS is given in Figure 4.8. Note that log-distance based lateration is not evaluated with scaled RSS as that may ruin the RSS vs distance relationship. The performance of the remaining methods have similar trend as for the unscaled data. LEMON performs best, Bayesian Networks and GP follow it. However, SVM and MLE are the worst. In SVM, scaling the RSS may affect its regression on the feature space.

4.3 Impact of Distance Between Profiled Samples

In this section we focus on LEMON and scaled RSS, and present experimental outcomes of LEMON to show its behavior with a varying sample density. The number and arrangement of the reference points have impact on the localization performance [25]. In addition we may also expect that the density (distance between adjacent reference points) may influence

the estimation. We have tested the performance of LEMON, LANDMARC, and RADAR using reference points with different densities. The outcome is shown in Figure 4.9. Note that in the cases of distance 0.71m and 1.6m, the reference points are placed both at the grid point and in the center of the grid-cell, whereas all these points are placed only at the grid points in the remaining two cases of 1m and 2m distances. Recall that the tested tag's locations are at the center of the grid-cell.

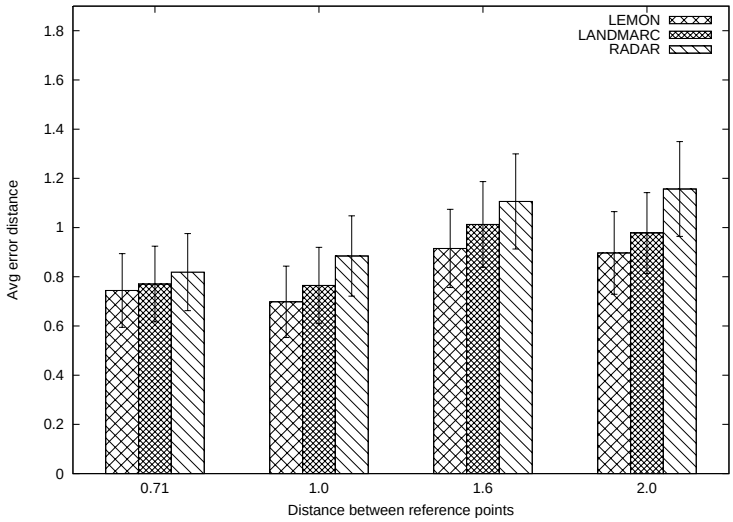


Figure 4.9: Impact of density of reference points.

The worst performance is exhibited by RADAR and it decreases gradually (almost linearly) with decreasing density. RADAR chooses K closest neighbors based on the minimum RSS discrepancy and then averages those coordinates in a straightforward manner to produce the estimate. On the other hand the other two schemes apply weighted average of the chosen K coordinates, where the weight is based on the RSS discrepancy. The impact of the RSS discrepancy along with the density is noticeable. The initial set up of the reference points considers 0.71m distance between them. In the next step we choose 1m and the error is reduced. This deterioration can be attributed to the inherent limitation imposed by the RF module on the accuracy of its RSS reports, as well as on the meaning of actual (ideal) values of RSS in the complicated propagation environment. In such a situation, the

overly dense coverage with samples tends to contribute noise rather than information, and readings from close locations introduce even more uncertainty into the radio map. Increasing the distance further decreases the performance, as we also expect, but the interesting point is that the highest error is observed with distance of 1.6m rather than 2m. Thus, as we demonstrated in [25], the arrangement of sample points in the monitored area is of primary importance, their density ceases to improve things below a certain threshold, which, according to our observations, lies close to the minimum distance of 1m.

4.4 Multi-channel RF-based Indoor Localization

The purpose of considering multiple channels is to observe their effect on RSS. Channels may have different degree of noise sensitivity, and RSS from the same location may behave differently across different channels. Some channels may be less affected by multipath compared to others. Thus we may able to define reliability of channels in terms of RSS fluctuations. In particular, cross correlation of the pegs can be calculated to rank the channels in terms of the number of highly correlated pegs. The *cross correlation coefficient c* between the RSS and distance for each peg can be defined as:

$$c = \frac{\sum_{i=1}^{n}(r_i - \overline{r})(d_i - \overline{d})}{\sqrt{\sum_{i=1}^{n}(r_i - \overline{r})^2 \sum_{i=1}^{n}(d_i - \overline{d})^2}} \tag{33}$$

where n, r, and d are the number of profiled samples, the RSS value, and the known distance between the peg and the profiled point, respectively. The value of c is between -1 and 1, and the smaller the value the weaker the correlation between the distance and RSS.

Definition. *Reliable peg is a peg with cross-correlation coefficient greater than a threshold.*

where threshold is an estimated parameter.

Definition. *Channel reliability is the ratio of the number of reliable pegs to the total number of pegs,* $CH_{re} = \frac{R_p}{T_p}$.

R_p and T_p are the number of reliable pegs and total number of pegs, respectively. The estimated locations across different channels can be further averaged as follows;

$$(x_e, y_e) = \sum_{i=1}^{ch} w_i(x_i, y_i), where\ w_i = \frac{CH_{re_i}}{\sum_{i=1}^{ch} CH_{re_i}} \tag{34}$$

ch is the total number of channels and (x_i, y_i) is the estimated location using channel *i*. Note that if CH_{re} is 1 for each channel, i.e., all of them are perfectly reliable, the estimation becomes a simple averaging across the channels.

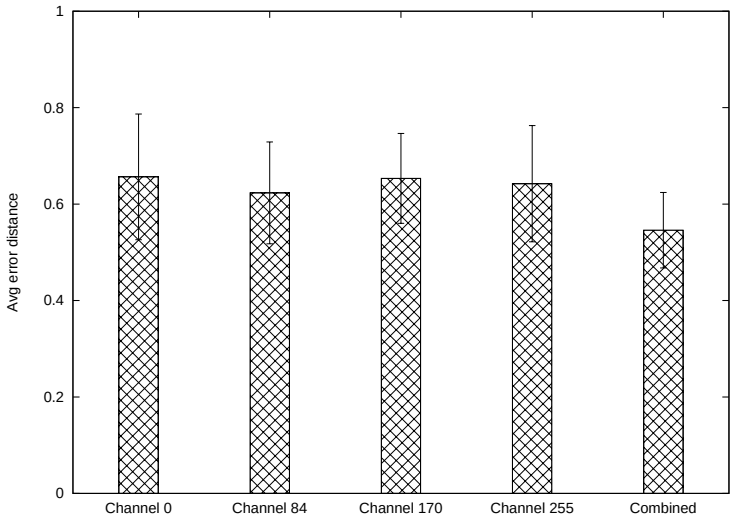


Figure 4.10: The effect of using multiple channels on localization.

We perform an experiment using three consecutive rooms and set a 13m$\times$3m grid, where 4 different RF channels (0, 84, 170, and 255) (channel 0 is 800MHz and consecutive channels are 200kHz apart) are used to gather RSS values. One of the walls is about 1m thick whereas the other one is 0.15m. As in other experiments, we gather profiled points from every grid point but the test points are located at the center of each grid-cell. There are 73 profiled points to test 36 test points. We place 22 pegs to cover the entire area. The idea is to place more pegs near the walls shared by consecutive rooms. The best RSS

scaling factor (γ) and the number of nearest neighbors (K) for each channel is presented in Table 4.2.

Channel	RSS scaling factor (γ)	Nearest neighbors (K)
CHANNEL 0	4.0	7
CHANNEL 84	3.5	8
CHANNEL 170	4.5	9
CHANNEL 255	3.0	5

Table 4.2: Best parameters for channels.

The localization is first performed using individual channels and the performance is shown in Figure 4.10. Then we take the average of these estimated locations across the channels as the new estimation. Applying weighted average, based on above definitions, helps to improve the estimation accuracy.

4.5 RF-based Room Localization

Buildings are usually composed of multiple rooms. Indeed, in many applications of indoor localization room identification may be sufficient instead of knowing the exact location of the tag. Given a set of samples from the reference points, the room localization problem is to identify the room from where tag sends its query. We tested 33 tag's queries using 5 nearest neighbors and identified the room where these tags belonged by using the label of these 5 nearest neighbors. In particular we choose 5 nearest neighbors that minimize the RSS discrepancy. Then we check their room label and count for the majority, which is chosen as the label of the query tag. Out of the 33 queries, only one is misclassified.

As room localization performs adequately and has many potential applications, we perform further experiments in multiple rooms (mentioned in Section 4.4) and test the room localization for that setup. The outcome is shown in Table 4.3

Channel	Correct Localization	Wrong Localization
CHANNEL 0	35	1
CHANNEL 84	35	1
CHANNEL 170	34	2
CHANNEL 255	34	2

Table 4.3: Room localization using multi-channels.

4.6 Conclusions

In this chapter we analyzed different localization methods ranging from a simple nearest neighbor search to the complex SVM. We tested localization based on lateration, which estimates the path-loss exponent to relate the distance between the transmitter and receiver and use this distance to perform lateration. The best performance is offered by the nearest neighbor search which is the mechanism at the heart of LEMON. This is because it does not need to separate the LOS and NLOS components of signal strength, instead it considers the signal strength as an observed quantity of the monitored area and performs localization by computing the discrepancy between the stored and query signal strength observations to choose the close-match neighbors with known locations to report the estimated location as the weighted sum of those locations. Also, it does not assume any distribution of signal strength (as other methods that assume the signal to be Gaussian) or the independency of X and Y coordinates. All these features make the nearest neighbor search based approach very flexible and effective for profiling based localization.

We performed a new experiment with a bigger setup to evaluate the performance of the above mentioned localization methods. The performance of LEMON is better than the other approaches or it is almost equal but at a lower cost.

In addition, we analyzed some other properties of LEMON like the impact of the profile points density. We have also performed localization while considering multiple channels.

Finally, we have proposed a simple room localization technique and verified its performance under both single and multiple RF channels. Experimental results show that the multiple channels help to improve the localization accuracy. Also, the accuracy of the room localization is promising.

Chapter 5

Localization Robustness

In this chapter we study the robustness of LEMON. For this purpose, we first study the impact of dimensionality expansion of RSS on LEMON and conclude that LEMON does not need such techniques to further improve its accuracy. The robustness of LEMON is then tested under erroneous RSS (introduced in the profile data on purpose). Finally, we propose a couple of two-step localization schemes namely, *midrange* and *peg reliability* based localization, in order to take care of noisy RSS and make LEMON robust.

5.1 The Impact of Dimensionality Expansion

The use of RSS in its scaled form (versus non-scaled) can appear at first to be an arbitrary fix. Indeed, we are interested in identifying what kind of RSS pre-processing can add to the accuracy and robustness of the localization. To this end, we consider a number of alternatives outlined in this and in subsequent sections. We start with a technique called *dimensionality expansion*. In a localization system based on WiFi APs, the number and placement of infrastructure nodes are not flexible, because they are decided by the requirements of their primary role of being APs. In [28], four such nodes are utilized to cover a comparatively large monitored area and perform localization. To improve the accuracy, the authors suggested a scheme to expand the RSS dimensionality by using pairwise

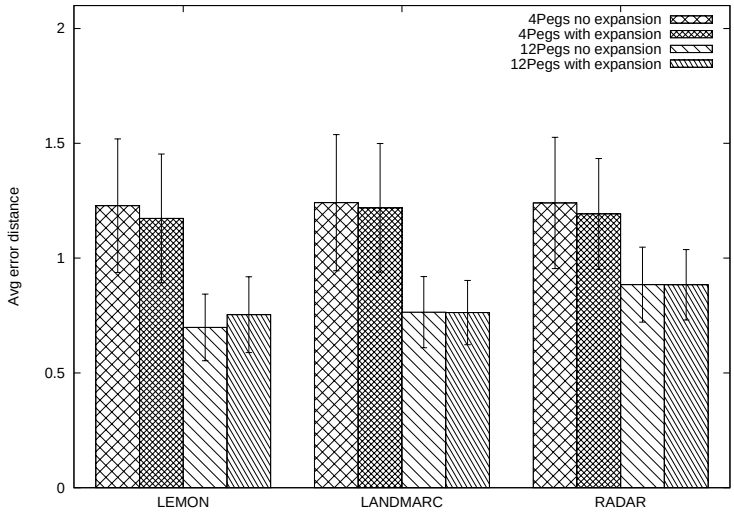


Figure 5.1: Effect of RSS dimension expansion.

differences between RSS values perceived by different pegs. Normally, with m pegs, the compound RSS sample (viewed as a vector of RSS readings) is at most m-dimensional. Its dimensionality will be exactly m, if all m pegs perceive the tag, which may be a rule in a sparse high-powered WiFi setup but it is much less natural with LEMON. By using pairwise differences, we can increase the dimensionality of a sample vector stored for each reference point to $M = \binom{m}{2}$, which represents the differences in (non-scaled) measurements between pairs of pegs/APs for each given reference point. In [28], the authors reported an improvement in the localization accuracy when using 6 differentials instead of the 4 straightforward RSS readings. LEMON already utilizes a large number of infrastructure nodes achieving an impressive accuracy. We may thus expect that dimensionality expansion does not help much in improving LEMON's localization accuracy. The results of such experiments are shown in Figure 5.1. We use 4 pegs at the four corners of the monitored area and apply the pairwise RSS difference among the pegs to obtain 6-dimensional RSS vectors. This improves the results slightly compared to the 4-peg setup. In the case of 12 pegs, LANDMARC and RADAR experience no improvement, while the performance of LEMON slightly degrades. In a nutshell, the dimensionality expansion might be helpful in

a setup of limited number of infrastructure nodes, but deploying a large number of low-cost low-power devices yields good estimation without resorting to such techniques. Using a large number of devices helps to (naturally and independently) generate high dimensional RSS values to better capture the specific environment, which also helps the localization. However, a small number of infrastructure nodes may not allow one to get such a detailed view of the environment, thus RSS dimension expansion could be used in those cases.

5.2 Robustness Under Imperfect RSS Measurements

In this section we consider the various ways in which the inherent unreliability of RSS measurements impacts the localization and mitigation techniques that can be applied to deal with such unreliable measurements.

5.2.1 The impact of erroneous RSS measurements

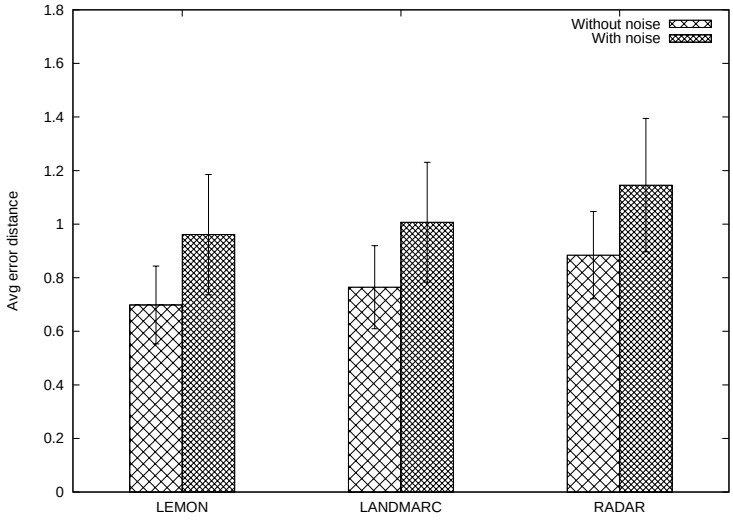


Figure 5.2: Effect of noisy RSS on localization.

In the presence of multipath propagation, RSS may correlate with distance in an unpredictable way. While the primary role of profiling is to capture *any* correlation that might

exist between RSS and locations in space, the ultimate case of a difficult environment occurs when there is no apparent correlation at all. Clearly, the total lack of any correlation whatsoever renders all attempts to use RSS for location estimation futile. However it still makes sense to ask this question: how resistant an RSS-based scheme is to the partial loss of correlation. Such a loss may result from a particularly malicious propagation properties in a certain region of the monitored area (e.g., lots of metallic objects), or by a malfunctioning peg. The latter problem may become particularly relevant in systems with a large numbers of cheap nodes, as the ones targeted by LEMON.

To test the sensitivity of our scheme to such properties of the environment, we have performed an experiment whereby a selected peg (one such peg considered at a time) would produce uniformly distributed random RSS readings between *MIN* and *MAX* (minimum and maximum value of RSS). We have tested 33 localization queries over all possible scenarios in our setup involving one peg (a different peg in each scenario) misbehaving in this fashion. As the same problem can affect any RSS-based scheme, we have subjected LANDMARC [45] and RADAR [3] to the same test. Note that the implementation of those schemes was in a sense virtual, and more favorable than of their original versions, as they operated on our hardware (with the same coverage by pegs as LEMON), the only difference being their way of transforming the RSS data into location estimates. Needless to say, all schemes suffered a drop in the accuracy of their estimates (see Figure 5.2) but the relative order in terms of performance, with LEMON outperforming LANDMARC and RADAR, was preserved.

5.2.2 The impact of outlier RSS values

Next we consider a closer inspection of the collected RSS measurements with the purpose of discarding the ones that are likely less useful. Eyeballing the collected RSS values one can identify three “regions,” two at the extreme ends, and a midrange of values that are nicely related to distance. RSS values are ranging from -42 to -110 dBm, where the high

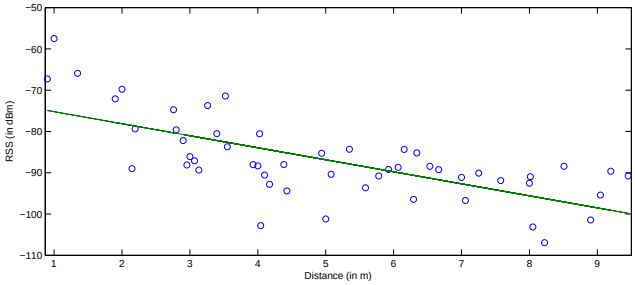


Figure 5.3: The RSS vs distance for peg 11.

values fall around -70 dBm, low ones around -95 dBm, and the rest are midrange RSS values (see Figure 5.3). We observe that the low and high values can be unreliable compared to midrange values, i.e., they are unrelated to the corresponding physical distances between sender and receiver. For example the points that fall below -100dBm in Figure 5.3 appear to be outliers (e.g., at distance 4m and 5m) and so appears to be the point above -60dBm at distance 1m. The low RSS values are not far from the noise floor, while the high RSS values denote a peg that is probably close enough to the transmitting tag and its receiver may be saturated by the impact of nearby transmission. To mitigate the effect on localization of noisy RSS from the "unreliable" extremes of the range of values, we introduce a two-step localization process. First, we compute the cross correlation coefficient c between the RSS and distance for each peg. We notice that, for each peg, c can be further improved by trimming off the extremes of the RSS values for the peg. This simply implies that those two extremes of the range introduce noise to the captured RSS values. We therefore isolate the "midrange" values of RSS as the most useful, while replacing the extremes with two representative values (one for the high end and one for the low end values). The list of upper and lower values of RSS for each peg is shown in Table 5.1.

Thus we first compute the coefficient c for each peg and then attempt to trim t% of RSS values from both ends of the RSS range for the peg. We eliminate outliers by starting with the furthest RSS extremes and trimming performed in such a way that the value of

Peg	Lower value	Upper value
1	−94	−52
2	−105	−72
3	−100	−72
4	−94	74
5	−96	−71
6	−95	−74
7	−94	−67
8	−99	−78
9	−102	−66
10	−102	−81
11	−97	−65
12	−105	−59

Table 5.1: Upper and lower values of RSS (in dBm) in midrange localization.

c improves. The trimmed values are replaced as we mentioned them above. This is done (instead of discarding the trimmed RSS completely) because we would not like to distort the dimensionality of the measurements. Clearly, we do not perform any trimming of the RSS values for the measurements of the tag that is querying about its location. (Plus, it would have been dubious at best given that we do not know its distance from the pegs.) For each peg it is possible to find its best *c* by trimming any percentage of RSS values from either end. However, this also means that different pegs may loose different percentage of RSS values and may have different impact on localization. Thus we trim up to a fixed percentage of RSS values such that we are not loosing much but at the same time are able to avoid extreme RSS (at both extremes) as well as improve *c*. Finally, we perform localization using the newly constructed association list of profiled samples.

The performance of midrange RSS based localization is shown in Figure 5.4. Note that we used unscaled RSS in midrange as well as in the next section on peg reliability. The reason of doing so is to capture the relationship between RSS and absolute distance; whereas, if we perform scaling, we lose that RSS vs. distance relationship. We start with 5% trimming, then move to 10%, and finally consider 20%. For 5% trimming localization

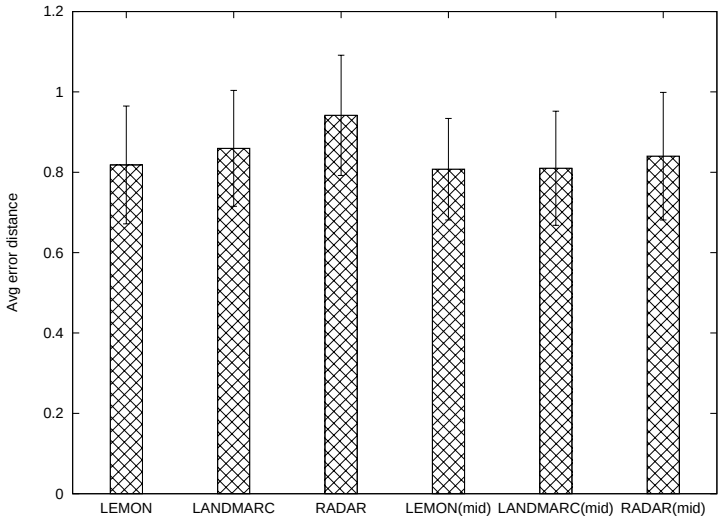


Figure 5.4: The midrange RSS based localization.

accuracy remained almost the same. However, it improved when trimming was increased to 10%. Further trimming though makes the localization results worse. Thus there is a trade-off between improving c and localization accuracy. If we trim more RSS values c improves but that also decreases the localization accuracy. Trimming very small percentage also does not help as we still may have outlier RSS values in our set of measurements. We found 10% is a good choice to improve the localization accuracy. We also compare the midrange localization performance of LEMON with LANDMARC and RADAR, and they also experience performance improvement when consider midrange localization. In a nutshell, midrange RSS based localization seems promising in terms of improving localization accuracy, where we need to reconstruct the association list of each profiled sample based on the cross correlation coefficient of each peg and trimming the unwanted RSS values. An open question is that of determining a systematic RSS measurement trimming/rejection strategy that prescribes what needs to be trimmed automatically, and takes the appropriate action.

5.2.3 Reliability assessment of pegs

Another way to study the RSS data set is to attempt to characterize the reliability of the pegs without resorting to any (as we set forth in the beginning) assumption of a relation between distance and RSS value. RSS measurements, in addition to the contamination by multipath propagation, may also be affected by the hardware unreliability of the infrastructure nodes. We have mentioned before that our infrastructure nodes (pegs) are low-cost and low-power devices, which may introduce unreliability on the measured RSS. Here, we will attempt to introduce metrics to measure the reliability of the pegs. We use the correlation between the measured RSS and the corresponding distance between the transmitter and receiver based on the profiled samples. However, instead of using the original RSS, we consider midrange values to compute the correlation coefficient, c_m and use it as a reliability measure. The higher the coefficient the better its reliability. This measured reliability factor can be used to control the influence of a peg in the RSS discrepancy as follows:

$$(RSS_{ti} - RSS_{fi})^2 \times w_i \tag{35}$$

where RSS_{ti} and RSS_{fi} are the RSS values from the association list of a tag and reference point, for peg i, respectively. We consider the following weight w_i for a peg i;

$$w_i = \frac{c_{m_i}}{\sum_{i=1}^{m} c_{m_i}}$$

The accuracy of localization based on peg reliability is tested using both original and midrange RSS. We found that the midrange RSS helps to further improve the accuracy if peg reliability is introduced. However, the original RSS along with peg reliability does not help. This may demonstrate that trimming noisy or unreliable RSS is a good idea to improve the localization accuracy. The outcome of our experiments is shown in Figure 5.5. The performance of LEMON with original RSS is not improved, it rather deteriorates a bit under consideration of peg reliability. However, accuracy experiences improvement if

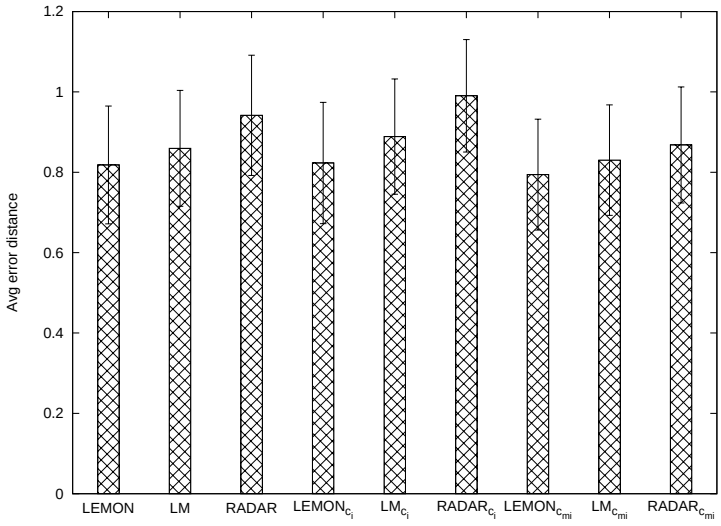


Figure 5.5: The effect of peg reliability on localization.

midrange RSS values and peg reliability is used.

5.3 Conclusions

In this chapter we study the robustness of LEMON and then two-step localization techniques are introduced. Midrange based localization modifies the RSS vector or association list of profiled samples based on the cross correlation of a peg. This modified RSS vector is then used for localization. In peg reliability based localization, c is used as weight during RSS discrepancy measurement and localization is performed accordingly. The used c is computed using the midrange RSS of a peg. These methods improve the estimation accuracy of tags. Thus we conjecture that the presented techniques can make localization robust in the presence of noisy RSS and ensure good estimation accuracy. In all cases, the presented extensions and modifications result in LEMON equaling our outperforming the state-of-the-art schemes.

Chapter 6

Conclusions

We may divide the contribution of this thesis in two parts. In the first part, we address 1) scalability issues in ad hoc and sensor networks, 2) local maxima and routing loops problems in location-based routing protocols, and 3) load balancing and throughput issues for the same group of protocols. To address the first issue we choose location-based routing protocols as this class of protocols require no maintenance of routing tables. This helps location-based routing scale despite the nodes' limited resources. Secondly, we introduce and propose a set of randomized routing protocols to handle the local maxima and loop problems of location-based Greedy and Compass routing. We further analyze the behavior of deterministic and randomized routings under low to high traffic load and also considered random and biased traffic. Based on the analysis we propose randomized protocols to help avoiding congested nodes on the way to a destination and balance the loads among the nodes. This in turn also helps the proposed protocols obtain high throughput.

Load balancing, throughput, packet delivery ratio, and path length all are important performance metrics for any routing protocols. However, the main concern of any sensor network is to optimize the energy consumption of the nodes to prolong the network life-time. Note that nodes utilize energy for both computation and communication, where the latter one is more dominant. If a node fails to forward a packet to a neighbor according

to the definition of a protocol, it retransmits the packet. The more the retransmissions the higher the energy consumption. Thus by keeping the total number of transmissions across the network as limited as possible we may be able to optimize the energy utilization. Thus the challenge is to integrate the load balancing and the energy optimization to obtain better routing performance in terms of the throughput, the packet delivery ratio and the path length. As future work we plan to propose such routing protocol for ad hoc and sensor networks. In particular, nodes may operate in two modes namely: *regular* and *congested*. Initially all nodes are in regular mode. If a node detects that a certain percentage of its packets are not getting through, it may switch to the congested mode and subsequent packets are delivered based on new routing criteria. In particular, the next node may be chosen based on the remaining queue length of the neighbors (both deterministic and weighted randomized options could be tried). Also a regular node after learning about a congested neighbor (by hearing its broadcast) may change the routing strategy. For instance, if a congested node is chosen as the next node, alternate routes considered. A congested node may switch back to the regular mode once congestion is overcome.

In the second part of this work we consider indoor localization, i.e., computing the Cartesian coordinates of nodes indoors where GPS signals are usually unavailable. In the presence of multi-path, indoor localization becomes a challenging problem and out of different existing solutions RSS profiling received attention due to its ability to offer high estimation accuracy. The profiling based localization is a two-step process where radio map of the monitored area is first constructed by gathering RSS from know locations. The estimation can be performed by searching the K nearest neighbors of the query node to take the weighted average of their coordinates. We have proposed such localization scheme, dubbed LEMON, that uses a densely deployed set of low-cost low-power sensors to gather the RSS. In turn it experiences better estimation accuracy compared to the state-of-the-art. In addition to the KNN method, it is possible to use lateration, Bayesian Networks, MLE,

Gaussian Process, and SVM for the profiling based localization. However, our experimental results show that the KNN based approach is low cost and effective for such profiling based localization and can outperform other methods. We have also analyzed various properties of LEMON and proposed methods to improve the accuracy of LEMON.

Room localization is a problem where the room label of the query node is identified rather than computing its actual location. We have conducted a series of experiments to perform room localization based on KNN. The outcome of these experiments is promising, where both single and multiple channels are used. Finally, we have proposed a set of two-step localization schemes based on LEMON. The purpose of these schemes is to make LEMON robust against noisy RSS.

Given the extensive evaluation we performed, we can state with some confidence that the proposed LEMON could be considered as a complete solution of indoor localization in various environmental conditions.

One issue of profiling based localization is how to make the offline profiling phase simple. It may even possible to avoid this phase completely by densely deploying the pegs to cover the monitored area. Each peg will send a transmission that will be captured by the remaining, say $(m-1)$, pegs. Thus for m pegs we will obtain m association lists each with $(m-1)$ dimensional RSS vector. Then the location of these pegs can be estimated to learn best K, which will be used to answer new queries from the tags.

The profiling-based localization that we described above utilizes a single-hop network for the localization purposes. The multihop setup is also possible where pegs may use simple routing protocol to pass the received signal strength to the master node. However, we may even deploy a network of a large number of sensors indoors or outdoors such that single-hop communication is not possible to reach all the sensors. Instead of using profiling-based scheme we may consider cooperative localization. In this scheme a tag instead of sending signal for localization may search for the nearby pegs it can hear. The tag can simply measure the RSS from these pegs to assign them weight. The stronger

the RSS the higher the weight and the weighted average of the coordinates of the nearby pegs may boil down to the estimated location. Thus deploying the pegs is crucial for the estimation accuracy. Once such deployment is discovered, pegs may periodically broadcast their locations and nearby tag may capture it. The tag may then apply the above mentioned weighted average based estimation. In a sense, the pegs can act also as ”anchor” nodes.

In [40], a cooperative localization, CCA-MAP, for multihop sensor networks is proposed. In this scheme, a small number of sensors equipped with GPS are deployed as anchors. Regular nodes first compute a local map based on the connectivity information among neighbors. A nonlinear data projection is then applied to obtain relative coordinates. The local maps are then combined together to get a global map. Finally using the locations of the anchors, absolute coordinates are computed. This localization scheme uses mainly connectivity information among neighbors. It was found that applying range-based approach to get distances among the neighbors does not help. The CCA-MAP could be a choice for localization in a large network where generating the radio map for profiling-based localization is not trivial. However, the accuracy of CCA-MAP depends on the number and the placement of the anchors. We may define the anchor deployment problem as an optimization problem with constraints on the estimation accuracy. Tags may frequently appear and disappear, causing CCA-MAP to re-execute frequently. We can propose an incremental CCA-MAP localization to reduce the communication and computation cost, a key concern in large networks. Finally, we can investigate the relationship between weak/unreliable RSS readings and connectivity to further improve CCA-MAP, using results/insights from our work on LEMON.

Bibliography

[1] S. Ali and P. Nobles. A novel indoor location sensing mechanism for IEEE 802.11 b/g wireless LAN. In *4th Workshop on Positioning, Navigation, and Communication, WPNC'07*, pages 9–15, Hannover, Germany, March 2007.

[2] C. Alippi and G. Vanini. An RSSI-based and calibrated centralized localization technique for wireless sensor networks. In *Proceedings of the 4th IEEE International Conference on Pervasive Computing and Communications*, pages 301–305, Washington, DC, USA, March 2006.

[3] P. Bahl and V.N. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM 2000)*, pages 775–784, Tel Aviv, Israel, March 2000.

[4] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward. A distance routing effect algorithm for mobility (DREAM). In *Proceedings of the 4th ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 1998)*, pages 76–84, Dallas, Texas, USA, October 1998.

[5] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. Macaw: A media access protocol for wireless lans. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM 1994)*, pages 212–225, London, UK, September 1994.

[6] N. M. Boers, I. Nikolaidis, and P. Gburzynski. Patterns in the rssi traces from an urban environment. In *Proceedings of the IEEE International Workshop on Computer-Aided Modeling Analysis and Design of Communication Links and Networks (CAMAD 2010)*, Miami, Florida, USA, December 2010.

[7] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[8] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5):28–34, 2000.

[9] United States Coast Guard Navigation Center. Global positioning system standard positioning service specification, June 1995.

[10] L. Chan, J. Chiang, Y. Chen, C. Ke, and J. Hsu. Collaborative localization: Enhancing WiFi-based position estimation with neighborhood links in clusters. In *Proceedings of the International conference on Pervasive Computing (PERVASIVE 2006)*, pages 50–66, Dublin, Ireland, May 2006.

[11] T. W. Chen and M. Gerla. Global state routing: A new routing scheme for ad-hoc wireless networks. In *Proceedings of the IEEE International Communications Conference (ICC 1998)*, pages 171–175, Atlanta, GA, USA, June 1998.

[12] C. C. Chiang, H. K. Wu, W. Liu, and M. Gerla. Routing in clustered multihop, mobile wireless networks with fading channel. In *Proceedings of the IEEE Singapore International Conference on Networks (SICON 1997)*, pages 197–211, Singapore, April 1997.

[13] I. Chlamtac, M. Conti, and J. Liu. Mobile ad hoc networking: Imperatives and challenges. *Ad Hoc Networks*, 1(1):13–64, 2003.

[14] R. R. Choudhury and N. H. Vaidya. Impact of directional antennas on ad hoc routing. In *Proceedings of the 8th International Conference on Personal Wireless Communication (PWC 2003)*, pages 590–600, Venice, Italy, September 2003.

[15] Campbell Scientific (Canada) Corporation. GPS16X-HVS GPS Receiver, GPS Receiver Manual. www.campbellsci.com/documents/manuals/gps16x-hvs.pdf, January 2008.

[16] G.G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISU/RR-87-180, USC ISI, Marina del Ray, CA, March 1987.

[17] V. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello. Bayesian filtering for location estimation. *Pervasive Computing*, 2(3):24–33, 2003.

[18] K. Gabriel and R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18(3):259–278, 1969.

[19] P. Gburzynski. SMURPH (a system for modeling unslotted real-time phenomena). http://sheerness.cs.ualberta.ca/ pawel/SIDE/MANUAL/manual.htm.

[20] P. Gburzynski and W. Olesinski. On a practical approach to low-cost ad hoc wireless networking. *Journal of Telecommunications and Information Technology*, 2008(1):29–42, 2008.

[21] S. Giordano, I. Stojmenovic, and L. Blazevic. Position based routing algorithms for ad hoc networks: A taxonomy. In X. Cheng, X. Huang, and D.Z. Du, editors, *Ad Hoc Wireless Networking*, pages 103–136. Kluwer Academic Publishers, December 2003.

[22] M. Gunes, U. Sorges, and I. Bouazizi. ARA—the ant-colony based routing algorithm for manets. In *Proceedings of the International Workshop on Ad-hoc Networking (IWAHN)*, pages 79–84, Vancouver, Canada, August 2002.

[23] I. Haque, I. Nikolaidis, and P. Gburzynski. On the pitfalls of directional location-based randomized routing. In *Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2007)*, pages 41–48, San Diego, CA, July 2007.

[24] I. Haque, I. Nikolaidis, and P. Gburzynski. On the benefits of nondeterminism in location-based forwarding. In *IEEE International Communications Conference (ICC 2009)*, Dresden, Germany, June 2009.

[25] I. Haque, I. Nikolaidis, and P. Gburzyński. On the impact of node placement and profile point selection on indoor localization. In *Proceedings of the 2nd International IFIP/IEEE WG 6.8 Joint Conference on Wireless and Mobile Networking (WMNC 2009)*, pages 220–231, Gdańsk, Poland, September 2009.

[26] I. Haque, I. Nikolaidis, and P. Gburzyński. A scheme for indoor localization through RF profiling. In *Proceedings of the International Workshop on Synergies in Communications and Localization (SyCoLo'09)*, Dresden, Germany, June 2009.

[27] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009.

[28] M. Hossain, H. N. Van, Y. Jin, and W. Soh. Indoor localization using multiple wireless technologies. In *Proceedings of the IEEE International Conference on Mobile Adhoc and Sensor Systems*, pages 1–8, Pisa, Italy, October 2007.

[29] G. Jin, X. Lu, and M. Park. An indoor localization mechanism using active RFID tags. In *Proceedings of International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, pages 40–43, Taichung, Taiwan, June 2006.

[30] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[31] S. Jung and W. Woo. ubiTrack: Infrared-based user tracking system for indoor environments. In *Proceedings of the International Conference on Artificial Reality and Telexistence (ICAT 2004)*, pages 181–184, Coex, Korea, December 2004.

[32] J. Kang, D. Kim, and Y. Kim. RSS self-calibration protocol for WSN localization. In *Proceedings of the 2nd International Symposium on Wireless Pervasive Computing*, pages 181–184, San Juan, Puerto Rico, February 2007.

[33] B. Karp and H. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th ACM Conference on Mobile Computing and Networking (Mobicom 2000)*, pages 243–254, Boston, MA, USA, August 2000.

[34] Y.B. Ko and N.H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. *Wireless Networks*, 6(4):307–321, 2000.

[35] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proceedings of the Canadian Conference on Computational Geometry (CCCG 1999)*, pages 51–54, Vancouver, BC, Canada, August 1999.

[36] Axel Küpper. *Location–Based Services: Fundamentals and Operation*. Wiley, 2005.

[37] J. Kuruvila, A. Nayak, and I. Stojmenovic. Progress based localized power and cost aware routing algorithms for ad hoc and sensor wireless networks. In *Proceedings of the International Conference on AD-HOC Networks and Wireless*, pages 294–299, Vancouver, BC, July 2004.

[38] K. Langendoen and N. Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 43(4):499–518, 2003.

[39] J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *Proceedings of the ACM/IEEE International*

Conference on Mobile Computing and Networking (MobiCom 2000), pages 120–130, Boston, MA, USA, August 2000.

[40] L. Li and T. Kunz. Cooperative node localization using nonlinear data projection. *ACM Transactions on Sensor Networks*, 5(1):1–26, 2009.

[41] D. Lowd and P. Domingos. Naive bayes models for probability estimation. In *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*, pages 529–536, Bonn, Germany, August 2005.

[42] J. P. Macker and M. S. Corson. Mobile ad hoc networking and the ietf. *Mobile Computing and Communications Review*, 2(1):9–14, 1998.

[43] M. Mauve, J. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad-hoc networks. *IEEE Network*, 15(6):30–39, 2001.

[44] S. Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM Mobile Networks and Applications Journal*, 1(2):183–197, 1996.

[45] L.M. Ni, Y. Liu, Y.C. Lau, and A.P. Patil. LANDMARC: indoor location sensing using active RFID. *Wireless Networks*, 10(6):701–710, 2004.

[46] D. Niculescu and B. Nath. Ad hoc positioning system (APS). In *Proceedings of the IEEE International Conference on Global Communication (GLOBECOM 2001)*, pages 2926–2931, San Antonio, Texas, November 2001.

[47] D. Niculescu and B. Nath. VOR base stations for indoor 802.11 positioning. In *Proceedings of the 10th International Conference on Mobile computing and networking (MobiCom 2004)*, pages 58–69, New York, NY, USA, October 2004.

[48] W. Olesinski, A. Rahman, and P. Gburzynski. Tarp: A tiny ad-hoc routing protocol for wireless networks. In *Proceedings of the Australian Telecommunications Networks and Applications Conference*, pages 8–10, Melbourne, Australia, December 2003.

[49] V. Otsason. Accurate indoor localization using wide GSM fingerprinting. Master's thesis, Department of Computer Science, University of Toronto, June 2005.

[50] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM 1997)*, pages 1405–1413, Kobe, Japan, April 1997.

[51] V. D. Park and M. S. Corson. A performance comparison of TORA and ideal link state routing. In *Proceedings of the IEEE Symposium on Computers and Communications*, pages 592–598, Athens, Greece, June 1998.

[52] S.N. Patel, K.N. Truong, and G.D. Abowd. Powerline positioning: a practical sub-room-level indoor location system for domestic use. In *Proceedings of the International Conference of Ubiquitous Computing (UBICOMP 2006)*, pages 441–458, California, USA, September 2006.

[53] C. Perkins, E. Belding Royer, and S. Das. Ad-hoc On-demand Distance Vector Routing (AODV), February 2003. Internet Draft: draft-ietf-manet-aodv-13.txt.

[54] C.E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proceedings of the ACM Conference on Communications Architectures, Protocols and Applications (SIGCOMM 1994)*, pages 234–244, London, UK, August 1994.

[55] A. Rahman, W. Olesinski, and P. Gburzynski. Controlled flooding in wireless ad-hoc networks. In *Proceedings of the International Workshop on Wireless Ad-hoc Networks (IWWAN 2004)*, Oulu, Finland, June 2004.

[56] Theodore S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, 2nd edition, 2003.

[57] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[58] T. Roosta, M. Menzo, and S. Sastry. Probabilistic geographic routing in ad hoc and sensor networks. In *Proceedings of the International Workshop on Wireless Ad-Hoc Networks*, London, UK, May 2005.

[59] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: Modern Approach*. Prentice Hall, 3rd edition, 2003.

[60] F. Seco, A. R. Jimenez, C. Prieto, J. Roa, and K. Koutsou. A survey of mathematical methods for indoor localization. In *IEEE International Symposium on Intelligent Signal Processing*, pages 9–14, Budapest, Hungary, August 2009.

[61] S. Singh, M. Woo, and C.H. Raghabendra. Power aware routing in mobile ad hoc networks. In *Proceedings of the 4th ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 1998))*, pages 181–190, Dallas, Texas, USA, October 1998.

[62] I. Stojmenovic. Position-based routing in ad hoc networks. *IEEE Communications Magazine*, 40(7):128–134, July 2002.

[63] R. Stoleru, T. He, and J. A. Stankovic. Range-free localization. In R. Poovendran, C. Wang, and S. Roy, editors, *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks*, chapter 30. Springer, 2007.

[64] E. Stroulia, D. Chodos, N.M. Boers, J. Huang, P. Gburzynski, and I. Nikolaidis. Software engineering for health education and care delivery systems: the Smart Condo project. In *Proceedings of the Workshop on Software Engineering in Health Care (ICSE 2009)*, Vancouver, BC, May 2009.

[65] N. Swangmuang and P. Krishnamurthy. An effective location fingerprint model for wireless indoor localization. *Pervasive and Mobile Computing*, 4(6):836–850, 2008.

[66] The VINT project. The ucb/lbnl/vint network simulator ns (version 2). http://mash.cs.berkeley.edu/ns.

[67] C-K. Toh. A novel distributed routing protocol to support ad-hoc mobile computing. In *Proceedings of the IEEE 15th International Phoenix Conference on Computer and Communications*, pages 480–486, Phoenix, USA, March 1996.

[68] C-K. Toh. Associativity-based routing for ad-hoc mobile networks. *Wireless Personal Communications*, 4(2):1–36, 1997.

[69] G. Toussiant. The relative neighborhood graph of finite planar set. *Pattern Recognition*, 12(4):261–268, 1980.

[70] A. Ukil. *Support Vector Machine, Chapter 4, Intelligent Systems and Signal Processing in Power Engineering*. Springer-Verlag: Heidelberg, Germany, 2007.

[71] M. Wallbaum and S. Diepolder. Benchmarking wireless LAN location systems. In *Proceedings of the 2005 Second IEEE International Workshop on Mobile Commerce and Services (WMCS 2005)*, pages 42–51, Munich, Germany, July 2005.

[72] R. Want, V. Falcao, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 40(1):91–102, January 1992.

[73] A. Ward and A. Jones. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47, 1997.

[74] K. Whitehouse, C. Karlof, and D. Culler. A practical evaluation of radio signal strength for ranging-based localization. *ACM SIGMOBILE Mobile Computing and Communications Review*, 11(1):41–52, 2007.

[75] S. Yeh, K. Chang, C. Wu, H. Chu, and J. Hsu. GETA sandals: a footstep location tracking system. *Personal and Ubiquitous Computing*, 11(6):451–463, 2007.

[76] M. Zorzi and R. R. Rao. Geographic random forwarding (GeRaF) for ad hoc and sensor networks: multihop performance. *IEEE Transactions on Mobile Computing*, 2(4):337–348, 2003.