Carnegie Mellon University

CARNEGIE INSTITUTE OF TECHNOLOGY

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF Doctor of Philosophy

TITLE **Controlled-mobile Sensor Networks for Dynamic Sensing & Monitoring Applications**

PRESENTED BY **Aveek Purohit**

ACCEPTED BY THE DEPARTMENT OF

Electrical and Computer Engineering

Pei Zhang — ADVISOR, MAJOR PROFESSOR — 3/28/14 — DATE

Raj Rajkumar — ADVISOR, MAJOR PROFESSOR — 3/28/14 — DATE

Larry Pileggi — DEPARTMENT HEAD — 3/28/14 — DATE

APPROVED BY THE COLLEGE COUNCIL

Vijayakumar Bhagavatula — DEAN — 3/28/14 — DATE

Controlled-mobile Sensor Networks for Dynamic Sensing and Monitoring Applications

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Aveek R Purohit

M.S., Electrical and Computer Engineering, Carnegie Mellon University

B. Tech., Electrical Engineering, MANIT, Bhopal, India

Carnegie Mellon University

Pittsburgh, Pennsylvania

March, 2014

ABSTRACT

Many potential indoor sensing and monitoring applications are characterized by hazardous and constantly-changing operating environments. For example, consider emergency response scenarios such as urban fire rescue. Traditionally, first responders have little access to situational information. In-situ information about the conditions, such as the extent and evolution of the indoor fire, can augment rescue efforts and reduce risk to emergency personnel. Static sensor networks that are pre-deployed or manually deployed have been proposed for such applications, but are less practical due to need for large infrastructure, lack of adaptivity and limited coverage.

The main hypothesis of this thesis is that controlled-mobile networked sensing – the capability of nodes to move as per network needs, is a novel, feasible, and beneficial approach to monitoring dynamic and hazardous environments. Controlled-mobility in sensor networks can provide the desired autonomy and adaptability to overcome the limitations of static sensors.

The research focuses on four of the major challenges in realizing controlled-mobile sensor networking systems:

- Understanding the trade-off between cost, weight, and sensing and actuation capabilities in designing a hardware platform for controlled-mobile sensing together with a complementary firmware architecture.

- Designing simulation environments for controlled-mobile sensing platforms that adequately incorporate both the cyber (network, processing, planning) and physical (motion, environment) components of such systems.

- Investigating the effects of controlled-mobility on network group discovery and maintenance protocols and designing approaches that meet the mobility, latency and energy constraints.

- Exploring novel low-overhead infrastructure-less mechanisms for collaborative coverage, deployment and navigation of resource-constrained controlled-mobile nodes in previously unseen environments.

The thesis validates and evaluates the presented architecture, tools, and algorithms for for controlled-mobile sensing systems through extensive simulations and a real-system testbed implementation. The results show that controlled-mobility is feasible and can enable new class of sensing and monitoring applications.

ACKNOWLEDGEMENTS

First, I would like to thank my advisor Professor Pei Zhang. Thank you for your endless encouragement and help over the last five years. Thank you for giving me the freedom to try out ideas while still keeping me on a cohesive research path. Thank you for challenging me and pushing me to focus and finish things I started. Thank you for making the experience of graduate school so much fun. And thanks for putting up with and correcting my last minute paper drafts. I feel very fortunate to have had you as my mentor and advisor.

I would like to thank my co-advisor Professor Raj Rajkumar. Thank you for being a mentor to me since my early days at Carnegie Mellon. Your sensor networks class and motivating anecdotes were pivotal in me deciding to pursue a doctorate in this research area. Your insightful comments have always challenged me and forced me to look at things in a new light.

A special thanks to all my other committee members. Thank you Professor Anthony Rowe for many informal discussions, your insightful comments on my thesis proposal, and especially for showing me the ropes as a grad student during my first few semesters at Carnegie Mellon. Thank you Dr. Jie Liu for being a guide and mentor during my summer at Microsoft Research and later through my thesis. I've learned a lot about defining and scoping my research from you. My internship at MSR was one of the most educational experiences of my PhD.

A big thank you to Professor Patrick Tague for your valuable insights on research, presentation and graduate school life. Thanks to Professor Dan Siewiorek, Professor Asim Smailagic, Professor Ian Lane, Professor Ying Zhang, Professor Martin Griss and Professor Ed Katz for you encouragement, help and advise at various junctures of my thesis work.

Many thanks to all my labmates and friends at Carnegie Mellon: Karthik Lakshmanan, Zheng Sun, Shijia Pan, Frank Mokaya, Arvind Kandhalu, Senaka Buthpitiya, Faisal Luqman, David Huang, Lu Zheng, Le Nguyen, Rahul Rajan, Bruce DeBruhl, Yuseung Kim, Eric Chen, Vishwanath Raman and Avneesh Saluja, for all the help, advise and fun.

Thank you to my friends, especially Snehal, Sravanthi, Risha, Malte, Chanchala, and Mukta for the food, fun, many debates, and for tolerating my philosophy rants.

Most importantly, I want to thank my parents. Thank you for suffering through my many highs and lows, and encouraging me to think positive, enjoy life and be a better person. Thank you for being my rock. I owe everything I am to you.

Finally, this thesis work was partially supported by the U.S. National Science Foundation under awards CNS-1135874 and CNS-1149611, and by DARPA under grant 1080247-D11AP00265-DOI-ZHANG. I greatly appreciate the financial support.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Many potential sensing and monitoring applications involve harsh and dynamic environments. Consider, for example, indoor emergency response scenarios such as urban fire, earthquakes, gas leaks or hostage situations. On one hand, rescue personnel have little prior information about the scene. On the other hand, evolving adverse conditions such as smoke or structural collapse may impede the planning and co-ordination efforts. Having access to finer-grained information about the environment could potentially enable rescuers to anticipate the evolution of an emergency and adopt effective strategies to minimize injury, loss of life and damage to property. For instance, sensing the temperature profile within a building structure on fire can be used to predict the fire's propagation through building.

Existing solutions to this problem involve installing static sensors (wired or wireless) in the environment [1, 2, 3]. The sensor nodes either must be pre-installed as part of an infrastructure, or be deployed manually by people at the scene. However, such static sensor networks do not address some major challenges that inhibit the widespread adoption and use of such systems, specifically:

- **Infrastructure Independence**. In most approaches, a pre-installed sensing infrastructure is assumed to be in place. The cost of universally creating (placing of nodes) and maintaining (battery replacement) such infrastructure remains high.

- **Robustness**. The dynamic or harsh environment of target application scenarios makes sensing nodes susceptible to damage and failure. A pre-installed static network infrastructure cannot effectively adapt to the destruction of parts of the network and requires high redundancy.

- **Adaptability**. With only a limited number of sensors, spatial coverage of sensed data is also restricted at deployment time. Repositioning or re-tasking nodes as per situation is not possible.

Recent literature has proposed the vision for how mobile sensors or *mobiscopes* can be used to monitor human spaces [4]. One envisioned idea is that of actuated or controlled mobility. Controlled mobility enables a network to mobilize its nodes to suit its demands. Such needs could involve tasks such as data gathering or maintaining network connectivity. Since the network can deploy autonomously, it can replace faulty nodes or reorganize as per its application requirements, addressing many limitations of static sensor networks. Moreover, considering that no universal sensing infrastructure must be installed and maintained, a much larger number of devices can be deployed economically. Sensing nodes like these that combine mobility with low-cost large-scale deployments can provide effective solutions for information gathering in emergency response scenarios.

While robotic platforms [5, 6] exist, in their current form, they are unsuitable for use as *mobile carriers* for sensor nodes in scenarios such as indoor emergency response. This is because the largely single-unit monolithic robot platforms available today, do not provide

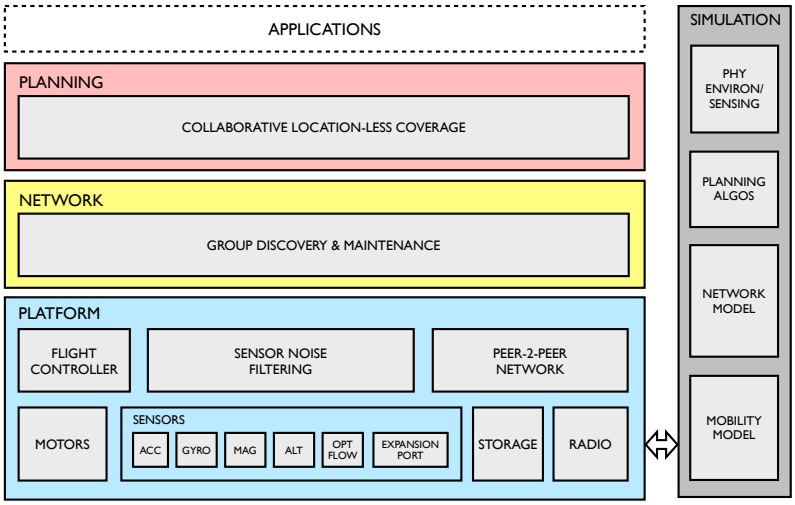


Figure 1.1: The major components of the controlled-mobile sensor networking system.

the robustness and coverage of highly distributed sensor networks. Furthermore, most robot platforms require sensors such as laser range finders or GPS for navigation, making them considerably more expensive than traditional sensor nodes and uneconomical for large or expendable deployments.

The main hypothesis of this thesis is that controlled-mobile networked sensing – the capability of nodes to move as per network needs, is a novel, feasible, and beneficial approach to monitoring dynamic and hazardous environments. This thesis addresses the challenges of dynamic sensing by conducting research on practical system design, software architecture and algorithms required for realizing controlled-mobile sensor networks, and validates them through real implementations. Figure 1.1 shows the major components of the research. Primarily, the research focuses on four major aspects,

- Understanding the trade-off between cost, weight, and sensing and actuation capabilities in designing a hardware platform for controlled-mobile sensing together with a complementary firmware architecture.

- Designing simulation environments for controlled-mobile sensing platforms that adequately incorporate both the cyber (network, processing, planning) and physical (motion, environment) components of such systems.

- Investigating the effects of controlled-mobility on network group discovery and maintenance protocols and designing approaches that meet the mobility, latency and energy constraints.

- Exploring novel low-overhead infrastructure-less mechanisms for collaborative coverage, deployment and navigation of resource-constrained controlled-mobile nodes in previously unseen environments.

1.1 EXAMPLE APPLICATION: EMERGENCY RESPONSE

A controlled-mobile sensor network can be deployed in several sensing and monitoring applications such as survivor search after earthquakes, reconnaissance in urban combat, or indoor toxic plume sensing. This thesis considers an indoor fire emergency monitoring application to evaluate the effectiveness of the proposed platform in providing autonomous, timely, and high fidelity information to aid fire-fighter operations.

Fire response and fire rescue remain extremely challenging operations that annually claim the lives of over 100 firefighters in the United States [7]. Lack of situational information has been identified as a critical limitation for firefighters [1]. On average, firefighters reach sixty-one percent of urban fires in six minutes. However, the fire may spread extensively within that period. Firefighters have little or no knowledge of the location, extent, or advance of the fire in these situations.

Sensors for real-time sensing and prediction of fire propagation are an active area of

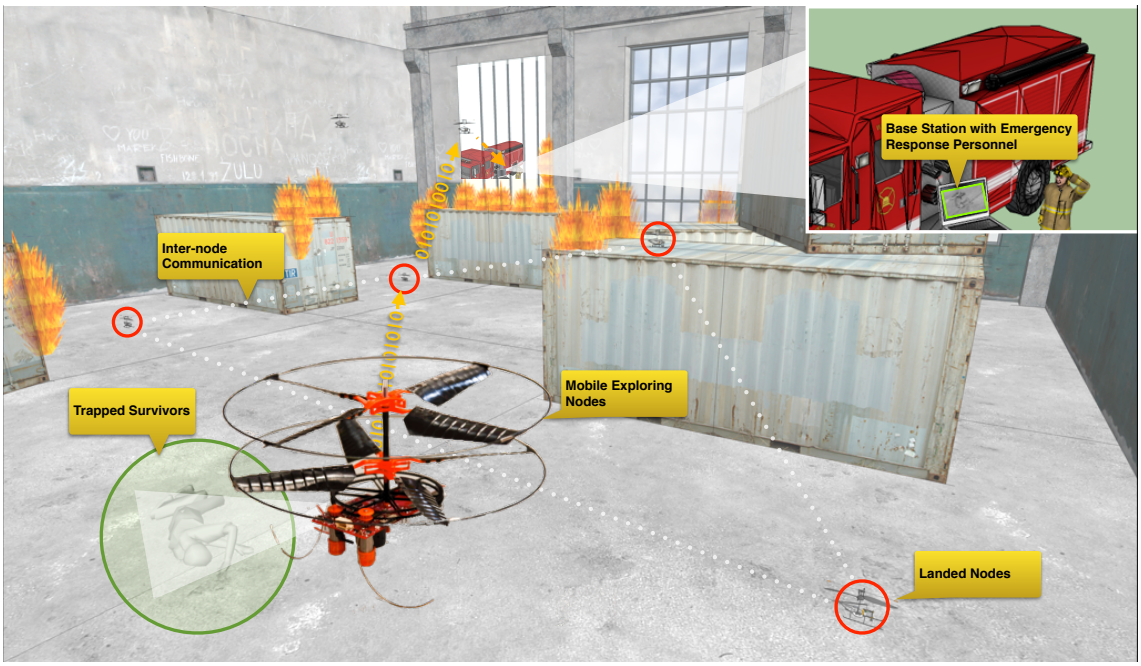


Figure 1.2: Controlled-mobile sensing in a example emergency response scenario.

research. Researchers have proposed fire models to predict the advance of fire from in-situ sensor readings.

This information can be valuable for several fire fighting tasks:

- Firefighters can determine the extent of fire and predict its progression.
- Firefighters can identify regions with possible survivors.
- Firefighters become aware of hazardous areas to avoid.
- Firefighters can effectively plan evacuation routes.

While many sensor networking systems have been proposed for fire monitoring, they are essentially composed of pre-deployed static sensor nodes [1, 3, 8, 9]. Such infrastructure is expensive to deploy and universal adoption remains far into the future. In contrast,

a large number of controlled-mobile nodes can be deployed at the time and location of fire.

Figure 1.2 shows an illustration of a fire monitoring scenario. Firefighters introduce controlled-mobile nodes into connected spaces as they enter the building structure. The nodes autonomously deploy to create a wireless sensing network consisting of landed as well as mobile nodes. Collaboratively, the system achieves a number of sensing objectives, such as –

- Nodes collaboratively search a space for survivors and relay this information to the emergency response personnel (base station).
- Node deploy to assigned locations in the building to monitor numerous parameters such as temperature, gas, pressure, and ceiling height that are invaluable to firefighters.
- Nodes establish a navigation infrastructure and guide personnel to trapped victims or points of interest in absence of accurate maps.

1.2 PLATFORM

My research explores hardware and firmware design architecture for controlled-mobile systems by developing an aerial sensor network platform – SensorFly, for an illustrative indoor emergency response application. The miniature, low-cost sensor platform has capabilities to self deploy, achieve 3-D sensing, and adapt to node and network disruptions in harsh environments. The hardware design trade-offs, the software architecture, and the implementation enables limited-capability nodes to collectively achieve application goals.

1.3 SIMULATION

Cost-effective development, analysis and evaluation of controlled-mobile systems require simulation frameworks that simultaneously model its many computational and physical components. Existing multi-sensor/robot simulation environments are inadequate for this purpose because their primary focus is either on the mechanical aspects of robot movement or on their collective behavior. A controlled-mobile network simulator requires an adequate modeling of motion, wireless networks and sensing applications.

Consequently, this thesis in Chapter 3, proposes a simulation framework that incorporates a radio path loss model, wireless network model, mobility model of a controlled-mobile sensor network, and an application sensing scenario to achieve a more comprehensive representation of such cyber-physical systems. As an example, the framework includes the indoor fire monitoring application scenario incorporating a realistic indoor fire growth model to analyze the effectiveness of the sensing platform.

1.4 GROUP DISCOVERY AND MAINTENANCE PROTOCOL

The nodes of a controlled-mobile network are constantly in motion. As a result, many assumptions of wireless communication protocols for static sensor networks are no longer valid. For example, the overhead of neighbor discovery, route creation and maintenance with respect to data transfer is extremely large as the network configuration constantly changes due to node movement. At the same time, the traditional static sensor-network constraints on energy still exist in this network of mobile nodes.

Therefore, Chapter 4 of this thesis focuses on designing a low energy neighbor discovery, group formation, and group maintenance protocol for controlled-mobile sensor net-

works. The research develops an energy-efficient protocol that combines discovery and maintenance using a collaborative beaconing mechanism to attain better discovery latency and scalability for controlled-mobile networks.

1.5 COLLABORATIVE PLANNING

The application scenarios for controlled-mobile sensor-networks mandate that the network be autonomous and not depend on any existing infrastructure for coverage, deployment and navigation. A major challenge in exploring unknown environments without maps or existing infrastructure is the absence of location information for individual nodes. In absence of location information, network nodes require an in-system mechanism for co-ordinating their paths and collaboratively monitoring environments.

Depending on application objectives, this thesis presents the following novel techniques to achieve spatial co-ordination without depending on external infrastructure or sophisticated sensors and computation capability.

1.5.1 Coverage: Sweep Sensing Single-Space Scenarios

The first technique considers applications where a continuous single space must be covered by a swarm of controlled-mobile sensor nodes. Coverage here is akin to collaboratively sweeping the area, as is typically required in search and rescue mission and fine-grained monitoring tasks.

In Chapter 5, this thesis proposes SugarMap, a self-establishing system that uses approximate motion models of mobile nodes in conjunction with radio signatures from self-deployed stationary anchor nodes to create a common spatial frame of reference for the swarm. Consequently, the system coordinates node movements to reduce sensing overlap

and increase the speed and efficiency of coverage. The system uses particle filters to account for uncertainty in sensors and actuation of controlled-mobile nodes, and incorporates redundancy to guarantee coverage.

1.5.2 Deployment: Monitoring Multi-room Scenarios

The second technique considers multi-room sensing applications that require controlled-mobile nodes to autonomously navigate and deploy at suitable preassigned locations for subsequent monitoring.

In Chapter 6, this thesis introduces DrunkWalk. With help of radio fingerprints from a few landed node acting as radio beacons, the DrunkWalk algorithm detects intersections in trajectories of mobile nodes. The algorithm combines noisy dead-reckoning measurements from multiple nodes at the detected intersections to improve the accuracy of the nodes' location estimates. Most importantly, the algorithm plans intersecting trajectories of controlled-mobile sensor nodes to aid the location estimator and provide desired performance in terms of timeliness and accuracy of deployment. We analyze the performance of our algorithm through large-scale simulations and a real implementation on the SensorFly testbed and show that it compares favorably to existing autonomous deployment strategies.

1.5.3 Navigation: Guiding Personnel in Absence of Maps

Finally, a third approach is presented in Chapter 7 that addresses the scenario where a swarm of nodes explore an area and self-establish a navigation infrastructure to guide other nodes or users (for e.g. fire-fighters) to points of interest within that area. The technique assumes no pre-existing maps or location infrastructure.

With self-deployed anchor nodes, the system learns movement pathways of exploring

mobile nodes in interference-rich indoor environments through radio round-trip time-of-flight and relative compass signatures. These pathways are then used to build a navigable virtual roadmap of the environment that can provide directions to subsequent sensor nodes or users. We present results from a deployment in a complex indoor environment – an operational supermarket to show that SugarTrail system can navigate users with high rate of success and accuracy.

1.6 TAXONOMY OF MOBILITY

Mobility in sensor networks can be of different types depending on factors such as controllability, operating environment and operating medium. In some instances, a new mobility scenario only requires a suitable adjustment to the actuation mechanism of mobile nodes, while at other times a differing mobility scenario presents unique challenges requiring significant changes in approach.

This section discusses the taxonomy of mobility in sensor networks. Further, it examines the scope of my thesis with respect to the applicability of its components to the different mobility classes, as illustrated in Figure 1.3.

The first major classification for mobile sensor networks is based on the controllability of network nodes with respect to their motion. At one extreme, sensing devices such as mobile phones or wildlife tracking tags are mobile, but in an uncontrollable way. *Uncontrolled-mobility* is not addressed in this thesis.

The second class can be defined as *incetivized-mobility* – nodes that initially appear to have uncontrolled mobility but can actually be subject to indirect forms of steering. For example, consider a person carrying a cellphone equipped with sensors that is part of a crowd-sourced sensing application. While the system certainly cannot steer this person

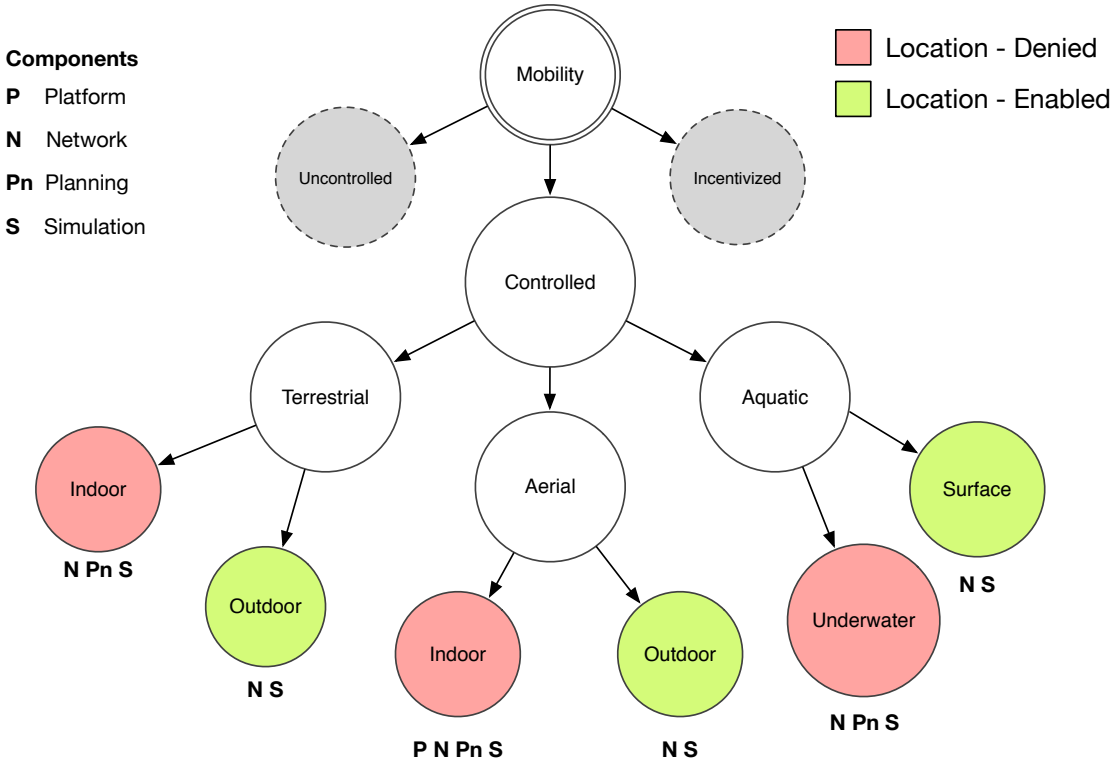


Figure 1.3: Taxonomy of mobility in sensor networks and components of the thesis directly applicable to corresponding mobility classes.

along a particular path, it can offer incentives that increase the likelihood of visiting certain positions. Mobile advertisement, review apps, emergency response, and games have all been used to influence the mobility of users carrying mobile devices.

Finally, autonomous mobile sensor networks such as the SensorFly presented in this thesis, allow the network to control movement of its nodes to the best of its ability. Such networks achieve the overall application goals by balancing multiple tasks assigned to miniature semi-steerable flying devices with heterogeneous sensing capabilities. This thesis largely focuses on *controlled-mobility*. However, the networks ability to control resource-constrained nodes is often not perfect, and the probabilistic approaches developed in this thesis to handle this uncertainty, are also applicable to incetivized mobility scenarios.

The second major classification of controlled-mobility is based on the operating environment of the sensor network. The network can be terrestrial, aerial or aquatic depending on whether it operates on land, air or water. This thesis focuses on design choices, network protocols and algorithms that are applicable irrespective of the actuation mechanism of network nodes used to operate in different mediums. However, certain operating environments present unique challenges that are specifically addressed by this research –

- **Aerial Platforms:** Unlike terrestrial and aquatic nodes, aerial nodes are subject to stricter weight limitations due to the large amount of energy needed for flight. Aerial networks also afford the maximum mobility in many indoor application scenarios. Therefore, as an example of a controlled-mobile sensor network, my thesis develops an aerial platform presenting the corresponding design choices and trade-offs. These trade-offs can be expanded to other mediums with some relaxation in weight and hence sensing constraints.

- **Location-denied Environments:** The second defining feature of the operating en-

vironment is availability of location – whether a network can localize its nodes. In some environments, such as outdoor (land and air) and surface water, technologies such as GPS (Global Positioning System) exist that enable fairly accurate location estimates of network nodes. This greatly simplifies the co-ordination and planning of network tasks for coverage, deployment, and navigation. However, in indoor environments (land and air) or underwater operating scenarios, no ubiquitous localization infrastructure exists. The planning section of this thesis specifically addresses these location-denied environments.

CHAPTER 2

PLATFORM

In this chapter, we present SensorFly, a controlled-mobile aerial sensor network platform for monitoring applications in indoor emergency scenarios. The SensorFly is a miniature ($29g$), low-cost($200), aerial sensor networking platform. To the best of our knowledge, it is significantly smaller than any other realized flying sensor network platform and its cost is comparable to that of low-cost traditional static nodes [10]. We present and evaluate our hardware design choices as well as examine trade-offs that achieve a delicate balance between individual node capability and node resources for miniature aerial controlled-mobile platforms.

This chapter is organized as follows. Section 2.1 describes the hardware design choices and tradeoffs. Section 2.2 presents the firmware architecture. In Section 2.3, we characterize the performance of the platform with regards to sensors and control. Finally, Section 2.4 presents related work.

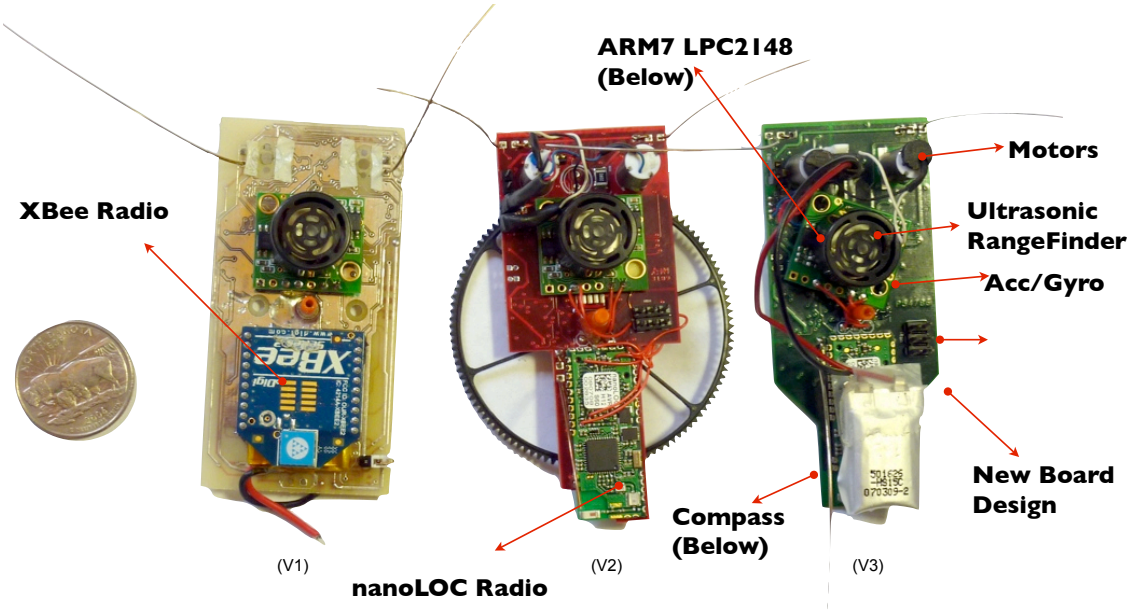


Figure 2.1: Three generations of SensorFly node design.

Figure 2.2: Fourth generation SensorFly node.

2.1 HARDWARE DESIGN

In order to demonstrate the feasibility of the SensorFly system, we have designed and built four generations of SensorFly nodes (Figure 2.1 and Figure 2.2). This section focuses on the hardware design choices and the trade-offs involved in building controlled-mobile aerial sensing platforms.

2.1.1 Key Constraints

The low-cost, low-weight aerial sensing platform presents several constraints and challenges, occupying a new and unique design space. The addition of mobility and control introduces new constraints like *weight*, *sensor interference*, and *higher noise*, to the traditional low-cost COTS(commercial off-the-shelf) based sensor node hardware architectures. Careful consideration is required in new aspects of sensor network hardware design such as *component placement* and *weight balance*. To achieve the desired level of collective system capability given the minimum available resources of individual nodes, a delicate balance must be attained and trade-offs examined. The following factors affect our design approach:

Cost. Low per-device cost allows us to scale up senor node deployments. As a result, for the equivalent cost of an intelligent robot, many more sensor nodes can be used, enabling higher sensing coverage as well as discovery speed. Utilizing a low-cost flight mechanism, common to off-the-shelf RC helicopters, allows us to achieve a prototype cost of about $200. In mass production, similar RC helicopter assemblies are commercially available for about $20 [11]. While larger flying platforms such as the Parrot.AR Drone [12] can provide better capability, they have higher production cost ($\sim$$300) and lower reach due to the bigger form-factor. The navigational capability of individual sen-

Table 2.1: Weight of several possible components of SensorFly nodes. X's mark the components included in the base configuration of SensorFly nodes.

	Component	**Weight**
X	Drive Motors and Propeller Assembly	15 grams
X	130mAh Lithium Polymer Battery	4 grams
	200mAh Lithium Polymer Battery	7 grams
X	Controller Board	10 grams
	Camera Board Add-on	3 grams
	Audio Board Add-on	4 grams
	LED Board Add-on	3 grams
	Ultrasonic Distance Sensor Add-on	4 grams
	Basic SensorFly Total Weight	29 grams
	Absolute Maximum Takeoff Weight	34 grams

sor nodes must be attained through low-cost COTS sensors. A trade-off must be made in forgoing accuracy for deployment scale to better realize our application objectives.

Weight. The miniature aerial platform adds a new metric, weight. The small weight enables longer flight times, greater reach, and better safety for indoor emergency response scenarios. The weight limit is decided by delicate trade-off point. Adding more weight requires bigger motors that in turn require a bigger battery. A larger craft eventually sacrifices the miniature form factor along with the mobility and scalability advantages that it provides. Table 2.1, shows the component-wise weight break-up of the 29 gram SensorFly node.

This weight constraint limits the number of sensors that can be carried. In addition, the weight must be balanced to achieve stability of the node in flight requiring careful component layout and board design.

Energy. Similar to many battery-operated systems, the SensorFly platform is highly energy constrained. Unlike most other sensor systems, however, SensorFly has many different operating modes with vastly different energy characteristics. Table 2.2 shows the

Table 2.2: SensorFly node's operation modes and their power usage breakdown.

Operation	Typical Power Usage
Mobile Mode	6.2W
Stationary Mode	
Data transmission	310mW
Data receive	330mW
Sensing only	225mW
Processing Only Mode	150mW
Idle Mode	1mW

energy consumption characteristics for some of these modes. Of all these modes, the ones involving flight are the most expensive in terms of energy usage.

This further underscores the need for a lighter node, as it enables us to reduce the power consumption of motors. The current battery and weight profile provides about five minutes of airborne flying time. However, by optimizing movements for deployment and reconfiguration, and managing energy consumption when landed and sensing, the overall life of the network can be extended. Utilizing physical characteristics such as the *ground effect* can further reduce movement energy. We evaluate the flight time of the SensorFly nodes in Section 2.3.4.

Interference and Noise. The small form-factor requires placing sensors and components very close to each other on a miniature circuit board. At the same time, use of low-cost brushed motors creates large electromagnetic noise that interferes with the proper operation of the sensors. The placement of sensors and components must be done keeping in mind the effect and nature of induced noise. Additionally, software-filtering approaches are also be needed to obtain better sensor readings.

2.1.2 Design Trade-offs

The resource constraints and capability requirements force our component selections for the SensorFly platform. In this section, we examine our design choices and trade-offs. We also discuss the evolution of the current third generation hardware platform which incorporates learning from previous iterations.

Processor. The processor used in the SensorFly node is the ARM7 based LPC2148. The micro-controller is capable of running at 60 MHz and features 512 KB of Flash memory as well as 42 KB of RAM. It is limited in comparison to the computers that are currently used for most robotics applications. Through various iterations, we have found the processor to be capable of running both the flight control algorithms and the sensor filtering algorithms. The second and third versions of SensorFly nodes incorporate a secondary external processor, an AVR AtMega644, for radio functions and control. By moving the majority of the time critical processes off-board, concurrency is handled better and application integration is simplified. The fourth generation SensorFly platform is equipped with a 8-bit 16Mhz AVR AtMega128rfa1 micro-controller, inertial motion sensors – 3-axis accelerometer and 3-axis gyro, an ultrasonic ranger for altitude estimation, an optical flow sensor for velocity estimation, and a 802.15.4a compatible radio with Round-trip time-of-flight (RToF) measurement capability.

Navigation Sensors. The choice of sensors requires careful consideration, due to the limit on their size and numbers. Navigational sensors have been explored extensively for robotic platforms [13]. Navigational sensors are needed to detect the motion of the node itself as well as sense the environment or other nodes. For motion estimation, miniature MEMS-based inertial sensors such as the accelerometer and gyroscope have become popular in consumer devices like mobile phones, and as a result are commercially available

Table 2.3: Comparison of Navigational Sensors.

Component	Cost	Weight	Accuracy
Accelerometer	Low cost COTS component	Low ¡ 1g	Analog inertial sensor. Unreliable for distance estimation due to accumulating error. Used to detect collisions.
Gyroscope	Low cost COTS component	Low ¡ 1g	Useful for angular velocity measurement. Unreliable for absolute angular position measurement but not affected by magnetic fields. Used for feedback to for yaw controller.
Compass	Low	Low ¡ 1g	Low indoors due to sensitivity to magnetic fields. Error does not accumulate. Used to provide absolute heading.
Ultrasonic Ranger	Low	Medium ∼4g	Fair. Depends on environmental factors such as interference and materials.
Nanotron nanoLoc RToF ranging	Low. Cost is amortized as radio is also used for communication.	Medium. (∼4g)	Better accuracy than RSSI based radio-ranging. Less accurate than ultrasound and laser range finders.
Laser Ranger	Medium	High 50g+	High. Not included due to weight constraints.
Vision	High	High	Accuracy depends on operation scenarios. Less effective in presence of smoke. Needs high processing power.

at low-cost. However, their susceptibility to noise is higher and a trade-off must be made against accuracy. For navigating the environment, a number of higher accuracy range based options such as laser range finders, multiple-ultrasound sensors, camera, and lidar sensors are unsuitable for use in SensorFly. Table 2.3 summarizes the strengths and weaknesses of available sensors on the basis of cost, weight and accuracy. Radio based RF-ranging is an attractive technique, especially because the radio can also be used for communication. However, multipath effects limit the accuracy of radio ranging in indoor environments. This limits precise navigation and movement and calls for collective stochastic exploration approaches. We examine trade-offs further in Section 2.3. Furthermore, these sensors are re-used for multiple purposes as described in Section 2.1.2.1.

In the first version of SensorFly, a two-dimensional compass and a three-dimensional accelerometer were included as flight state sensors. Since the craft is passively stable, the five-degree-of-freedom measurement should be enough to capture the full possible motion of the node. However, the magnetic interference from the motors is large due to the nodes small physical size. This interference during flight renders the readings from the compass

inconsistent and unsuitable for measuring rotation. To improve the flight controls, the later versions include a 3-D compass, a 3-axis accelerometer, as well as a 2-D gyro. In addition, the placement and physical design of the boards mitigate the noise characteristics as described later. These sensors provide a full eight degree-of-freedom measurement. During flight, the gyros provide a rotational sensor immune from magnetic noise, while the compass provides an absolute reading.

Radio. To aid navigational needs, the current version of SensorFly uses the nanoLOC TRX transceiver module [14]. Apart from offering better performance against indoor multi-path fading effects, the radio provides inter-node range estimates based on round-trip time of flight (RToF) computations.

A Digi XBee [15] was used for the V1 of the hardware. This radio is commonly used in sensor networks. Received signal strength indicator (RSSI) was used for range estimation in V1. Several factors prevented the success of this approach. First, in the indoor environment, the radio characteristics were extremely unpredictable, making the multi-path problem pronounced. Second, electromagnetic noise from the motor significantly increased the unpredictability of RSSI measurements. Finally, due to the physical orientations of the helicopter, the antenna cannot be placed at an omni-directional location. This increased the effect of node orientation on the RSSI. These factors prevented the use of RSSI as a viable solution for use in mobile sensor nodes.

Motors. A unique feature of the SensorFly nodes is the mechanical helicopter drives. We use two 7mm core-less motors. These motors drive two coaxial main rotors, and are a significant source of noise in the system, as we shall explore later. While brush-less motors would provide better noise and thrust performance, their size, cost, and circuitry requirements make them ill-suited for our target application.

Furthermore, the low-cost of these core-less motors implies substantial variations in their response to input voltages as well as degradation in performance with use. However, due to the coaxial helicopter design and its passive stability, simple control algorithms can be used to counter-effect these variations. A third motor can be added to the node to provide controlled forward flight. Currently, a weight difference, created by placement of components on the board, is used to provide a constant forward motion, while the craft rotates in small circles to hover in place.

2.1.2.1 Component Reuse

An important strategy for achieving the platform's stringent weight goals is component reuse. Several elements of the node's mechanical and electronic components are selected to be capable of perform more than one function.

On the *electronic hardware* side, the Nanotron nanoLoc radio module enables communication as well as Round-trip Time-of-Flight radio ranging. This ranging capability provides a primitive form of localization and is a substitute for having laser or ultrasound range finders. Although, a trade-off is made in the accuracy of range estimation, the weight and cost constraints are impossible to meet with the alternative ranging mechanisms mentioned.

Sensors such as the accelerometer are used as obstacle sensors, for their ability to detect contact. The lightweight and robust design of the SensorFly nodes enables them to tolerate bumping into obstacles without affecting their flight performance. Thus, dedicated obstacle sensors like infrared or ultrasound based detectors are avoided.

Amongst the *mechanical components*, the blades on the helicopter design serve to protect the body of the node from bumps. As the body is designed to smaller than the blades, it acts as a protective buffer for the on-board electronics.

The circuit board housing the SensorFly electronics, itself acts as the fuselage for the node. This requires careful selection of the board to provide enough rigidity and in turn, prevent stress on the connections and traces when the node lands. At the same time, a thick board adds more weight to the node. In V1 of the design, a 20-mill double layer board is used. While the design is a 1.6x3.1 inches square, the size reduced airflow thereby degrading the lift of the nodes. A 20-mill, 4-layer circuit board shaped as a 'T' was subsequently used to allow more air to flow through from the blades. Due to the reduced per-layer thickness nodes experienced higher failure rate due to stress on the metal traces caused by takeoff and landings. In the third version of SensorFly, a 30-mill board was selected, stress relief added to the 'T' shape and component placement staggered to further reduce single stress points. This redesign greatly improved the fuselage strength.

The legs of the SensorFly node are built with gold plated spring wire and are used as charging terminals as well as landing supports.

Each node can have different weight distributions due to modular design of the SensorFly nodes. For example, different sensors can be added to the extension port of the SensorFly. To counter this effect on the stability of the craft, the battery and the ultrasonic sensor act as adjustable counterweights and are positioned so as achieve the desired node balance.

2.1.2.2 Board Layout

The SensorFly board layout involves careful consideration of the noise characteristics of the components and their weight. The analog sensor components such as the accelerometer and gyroscope must be placed so that they are isolated from the noise sources. The main source of noise in the platform is the high power source and the pair of brushed motors causing high electromagnetic interference. Similarly, the compass must be isolated

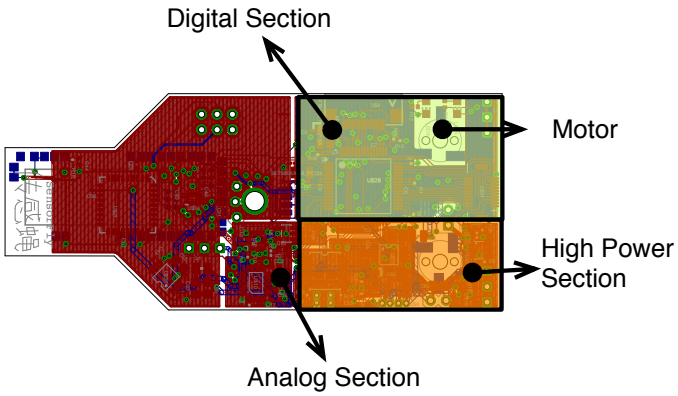


Figure 2.3: The SensorFly node layout. Analog sensors are isolated and the compass placed away from the motors to reduce interference. The digital, analog, and high power sections have separate power supply and the ground plane is interconnected through ferrite beads to filter noise.

from the motors as it is adversely affected by their rotation magnetic field.

Figure 2.3 shows the layout of the SensorFly board. The board is divided into 3 sections. The motors occupy the right (front) end, along with digital section, since the digital components are least affected by the EMI. The 3-D compass is placed towards the tail of the node to minimize the magnetic effect of the motors. The left-most section is the analog section which houses sensors such as the gyro and accelerometer that are adversely affected by noise in the power-lines due to the motors back-emf. The high power section that feeds the motors occupies the bottom-right corner of the board. Each section has separate power lines and a separate ground plane. These are interconnected through ferrite beads to filter out noise.

Another consideration in the placement of sensors is their weight. The board weight must be balanced for stability of flight. A slightly larger weight is maintained towards the front of the node, to substitute the tail-blade of the helicopter design, and enable forward movement. The 4g battery and 4g ultrasound height sensor are used as adjustable counter-weights to balance the weight of the node. The battery placed towards the tail of the node

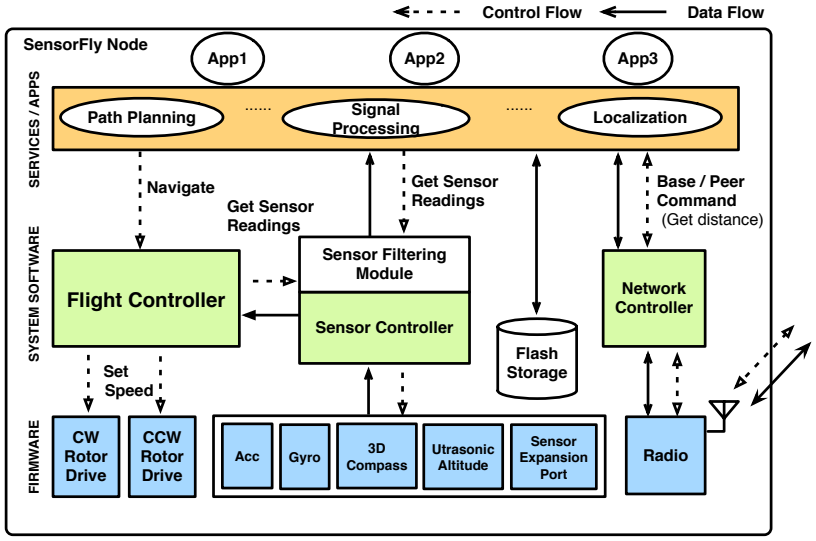


Figure 2.4: SensorFly Node Architecture

while the ultrasound sensor is placed in the middle to counteract the weight of the motors in the front.

2.1.2.3 Extensibility

Several types of sensors can be added to each node using the provided expansion ports, in accordance to the needs of different applications. The expansion port supports serial, SPI, 10-bit parallel, and I2C for sensor interconnection, as well as provides regulated and unregulated power pin through the node battery. Due to weight constraints, we plan to only include one additional sensor module per craft. Several sensors have been designed, including the camera, speaker, microphone, and infrared detectors. We plan to explore the use of additional sensors as our research progresses.

2.2 FIRMWARE ARCHITECTURE

The SensorFly is designed to provide a platform for controlled-mobile and collaborative sensing for emergency response. The SensorFly system architecture, shown in Figure 2.4, consists of the firmware, the node-level system software, customizable network level services, and user application layers. The node system software consists of three major modules corresponding to the capabilities of the platform,

- **The sensor controller** provides access to on-board sensors and expansion ports. Includes filtering modules to mitigate noise caused by motors and motion.
- **The network controller** provides peer-to-peer aggregation and broadcast communication, with support for inter-node range estimation through RToF.
- **The flight controller** provides a high-level navigation API for hover, turn and single-direction flight. A biased random-walk dispersion and exploration algorithm is implemented utilizing the node ranging capability. The algorithm enables nodes to navigate and deploy in unknown environments without need for localization.

2.2.1 Sensor Controller

The sensor controller provides access to the on-board sensors that include the ultrasonic altitude sensor, 3-axis accelerometer, 2-axis gyroscope, and a 3-D electronic compass, as well as to the sensor expansion port. The module provides an API for querying sensors, setting repetitive sample rates, as well as provides in-built filters for noise reduction.

The motors and miniature form factor of the SensorFly nodes affect the performance of sensors such as the compass, as described in Section 2.1. Moreover, some sensors have inherent noise characteristics that can be filtered with knowledge about the dynamics and

mobility of the node. Thus, sensor filtering must be performed for achieving useful capabilities such as altitude control and pose estimation.

The ultrasonic range finder used to measure the altitude of the SensorFly node, is affected by several environmental factors such as absorption characteristics of the ground and interference from other sources such as fluorescent lamps. The sensor controller utilizes the vertical motion dynamics of our platform to discard erroneous readings, using a first order recursive digital filter, as described in Section 2.3.

Similarly, the SensorFly has a 3-axis electronic compass [16] for direction sensing. This is useful in estimating the pose of the node. However, the small dimensions of the SensorFly node require the compass to be placed close permanent magnet DC motors that distort compass reading. Analysis of error induced by the motor's moving magnetic field, points to a symmetric distribution which can be filtered out to a large extent through a moving average filter implemented within the sensor controller module. The window size and other filter parameters are tuned through empirical analysis of sensor data, as detailed in Section 2.3.

The sensor controller also provides a virtual sensor for detecting obstacles using an accelerometer. An algorithm based on thresholds is able to distinguish bump events from the acceleration signal vector magnitude from normal flight.

2.2.2 Network Controller

Our networking implementation supports two major capabilities namely peer-to-peer data communication and radio based inter-node ranging. The SensorFly has a dedicated AVR AtMega644 microcontroller for radio control. This enables better handling of packet transmit-receive and ranging operations that require timely processing. Especially, since

flight control is the highest priority task on the primary microprocessor. The radio module, i.e. the nanoLOC transceiver and the AVR micro-controller, are connected via UART to the primary ARM7 LPC2148 microprocessor. The network controller implements the UART communication protocol, message queues for inbound and outbound packets, and provides a high level API for sending, receiving and forwarding data.

2.2.3 Data Communication

The network protocol for the mobile SensorFly nodes essentially consists of an aggregate and broadcast communication model. The monitoring network seeks to route all sensed data to the base station. Additionally, due to the constant motion of nodes, establishing routes and running explicit node discovery service is impractical. Nodes therefore periodically broadcast messages containing their sensor data. Neighboring nodes, on hearing the broadcast message aggregate the node's sensor data with their message.

Each node's sensor data consists of a sequence identifier and a time-to-live field. The time-to-live is decremented with the number of hops as well as on the expiration of a local time window, to control the time for which stale data propagates in the network. A node's data is propagated by other nodes only if the time-to-live is still not zero. Old sensor data from a node is replaced with fresh data, if it is received before the expiration of the time to live field. This scheme is akin to a controlled reverse-flood of data to the base station.

2.2.4 Ranging

The network controller also provides an API for node-to-node range estimation. The range estimates are used in the exploration algorithm currently employed by SensorFly nodes, which consists of biased-random walks [17] that disperse nodes away from each

other based on their distance from each other. Inter-node ranging is a primitive used by topology estimation schemes.

The radio has the capability to compute distance using a round-trip time-of-flight (RToF) based technique called Symmetric Double Sided Two Way Ranging, or SDS-TWR [18]. The round-trip time-of-flight method measures the elapsed time between the host node sending a data signal to the remote node and receiving an acknowledgment from it. Using the estimated speed of propagation of a typical signal through a medium and the signal turnaround time, i.e. the time for the remote node to send out an acknowledgment packet, the host computes the distance from the remote node. Using physical layer timestamps and hardware-generated acknowledgments, the nanoLOC TRX radio achieves a predictable turnaround time.

Unlike other time-of-flight methods, this method does not require tight clock synchronization between nodes. The time elapsed is computed from timestamps of individual nodes themselves. This removes the need for extra hardware for global time synchronization, which is the source of complexity and higher cost in other systems. Likewise, no special antenna arrays are required such as angle-of-arrival ranging methods. We perform an experimental evaluation of the ranging performance in Section 2.3.2

2.2.5 Flight Controller

A SensorFly's miniature helicopter flying mechanism has many advantages like the ability to takeoff, land and turn in confined indoor spaces, maximizing sensing coverage. Realizing and controlling a helicopter-based sensor-networking platform presents many interesting aspects.

On one hand, the helicopter has highly coupled dynamics. Prior autonomous heli-

copters [19] have required more accurate feedback sensors and computationally expensive algorithms for precise control. On the other hand, the sensor-networking platform has a single CPU with limited computation (60MHz) and memory (42Kb) resources that must perform sensing, control and network processing tasks. Besides, as described before, the performance of control strategies is limited by sensor noise, attributed to node form factor, and cost constraints.

The SensorFly overcomes these challenges by using a lightweight damage-resistant node design and by sacrificing precise control and navigation. The SensorFly system is designed to approach tasks as a networked group that achieves system-wide objectives while tolerating errors in individual node motion. The robust design ensures that nodes can collide with obstacles and still be able to fly, removing the need for precise obstacle avoidance. In fact, the nodes detect obstacles through contact. This allows the flight controller component to implement computationally inexpensive proportional-integral-derivative (PID) control loops that provide *good enough* stability and utilize a biased-random walk approach to navigation.

The flight controller provides a high-level navigation API with commands for hovering, turning, and moving forward. The following sections briefly describe the node dynamics, the navigation and exploration approach used, and the control algorithms.

2.2.6 Altitude and Yaw Control

The SensorFly uses a kind of co-axial counter-rotating dual rotor design, which is passively stable for hover and forward flight [20]. This reduces the number of sensors and computing power required to stabilize it. In addition, this configuration and SensorFly's low weight allow the rotors to operate at relatively low RPM compared to conventional

rotors, making them safer for indoor operation.

The control of the coaxial-helicopter based platform is simple compared to other helicopter design designs, with altitude and yaw being the controllable entities. A constant weight bias towards the front of the node enables it to move in the forward, when the yaw is held constant. The main features of controlling the node are:

- Altitude control is attained by controlling the speed of the two main rotors of the node.
- Yaw control, i.e. turning the helicopter from side to side is achieved by increasing the speed of one rotor and reducing the speed of the other rotor by the same amount.
- Forward flight of the helicopter is attained by placing the center of gravity towards the front of the aircraft. The main rotors follow the tilting of the node body and pull the helicopter forward. To hover, the helicopter rotates in small circles.

This flight controller provides SensorFly nodes with the capability to takeoff and maintains altitude, through a PID controller designed from first principle dynamic models of the node and empirical tuning. Similarly, another PID control loop is implemented for controlling the spin of the node and maintaining pose to achieve forward flight.

2.3 HARDWARE CHARACTERIZATION

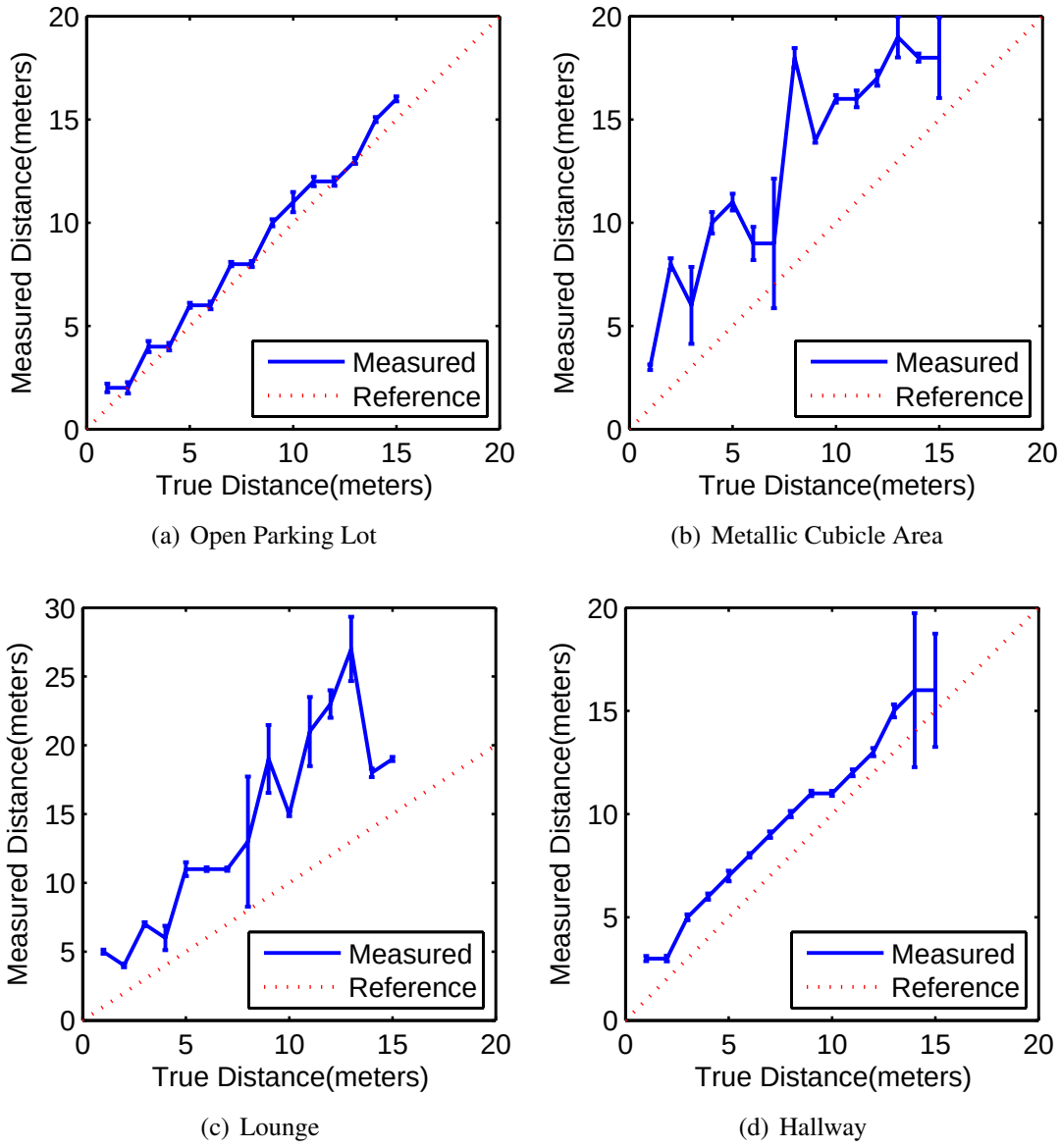
This section describes the implementation and characterizes the distinctive features of the SensorFly platform, in context of the fire monitoring application. Specifically, the altitude sensing, pose estimation, ranging and flight performance detailed and evaluated. Figure 2.5 shows the SensorFly node hovering using the height sensor and algorithms described bellow.

Figure 2.5: The 29g SensorFly node hovering in a hallway.

2.3.1 Height and Orientation Estimation

A 3-axis compass [16] with a maximum sampling rate of 10Hz is used to estimate the pose of the node and estimating yaw. We use the compass to offset the accumulating errors in gyroscope yaw measurements. However, compass readings are affected due its close proximity to motors. A moving average filter is used to minimize the distortion due to the periodic rotating magnetic field of the motor.

In addition to the orientation sensors, the SensorFly V3 nodes use a LV-MaxSonar-EZ1 [21] ultrasonic range finder mounted below the node fuselage, to measure the node's altitude from ground (for annotating sensed data) and provide feedback for altitude control. This is due to the inaccurate distance measurement of the radio described in the following section. The ultrasonic sensor weighs about 4 grams and has a narrow beam width that provides relatively stable readings. However, it is sensitive to building materials and interference from other sources. As described in Section 2.2, a recursive first order digital filter is used to reduce the effect of noise from the observations.



(a) Open Parking Lot

(b) Metallic Cubicle Area

(c) Lounge

(d) Hallway

Figure 2.6: Evaluation of RToF range measurements at different location. (a) Shows outdoor measurements, which characterize error sources other than multipath. (b) (c) and (d) show indoor locations. While the average error is high (4.2m), the measurements have a high correlation (94%) with distance.

2.3.2 Ranging

Radio ranging enables us to attain navigation capabilities, while at the same time, meet our weight and cost constraints. However, our indoor and mobile operating environments introduce multi-path and Non-Line-of-Sight (NLoS) errors. These errors are a feature of the specific space configuration near the node's location and a general model cannot be assumed. Thus, to characterize the accuracy of ranging obtainable, we evaluate the SensorFly node radio ranging in typical operating scenarios.

We performed range measurement tests at four different locations, three indoor, and one outdoors. The indoor locations included a metallic cubicle area, a hallway, and a classroom with furnishings representative of multi-path rich RF propagation environment. An empty parking lot was selected for the outdoor test, to minimize the effect of reflections. At each location, measurements were made for inter-node distances of 1m to 15m, taking 100 readings for each distance.

Outdoor tests illustrated in Figure 2.6(a), show the baseline measurement to be consistently within 1m, with an average of 0.6m. Figure 2.6 also shows indoor measurements for three separate locations. Indoor tests exhibit a much larger error in measured range. Moreover, the errors are not consistent across locations. While a large university hallway, shown in Figure 2.6(d), presented a largely linear relationship between distance and RToF measurements, more constrained cubicle floors and corridors, illustrated in Figure 2.6(b) and 2.6(c), show higher variations. The average error for our test setup was around 4.2m from the true value.

Overall, our experiments indicate that RToF measurements cannot be utilized directly to obtain accurate indoor locations. These errors are caused due to the specific configurations of the environment and a general model cannot be assumed. However, a higher

correlation exists with distance (when compared to other distance metrics like RSSI, hop-count, etc.), enabling the SensorFly nodes to use the measurements for a biased-random walk exploration algorithm. This also provides a better metric that RSSI and hop-count for use in-network localization and topology estimation protocols [22, 23] while meeting the platforms relatively strict cost, weight and accuracy constraints.

2.3.3 Motion

We use PID controllers to implement the desired height and yaw control for SensorFly nodes. PID control is simple, does not require detailed dynamic models, and can be implemented using minimal computing power.

The PID controller for maintaining altitude is based on a first principles model of the SensorFly node given by,

$$X(s) = \left(\frac{\frac{1}{m}}{s^2 + \frac{N}{m}s} \right) F(s). \tag{2.1}$$

where x is the altitude, F is the input, m is the mass of a SensorFly node and N is the coefficient of viscous drag.

Blade rotation causes a torque to be applied to the helicopter body. In stable hovering condition, the top and bottom rotor torques, T_{top} and T_{bot}, should be balanced. The dynamic equation is given by,

$$I \times \alpha = T_{top} - T_{bot}, \tag{2.2}$$

where α is the yaw rate and I is the moment of inertia of the node about the vertical axis through the center of gravity.

A second PID controller, which adjusts the individual rotor speeds while keeping the total thrust constant, enables the craft to hold a certain direction (trim) or turn at a desired rate.

Table 2.4: Performance of flight controller and sensor controller software in achieving stable hover on the SensorFly node.

Set Height	Maximum Over-shoot	Settling Time (70%)	Avg. Steady State Height	Flight Time
1ft.	4ft.	25sec	1.5ft.	6:20min
2ft.	6ft.	40sec	2.5ft.	5:30min
3ft.	6ft.	50sec	3.5ft.	4:50min

2.3.4 Flight Time

Table 2.4 shows the performance of the flight controller and sensor control software in achieving stable hover at a given height. The flight time is lower for hovering at higher target altitudes due to the higher overshoots and settling time. While the current system provides sufficient stability at very low computational cost, better control strategies remain the focus of our work in the future. The flight time of approximately 5 minutes is attainable with the prototype. Optimization of the mechanical design can extend flight time to 15 minutes as has been obtained by similar, although RF-controlled, flying crafts [24].

2.4 RELATED WORK

Research in sensor networks has been actively conducted for the better part of ten years. By far, the most explored area has been fixed networks [10, 25, 26]. These sensing applications have some similar characteristics to SensorFly, in that they are mostly networks with multiple neighbors, and nodes are composed of microcontrollers, radios, and sensors. The main metrics of the system are energy usage, data flow, and data aggregation.

Mobility has been explored in sensor networks largely in context of sensor nodes being carried by human beings or animals [27]. However, unlike SensorFly, the work focuses

mainly on adapting the system to user mobility.

Controlled-mobility has been envisioned by previous work in wireless sensor networks that proposes the idea of using controllably mobile elements in the network to alleviate resource limitations and improve system performance by adapting to deployment demands [28]. Somasundara et al. provide a theoretical analysis of the advantages of controlled-mobility in improving sensing fidelity and node lifetimes with prototype deployments using ground robot platforms [29]. The SensorFly, on the other hand, focuses on the hardware platform designed for enabling controlled-mobile indoor aerial sensing.

Mobility has often been explored as part of robotics research. A segment of research focuses on monolithic or a small team of robots, which may be remote-controlled or autonomous. Typically research on these devices focuses on individual stability and independent navigation, requiring sophisticated sensors [6]. Robotic platforms tend to have a higher per device cost and are economical for deployment in much smaller numbers compared to traditional sensor networks. Consequently, they are constrained in their ability to cover large areas simultaneously and rapidly.

The idea of miniature indoor flying platforms has been proposed before in literature. One work explores using a miniature electric helicopter to combine UAV flocking and wireless cluster computing [30]. Allred et al. present wireless link characterization for a network of semi-autonomous node's for atmospheric plume sensing [31]. SensorFly is the lightest functional system by at least a factor of 5, and targets a different design space. The SensorFly provides a platform for indoor sensing, accomplishing navigation, and networking under strict resource constraints, with a high degree of collaboration.

Wood et al. have worked on flapping-wing micro-mechanical flying devices capable of autonomous flight [32], weighing under 200mg. This is the target hardware platform for the

RoboBees [33] project. We believe these flying mechanisms represent exciting advances, and underscore the need for research into controlled-mobile, highly resource constrained collaborative sensing and coverage algorithms. With a fully functional hardware platform, SensorFly allows us to validate assumptions and determine true tradeoff points in realistic deployments.

2.5 CONCLUSION

In this chapter, we presented a novel 29g controlled-mobile aerial sensor network platform for indoor emergency fire monitoring applications. We identify the challenges in low-cost low-weight aerial sensing platform design and propose an architecture for limited-capability resource-constrained individual sensing nodes.

CHAPTER 3

SIMULATOR

In Chapter 2, we presented an aerial cyber-physical mobile sensing platform for indoor emergency response. As mentioned earlier, we consider an illustrative fire monitoring application to advise our design and evaluation. Cyber-physical systems, which combine sensing and controlled-mobility, enable a network to mobilize its nodes to suit its demands such as data gathering or maintaining network connectivity. They can provide effective solutions for information gathering in emergency response scenarios.

SensorFly nodes deploy autonomously after firefighters have introduced them into the building structure. They can provide real-time data without risk to firefighters. They do not require expensive prior infrastructure and can be deployed at the time and location of fire. In addition, a much larger number of nodes can be deployed at the actual point of emergency as opposed to maintaining universal infrastructures. Finally, the aerial mobility allows better spacial resolution of sensing.

The fire monitoring scenario requires 3-D sensing of vertical temperature profile and ceiling height in building structures, and the SensorFly with its ability to hover vertically can provide higher fidelity data through motion. The development and evaluation of

SensorFly-like systems presents interesting challenges in modeling and simulation methods due to the combination of many disparate computational and physical components.

It is difficult to create actual fire scenarios to evaluate the effectiveness of a mobile sensor network in predicting growth of an indoor fire. A simulation framework aids in analysis and testing of collaborative sensing algorithms of the mobile sensor network, prior to real environmental testing. Existing mobile-robot simulators [34, 35] are inadequate for a realistic analysis of such cyber-physical systems, since they do not –

- Provide a realistic fire growth behavior in the robot's virtual environment.
- Incorporate the wireless link characteristics and network models which are essential to evaluate such a cyber-physical system.

This chapter presents a simulation framework for controlled-mobile cyber-physical systems. It incorporates plug-ins to enable rapid restructuring to suit the particular need of the physical application. For the fire monitoring application, it incorporates an indoor fire propagation model (CFAST), to simulate the advance of fire and smoke in a specified building structure and configuration (arena). The CFAST model provides the temperature and smoke concentration at various regions in time for the specified structure, combustible materials, and fire source. It is widely used by fire investigators, safety officials, engineers, architects and builders. Simultaneously, the framework implements a radio path loss model, wireless network model, and mobility model based on collected experimental parameters and trace data of a controlled-mobile sensor network.

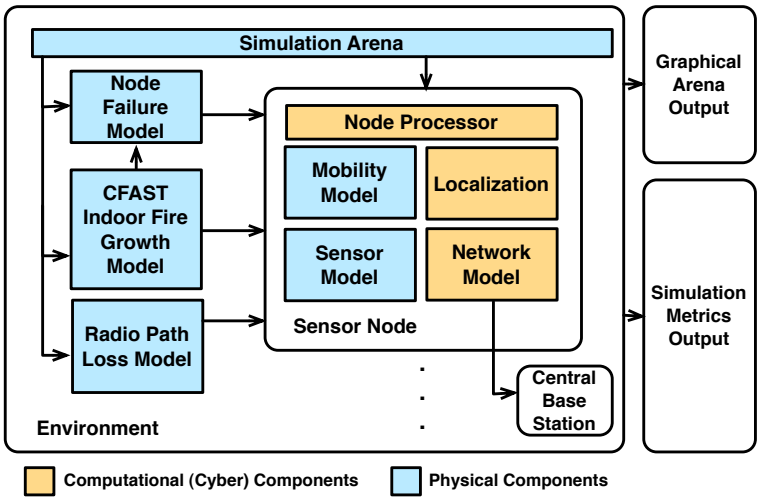


Figure 3.1: Simulation Framework.

3.1 SIMULATION FRAMEWORK

This section describes the major components of the framework as illustrated in Figure 3.1. The framework can be categorized into the environment modules and the sensor node module.

The environment modules describe the physical aspects of the surroundings. The fire growth model provides a realistic prediction of temperature and smoke in the specified simulation arena. The radio path loss model enable wireless link characterization for communication between nodes. The failure model incorporates the failure rate of nodes corresponding to environmental conditions. The simulation arena provides the configuration parameters for the system including the building geometry and material information. The environment modules act as inputs to the mobile sensor node.

The sensor node module consists of modeled components of the hardware such as sensors, radio and processor. In addition, it incorporates some software modules such as localization algorithms and network protocols.

Finally, the simulator outputs a real-time graphical representation of the simulation arena and movement of sensor nodes. In addition, the simulator computes metrics to evaluate the coverage of the deployed network over time. Each module is described in more detail in the following sections.

3.1.1 Indoor Fire Model

Sensors for real-time sensing and prediction of fire propagation is an active area of research. Researchers have proposed fire models to predict the advance of fire from in-situ sensor readings. Such a model is required to provide physical sensing layer for the simulation environment.

Our current implantation includes a popular model called CFAST(Consolidated Model of Fire and Smoke Transport)[36]. CFAST is a zone model where each space is split into two zones with uniform conditions in each zone. The top zone consists of the high temperature gases and smoke, while the lower layer consists of lower temperature gases. The height of the interface between the two zones is called the smoke layer height and this layer descends as smoke builds up in the room. The layer height is used as an indication of the extent of the fire.

The CFAST model is described as an initial value problem for a system of ordinary differential equations. These equations arise from first-principle laws of conservation of mass, the conservation of energy, the ideal gas law and relations for density and internal energy. The model estimates the pressure, layer height and temperatures given the accumulation of mass and enthalpy in the two layers as a function of time. The model takes as input the simulation arena, which is described in Section 3.1.2.

The outputs of CFAST are variables that are needed for estimating the conditions in

a building subject to a fire. These include temperatures of the upper and lower gas layers within each compartment, the surface temperatures within each compartment, and the visible smoke and gas species concentrations within each layer.

3.1.2 Simulation Arena

The simulation arena is the stage for the simulation. This is where the fire originates and propagates as well as the mobile nodes sense temperature and smoke information. The framework requires creating a simulation arena that supplies the configuration parameters required by the CFAST fire growth model, sensor node mobility model and the radio path loss model. The arena parameters include –

- Information about the building geometry such as compartment sizes, materials of construction, and material properties.
- Connections between compartments such as horizontal flow openings such as doors, windows, vertical flow openings in floors and ceilings, and mechanical ventilation connections.
- Fire properties including fire size and species production rates as a function of time.
- Common combustible material such as furniture, defined by their thermal conductivity, specific heat, density, thickness, and burning behavior.
- Placement of initial sensor node positions (entry points).

The parameters are defined through a configuration file which is read by simulation framework, which is implemented in MATLAB. CFAST compatible input files are generated by the MATLAB program.

3.1.3 Mobility Model

The mobility model specifies the algorithm for motion path planning and obstacle detection. The model can be configured to correspond to various platforms. The framework currently models the SensorFly system. The SensorFly platform has the ability to measure height through the ultrasonic range finder, measure its pose through an integrated 3-axis compass, and measure the distance from other nodes through the radio's time-of-flight capability. Assuming these capabilities, a motion model is implemented for the simulated SensorFly node using empirically collected data.

- The SensorFly nodes follow a biased random walk for exploring and deploying in the on-fire building. The nodes move at designated speed in random directions until they are within a minimum specified distance from only one other SensorFly node, to maintain connectivity as well as spread out through the building for exploration.

- Nodes again move in random directions if no other node is within a maximum specified distance or when no data route exists to the base station. This behavior is to guard against partitioning of the network.

- Nodes move vertically at a specified speed in a periodic fashion to obtain readings at different heights. The vertical position corresponding to a given reading is obtained from the height sensor.

This motion model is realistic and easy to implement scheme for our system and requires few physical and sensing assumptions for accurate simulation.

3.2 SENSORS

The framework allows for sensors that can sense the variables output by the fire model, as well as obstacles in the simulation arena. The virtual SensorFly node is equipped with sensors corresponding to the actual hardware, including, an ultrasound height sensor, an electronic compass, a gyroscope and an accelerometer. It includes an error model for sensors which is currently assumed to have a normal distribution based on empirical observations on our hardware.

3.2.1 Network Model

The framework implements an aggregate and broadcast communication model for communication. The monitoring nodes seek to route all sensed data to the base station. Accounting for constant motion of nodes, establishing routes and running explicit node discovery service is considered impractical and avoided. Nodes are simulated to periodically broadcast messages containing their sensor data.

Neighboring nodes, on hearing the broadcast message aggregate the node's sensor data with their message. Each node's sensor data consists of a sequence identifier and a time-to-live field. The time-to-live is decremented with the number of hops as well as on the expiration of a local time window, to control the time for which stale data propagates in the network. A node's data is propagated by other nodes only if the time-to-live is still not zero. Old sensor data from a node is replaced with fresh data, if it is received before the expiration of the time to live field. This scheme is akin to a controlled reverse-flood of data to the base station.

3.2.2 Radio Path Loss Model

A radio path loss model is required to provide a more realistic characterization of the wireless links for node-to-node communication. As complex models would be too computationally expensive and unfeasible, we use shadowing with a path loss exponent of 3 as the radio link model for simulations. This is similar to that employed by previous indoor fire monitoring work [3] and is an estimate for a single floor multi-room scenario [37].

3.2.3 Node Failure Model

The framework provides a configurable stochastic model for node failures. It allows a failure rate to be specified corresponding to maximum threshold for environmental variables such as temperature and gas concentrations, as predicted by CFAST. This allows testing of redundancy schemes and sensing performance for a realistic deployment.

3.3 OUTPUT

The simulator simulates the motion of the nodes at runtime. Detailed analysis of cyber and physical characteristics can be evaluated at each time step. However, in order to aid high level development of algorithms, we developed two metrics to describe the system. With the CFAST predictions data as ground truth, the simulator metrics give a measure of the accuracy of sensed fire growth.

1. **Average Model Error**. This is defined as the root-mean-square error of the predicted model obtained by interpolating the discrete sensor readings reported by the static nodes and mobile SensorFly nodes. The readings are interpolated in both the height and time dimensions to the maximum resolution of the CFAST simulator, 0.1m in

height and 10 seconds in time. This metric captures the performance of the system in terms of predicting an accurate model from sensed data.

2. **Spatio-Temporal Percentage Coverage**. The static nodes have limited resolution in the space dimension since only a few nodes can be economically installed as part of a universal infrastructure. Conversely, the mobile SensorFly nodes by virtue of being introduced into the environment and relying on autonomous means to deploy are constrained in the temporal dimension. That is, the mobile nodes may not be available at the desired location. The spatio-temporal percentage coverage captures the effect of both these characteristics. It is defined as the ratio of the number of sensor readings in the height-time plane to the maximum possible resolution obtainable.

These metrics can be computed from deployments with and without accounting for the effect of network disruptions and node failures.

3.4 EXAMPLE SIMULATION SCENARIO AND DISCUSSION

We present an illustrative fire scenario simulation to show an example comparison of performance between autonomously deployed mobile SensorFly nodes and a statically pre-deployed network of sensors. The simulation scenario considered is a 3-room building as shown in Figure 3.2. The simulator permits evaluation of sensing performance as a function of deployed network size and arena geometry using various network and physical parameters.

The CFAST fire simulation runs for a total of $t = 1800sec$, providing fire evolution data such as layer interface height, temperature, pressure and gaseous composition along the height of each room. Fires are set in the Kitchen and Living room compartments at

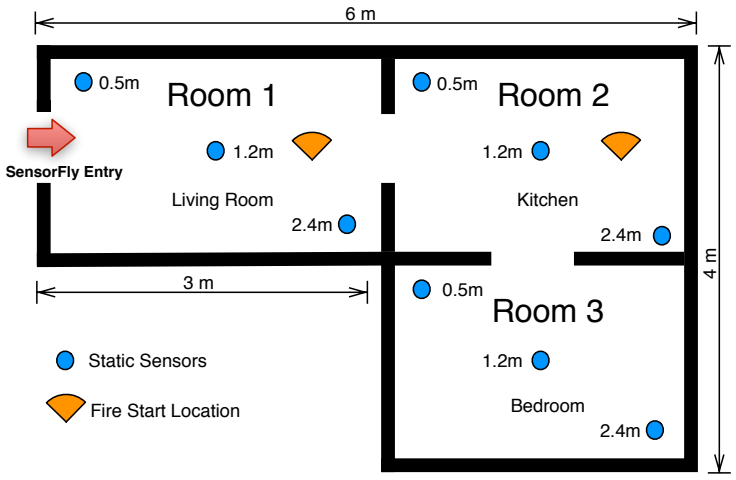
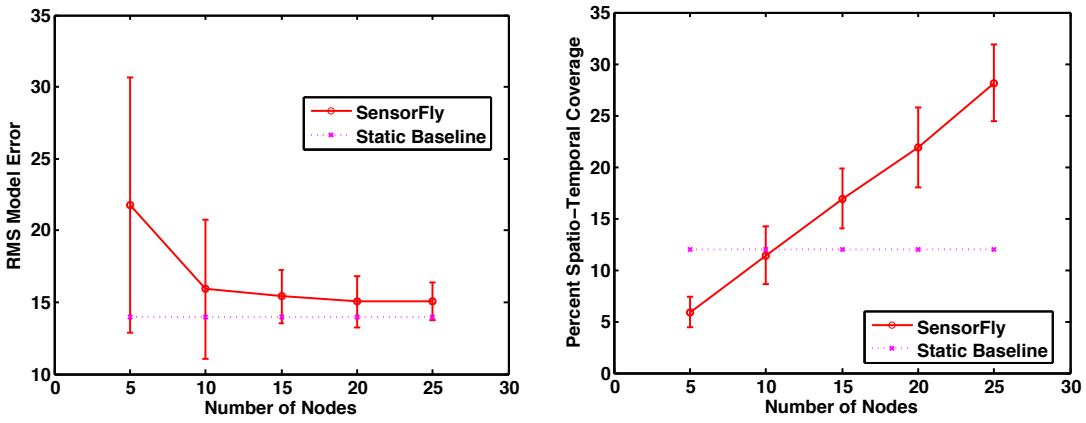


Figure 3.2: Arena for SensorFly and CFAST fire simulations. The figure shows the building geometry, the placement of nodes for the static network, the entry point for mobile SensorFly nodes and the point where fires are initiated at $t = 0sec$.

$t = 0sec$. Each room has typical furniture with their combustible properties as provided in the simulation environment. The CFAST zone model assumes the conditions at a certain height of the room to be uniform. Therefore, only variations along the vertical dimension in the building compartments are of interest in sensor placement.

The static network's location is pre-defined in the simulator consisting of 3 nodes placed at heights of 0.5m, 1.2m, and 2.4m in each of the 3 rooms. The nodes measure temperature at 10 second intervals and route the data back to the base node at the entrance of the building structure. The SensorFly nodes are introduced into the environment at $t = 0sec$ seconds into the simulation arena. The nodes deploy autonomously and route back sensed temperature readings to the base station also at 10 second intervals, moving vertically to obtain data at different heights. Figure 3.2 also shows the placement of sensors in the simulation arena as well as the entry point for SensorFly mobile nodes. The mobility models, network protocol, and radio link characteristics used for the simulation as described in Section 3.1.



(a) (b)

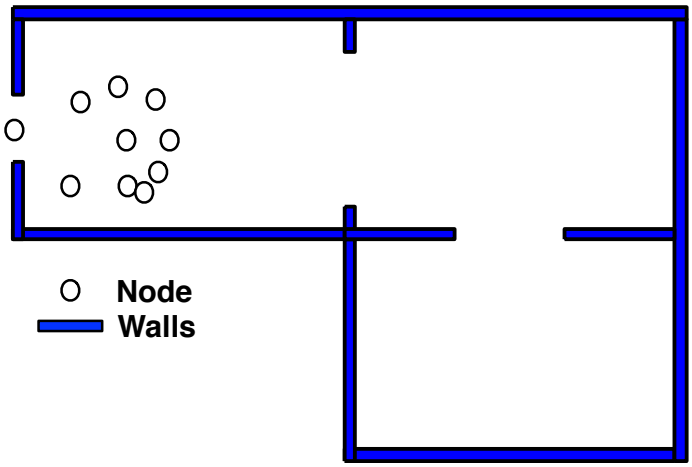
Figure 3.3: (a) Average error in predicted model shown as a function of the size of SensorFly deployment. The error decreases with increase in deployment size, though flattening out at 10 node sized deployments. This is similar to the size of static deployments and the difference in error between pre-deployed static nodes and autonomously deployed SensorFly nodes is about 14%. (b) The percentage coverage in time and height dimensions is shown as a function of deployment size. The simulator allows analysis of the coverage with deployment size. The coverage increases almost linearly as larger number of mobile nodes provide higher resolution data in space as well as time dimension. The coverage from static deployments is constant and similar to SensorFly deployments of similar size. The plots show the average over 50 simulation runs while the error bars represent the standard deviation observed.

The simulation starts at $t = 0$ with a fully deployed static network with nodes in all 3-rooms, and all SensorFly nodes introduced into room 1.

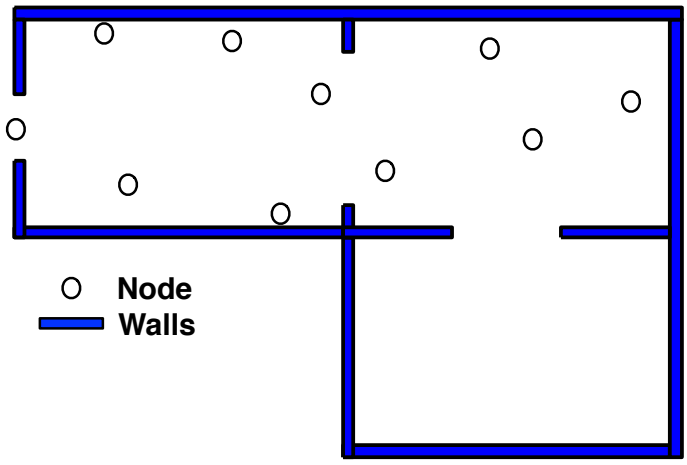
The simulator provides a graphical output of the arena and node motion within it. Figure 3.4(a), shows a snapshot of the graphical output of the simulator at time $t = 0secs$, when all the nodes are at their specified initial positions. Figure 3.4(b), shows an intermediate state at $t = 300secs$, where nodes have dispersed into 2 of the 3 rooms. Figure 3.4(c), shows a snapshot of the simulation at time $t = 900secs$, with the nodes having dispersed into the three rooms following the mobility algorithm specified in Section 3.1.

In addition, the simulator outputs the evaluation metrics and their variation with size of the deployment and failures introduced. Figure 3.3 (a), shows the average model error as the number of nodes introduced into the arena is increased. The error from the static deployment of 9 nodes, 3 in each room, is shown for comparison. The error is large for a small number of nodes primarily because of the lower coverage. The error and the variance in error decreases as the number of nodes in increased. The error stabilizes at 10 nodes, about the same as the size of the static node deployment. The difference in error between the autonomous SensorFly deployment and the base line static one is about 14% with similar sized deployments. It decreases further as the number of nodes increase. From the simulation graphical output, the stabilization can be attributed to the fact that once nodes are present in all 3 rooms, additional nodes increase redundancy but improve the error only slightly given the uniform model. A scenario with higher resolution sensing needs will benefit more with a larger number of nodes.

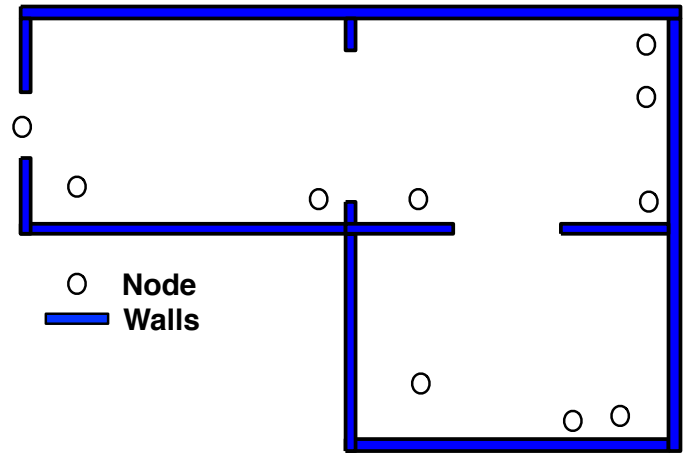
Figure 3.3 (b), shows the percentage coverage of SensorFly deployments of various sizes. The percentage coverage, as defined earlier, is the total points, in the height and time dimension, available from sensing to the maximum points provided by the CFAST



(a)



(b)



(c)

Figure 3.4: (a) Snapshot of simulation at $t = 0secs$. Shows the initial positions of the nodes in the arena. (b) Simulation snapshot at $t = 300secs$. Shows nodes have dispersed into 2 of the 3 rooms. (c) Simulation snapshot at $t = 900secs$. Shows nodes dispersed into all 3 rooms according to the mobility model described in Section 3.1.

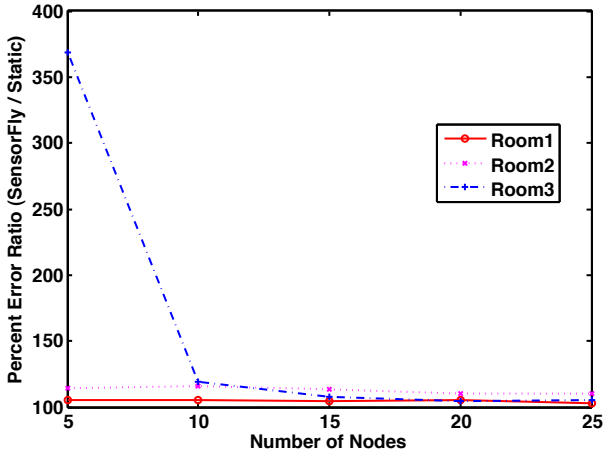


Figure 3.5: The figure shows the ratio of SensorFly errors to static node error by compartment location. SensorFly nodes achieves similar performance closer to areas where they are introduced. For Room 3, which is farther away and obstructed, the error is higher for small sized deployments.

simulator. In a real world scenario, the resolution would be infinite. However, this baseline corresponds to an ideal case sensing resolution suitable for the phenomena being sensed and its spatial distribution. The coverage increases with the number of nodes as expected, with SensorFly coverage being about equal to that of static nodes at the deployment size of 10 nodes. The coverage increases almost linearly thereafter. This is because the mobile SensorFly nodes can provide very high resolution in the height dimension. However, when the number of nodes is low, the nodes are slower to enter into a compartment due to the current mobility model and therefore have less temporal coverage. This is arguably due to the biased random exploration and deployment method employed.

The plots show the average over 50 simulation runs while the error bars represent the standard deviation observed.

The simulator also provides a break up of errors in different building compartments that illustrates the nature of mobile deployments. Figure 3.5, shows the ratio of the error of

SensorFly deployments to the baseline static deployments on a room-by-room basis. The geometry of the deployment arena quite obviously has an effect on the accuracy of the predicted model in case of SensorFly. As the SensorFly nodes are introduced into Room 1 (Living Room), the error matches or is lesser than the static deployment case due to the higher resolution possible due to mobility. Room 3 is farthest from the point of entry and obstructed by two doorways. Therefore, smaller deployments of autonomously deployed SensorFly nodes have large errors as compared to static nodes in Room 3. However, as the number of nodes increase, the nodes spread out faster and achieve better coverage in time, eventually matching performance shown by static nodes.

3.5 RELATED WORK

A number of multi-robot simulation environments exist for evaluating robotic platforms. Player/ Stage project [34] is an open source software project for robotic systems research. Stage is a player plugin that simulates a population of mobile robots, sensors and objects in a 2D bitmapped environment. Player is a networked device repository server to popular robotic platforms. It comprises of middleware to enable robot-server and robot-robot communications. However, it focuses on realistic robot hardware simulation and does not incorporate realistic fire growth models or wireless link characteristics.

Similarly, Sinbad [35] is an open source 3D java-based robot simulator for scientific research and learning. The simulator is characterized as a low fidelity but complex 3D scene modelling using built in simulation engine (Simbad) and provides a collection of tools for Evolutionary Robotics. The main design focus of this simulator is to offer a simple environment to study AI algorithms and machine learning in autonomous mobile robotics set up.

The framework presented in this chapter, combines a multi-robot simulator with fire-growth models and communication models to enable a more realistic evaluate of a cyber-physical system for fire monitoring, which is not adequately possible with existing packages.

3.6 CONCLUSION

This chapter presents a simulation framework for controlled-mobile cyber-physical systems employed in fire monitoring applications. The notable advantage of the framework is its ability to incorporate a physical indoor fire growth model (CFAST), with a radio path loss model, wireless network model, and mobility model of a controlled-mobile sensor network. This provides a more realistic simulation environment that can simultaneously model the many disparate computational and physical components contained in such cyber-physical systems.

CHAPTER 4

NETWORK

This chapter presents a discovery and network protocol - WiFlock, for communication in controlled-mobile systems that takes into account the transient nature of wireless links when energy-constrained nodes are in constant motion such as swarm of SensorFly nodes. The protocol has two goals - 1) to achieve a trade-off between energy and latency for discovery mechanism to enable nodes to constantly update their neighborhood information and 2) to enable a group management to leverage a transient group for data transmission at lower energy and latency. To account for the constant change in group membership the two goals are achieved through a single unified mechanism.

Neighbor discovery and group maintenance are not unique to mobile sensor networks. In any ad hoc sensor deployment, the network needs to initiate neighborhood tables and repair link connectivity over time. To encourage nodes to sleep for energy saving, a neighbor discovery protocol usually incorporates long preambles sent by the to-be-discovered node, and periodic waking up and listening by the discovery nodes. These same protocols can be applicable to networks with slow mobility or high energy budgets, where the nodes can afford to communicate frequently.

The challenge comes when the network exhibits "opportunistic flocking" behavior, where scattered mobile nodes occasionally come together for a period of time and then the groups disaggregate again, as in collaborating controlled-mobile SensorFly nodes with heterogeneous capabilities. When the system is expected to have a long lifetime with a constrained energy budget, the nodes need to carefully trade off between duty cycling and latency.

Traditionally, group discovery and maintenance are done in two separate phases: first, a neighbor discovery phase where each node builds its neighborhood table, and then a group maintenance protocol for propagating and aggregating neighborhood tables into a group table. Disco [38] and U-Connect [39] studied low energy neighbor discovery in mobile sensor networks. Under both approaches, nodes asynchronously beacon for advertising their existence, and listen for receiving beacons, although they differ in beacon and listen schedule. Group maintenance, when the network cannot afford time synchronization, is usually built on top of low power listening such as BMAC. In BMAC, a node sends long preambles to ensure that the targeted neighbor wakes up, and establishes a point-to-point communication link.

In this chapter, we argue that in flocking sensor networks, we have an opportunity to design neighbor discovery and group maintenance protocols jointly. Since these protocols must be executed all the time, a combined protocol can achieve high energy efficiency and low latency. We describe WiFlock that uses a unified and collaborative beaconing mechanism to achieve both discovery and group maintenance goals. The collaborative nature differentiates WiFlock from previous asynchronous neighbor discovery protocols, which mainly considered pair-wise discovery between two nodes. In fact, since group formation is the ultimate goal, we show that it is efficient to perform *one-way* discovery

and then use collaborative broadcasting for propagating the information.

The WiFlock protocol uses several techniques for performance improvement.

- When a node is alone, it needs to periodically wake up and check if it has any neighbors. If the node spends majority of its life in the alone mode, then reducing the wake up duty cycle directly translates to increasing node lifetime. We show that WiFlock can achieve $80\mu S$ carrier sense duration in practical environments with moderate interference. This is $\sim 3\times$ less than the state of the art $250\mu S$ reported by U-Connect and leads to much lower node duty cycles.

- When nodes are in a flock, their activities can be synchronized to speed up the propagation of neighborhood and group membership information. The whole network can quickly react to changes in terms of nodes joining and leaving the group. To achieve this, we use a distributed coordination to achieve synchronized listening and evenly spaced transmitting (SLEST) among groups of nodes. We show that it is a lightweight yet scalable technique for joint discovery and group management.

- When nodes within a flock exchange messages for maintaining group membership, WiFlock embeds data messages such as neighbor tables in the long beacon itself, so receivers do not need to keep waiting for the end of the long preambles to establish two way communication.

As a result, WiFlock easily accommodates frequent node mobility by combining the neighbor discovery and group maintenance in to a single long running service, rather than treating them as two distinct and sequential phases. WiFlock is a highly efficient and scalable protocol that can achieve node duty cycles as low as 0.2%. Unlike some of the existing solutions, where the performance degrades quickly as the group size increases, the

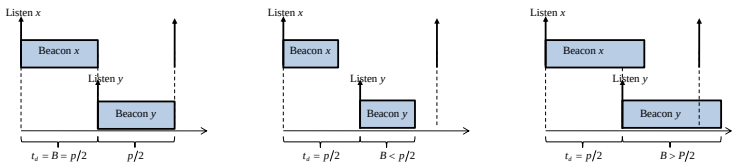


Figure 4.1: Impact of different beacon durations B vs. beacon period P on ability to discover neighbors and the average energy when two node beacons are separated by $t_d = p/2$. (a) $B = p/2$ (b) $B < p/2$, and (d) $B > p/2$.

proposed solution can easily accommodate large groups. We evaluate WiFlock on a test bed of 50 sensor nodes and show that a group can be formed within 3 minutes with 0.2% total duty cycle by using a 80μs listening duration under moderate WiFi interference.

4.1 NEIGHBOR DISCOVERY

To accommodate node mobility, the basic operation in neighbor discovery and group management is advertising the node's existence and listening to discover neighbors. However, when to perform these operations greatly impacts the node's energy expense and the network's reactiveness to changes. Therefore, we derive the optimal beaconing duration for advertisement and wakeup period for listening. In this section, we assume that the nodes do not belong to any groups. For many applications, a node spends majority of its time in such a mode.

Nodes use low-power sleep state to reduce energy consumption. The basic operations of a neighbor discovery protocol consists of two parts:

- **beaconing**: In order to advertise its existence, a node wakes up periodically, every q seconds, and transmits a beacon of duration B.

- **listening**: For detecting other nodes' beacons, the node also wakes up periodically,

at the period p, and listen to the RF channel for a duration D to detect any beacons from neighboring nodes.

In addition to periodic listening, discovering unknown neighbors involves periodic beaconing. Thus it is advantageous to seek ways to minimize *idle transmit*. We carefully pick the optimal length and period of the transmit beacon to reduce our average power consumption.

Our neighbor discovery protocol builds on the previous work by Kandhalu `et al.` [39], where they introduced the U-connect neighbor discovery protocol. Similar to U-connect, we choose the beacon duration B as:

$$B = \frac{p}{2},$$

which is the minimum length of the beacon that guarantees that at least one node from a pair of nodes will detect the presence of the other. We call this *one-way* discovery, in contrast to pair-wise discovery where both parties discover their counterparts.

Fig. 4.1 shows the intuition behind this choice. Consider two nodes x and y transmitting the beacons at times t_x and t_y. The time difference is $t_d = t_y - t_x$. Here we consider the case where $t_d < p$; due to periodicity, the same argument applies when $t'_d = n \cdot p + t_d$, where n is an integer. We assume that D is arbitrary small.

In Fig. 4.1(a), if $B = \frac{p}{2}$ then $t_d = \frac{p}{2}$. Here the beacons of each node barely overlap with the listening so that one (or both) of nodes can hear each other. If $0 \leq t_d < \frac{p}{2}$, then node x can hear node y's beacon, while for $\frac{p}{2} < t_d \leq p$, node y can hear node x's beacon. Thus $B = \frac{p}{2}$ guarantees one-way discovery.

As shown in Fig. 4.1(b), when $B < \frac{p}{2}$ none of the nodes can hear the other. On the other hand, as in Fig. 4.1(c), if $B > \frac{p}{2}$, either one of the nodes or both of the nodes can

hear each other depending on value of t_d. Although this still guarantees discovery, energy is wasted on the extra overlapping. Of course, in practice, the B can be slighter longer than $p/2$ to accommodate uneven clock drifts in the two nodes.

Note that, when $B = p$, the node behavior degenerates to low power listening (LPL) [40], which guarantees pair-wise discovery and that the listener can discover the advertiser within one beacon period. However the energy expense is significantly higher. The flip side of one-way discovery is that the nodes cannot establish two-way communication immediately. This asymmetry has ramifications on efficient group maintenance mechanisms, which we will elaborate in section 4.2.

Having decided on the beaconing length, we examine how to select the node beaconing period q to minimize detection latency given an energy budget.

Assuming that transmit and receive energy consumptions of a node are approximately the same [41], we analyze the energy consumption of the node using the its transmit and receive duty cycles. We define:

- receive duty cycle: $c_{rx} = \frac{D}{p}$;
- transmit duty cycle: $c_{tx} = \frac{B}{q} = \frac{p}{2q}$;
- total duty cycle: $c = c_{rx} + c_{tx}$; and
- receive duty cycle ratio: $r = \frac{c_{rx}}{c}$.

Then we can show that the worst-case discovery latency, which is equal to the beaconing period q, can be expressed as:

$$q = \frac{D}{2c^2 r(1-r)} \tag{4.1}$$

From the first and second derivatives of q w.r.t. r, we obtain that when $r = 1/2$, q

takes its minimum. Thus, for a given duty cycle (or energy budget), the minimum discovery latency is archived when transmit and receive duty cycles are equal. In this case, the beaconing period q is given by;

$$q = \frac{p^2}{2D} \tag{4.2}$$

This result establishes two principles. First, it gives the optimal beaconing period that must be chosen. Under the optimal beaconing period, the node spends half the energy listening and half the energy advertising. Second, it shows that the smaller we can make the listening period D the better. Small D means less energy consumed for each wake up for listening and more frequently a node can wake up, and therefore the less often an advertiser needs to beacon.

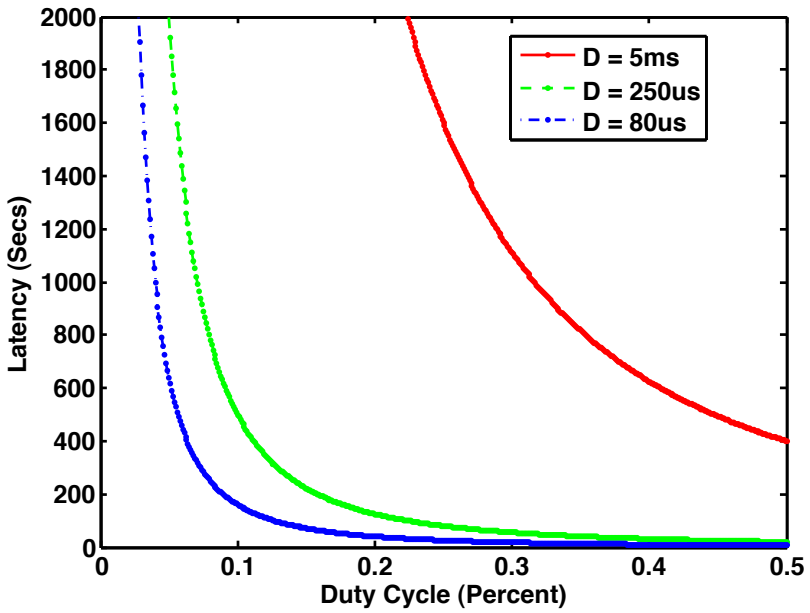


Figure 4.2: The figure shows the theoretical discovery latency for 2 nodes, with 5ms(minimum in Disco), 250μs (minimum in U-Connect) and 80μs(in WiFlock) wakeup duration, to discover each other as a function of their percentage duty cycle of operation.

Figure 4.2 shows a comparison of the theoretical neighbor discovery latency, vs. the

duty cycle, for different wakeup durations D, with $r = 1/2$. We observe, that for total duty cycles $c < 0.5\%$, the discovery latency quickly grows as the wakeup duration increases. We compare the minimum wakeup durations supported by prior neighbor discovery protocols Disco (5ms) and U-Connect (250μs) to that supported by WiFlock (80μs). This confirms that to achieve very small duty cycles, the focus must be to reduce the wake up duration D, to attain reasonable discovery latencies. Hence, when implementing WiFlock, we seek practical ways to shorten D by reducing carrier sensing time (section 4.3.2).

Supporting Multiple Duty cycles

Although the above discussion assumes nodes with the same energy budget and the same duty cycle, WiFlock can be easily extended to nodes with different energy capacity. A moving vehicle or a robot is likely to be willing to use higher duty cycles, while comparatively energy constrained nodes, such as an energy scavenging sensor node, is likely to tolerate long discovery latencies to preserve energy.

Nodes that are willing to spend more energy can transmit longer beacons and listen more often. For example, a node can double the length of its beacon duration to p, which causes the node power consumption to increase by 50%. Since their energy constrained neighbors would sample the channel at frequency p, this may make some of the neighbor discovers bidirectional, reducing the overall discovery time. Similarly, doubling the wake up frequency also increases the power by 50%, which can also decrease discovery time due to removal of some unidirectional discoveries. If a node doubles both beacon length and removal frequency, all its neighbor discoveries will become bidirectional, which will reduce the discovery time further. If a node wants to have finer-grained control over the power consumption, it can selectively increase the beacon length or sampling frequency

only for some of the beacons and sample intervals. If the node can afford to more than double its minimum energy consumption, it may increase the beacon frequency, while paying attention to the possibility of increased beacon collisions.

A node running WiFlock may reduce energy consumption by reducing the beaconing frequency and selectively turning off some of the sampling events. However, persistently reducing the beaconing frequency, and specially the sampling rate, may make neighbor discovery time arbitrarily long. In the rest of the discussion we assume that all nodes involved in group formation and maintenance agree with the same minimum duty cycle (such as 0.2% duty cycle in our evaluation).

4.2 GROUP-WIDE COORDINATION

While the standalone beaconing discussed above is a natural extension of U-connect, it does not solve the group maintenance problem completely. First, the one-way neighbor discovery guarantees only one of the nodes in a neighboring pair to discover the neighboring relationship. This unidirectional nature of discovery does not allow even adjacent nodes to discover all their neighbors. One way to overcome this is to implement explicit handshake mechanisms, where a node must schedule a special acknowledgment frame after every beacon where listeners can register their presence, as employed in [38, 39]. The downside is that this approach is not efficient for large group formation. Point-to-point communication between all group nodes gives rise to collisions and high message overhead for larger groups.

Second, a group may span more than one communication hop. The node-to-node neighbor discovery does not provide a mechanism to propagate neighbor information over multiple hops. Traditional solutions typically employ another group management layer after

neighbors are discovered. The group management layer may flood the network of known nodes periodically to keep the membership information fresh and deal with nodes that move away from the group. However, at the same time, since the network is mobile, all nodes still need to run the discovery protocol for new nodes to join the group.

In WiFlock, the continuous group maintenance and neighbor discovery is combined into a single beacon schedule to achieve high energy-efficiency and scalability. To take advantage that multiple nodes are within a small neighborhood as in the flocking configuration, we exploit a collaborative technique for performance improvement: Synchronized Listening and Evenly-Spaced Transmitting (or SLEST for short). Throughout this section, our analysis assumes perfect radio channels. We consider realistic radio channels in our implementation and real test bed evaluation.

4.2.1 Synchronized Listening

Nodes participating in neighbor discovery transmit periodic beacons of duration $p/2$. These beacons are typically large enough to accommodate more than an entire packet. For example, in our implementation with 0.2% duty cycling, the beacon lasts for 40ms, which can accommodate twenty 128Byte long RF packets at a 256kbps data rate. WiFlock takes this as an advantage and re-use these beacons to propagate group membership information as well.

To achieve high propagation speed of membership information, WiFlock efficiently synchronizes node listen times. As an illustration, Fig. 4.3(a) shows three neighboring nodes whose listen instances are not synchronized. Here node A can hear the beacons of nodes B and C. Node C can hear the beacons of B. However, node B does not hear the beacons of both A and C, while node C does not hear the beacons of A. Hence, with unsyn-

chronized listening, only unidirectional neighbor discovery is possible among neighboring nodes. In contrast, Fig. 4.3(b) show the three nodes with their listen instances synchronized. Assuming the listen duration is arbitrarily small and assuming that no two neighboring beacons attempt to beacon at the same time, with synchronized listening, each pair of nodes can hear each other's beacons (Section 4.3 describes how we deal with finite listen durations, as well as possible simultaneous beacon transmissions).

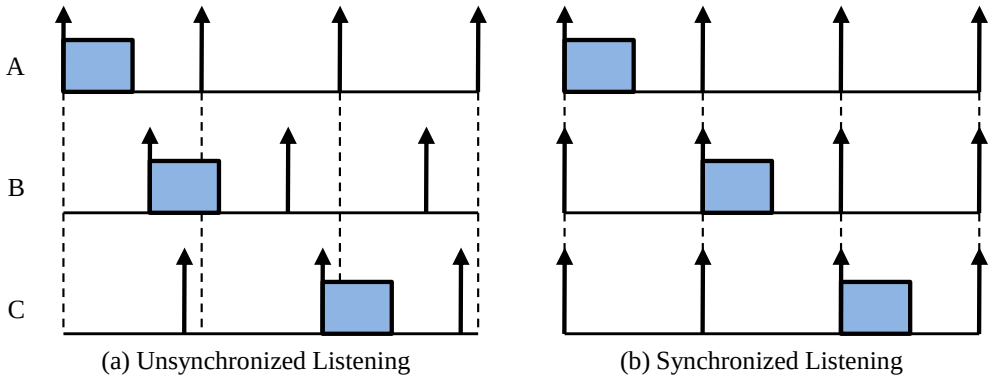


Figure 4.3: An example showing the difference between unsynchronized vs. synchronized listening among three nodes. The rectangles represent the periodic beacons while the impulses are the listening instances.

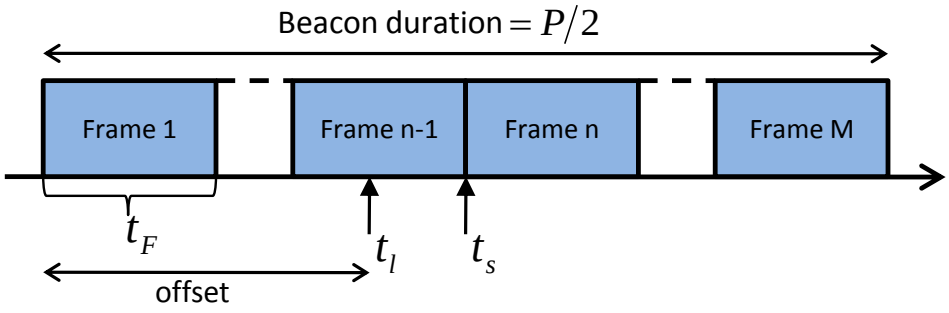


Figure 4.4: An example of computing the listen offset using multi-frame beacon.

To realize efficient listen time synchronization, WiFlock uses an approach similar to

XMAC [42], where each beacon is transmitted as a sequence of data frames. Each frame acts as a self-contained packet, so that if a node wakes up in the middle of a beacon, it can decode the next frame. Each frame within a beacon includes a *frame offset* n from the start of the beacon, thus, by measuring the time interval between its current listen time t_l, the start of the next frame t_s, and using the frame offset n, the receiving node can compute the offset between its current listen time and the start of the beacon by $(n-1)t_F - (t_s - t_l)$, where t_F is the frame duration (Figure 4.4). Since a beacon transmission starts at the listen time of a node, this offset is equal to the offset between the listen times of the two nodes.

Nodes use these offsets to achieve group-wide time synchronization and adjust their activities. Given time offsets between neighboring nodes, Werner-Allen `et al.` have shown that it is possible for an uncoordinated group of nodes to achieve time synchronization using only peer-to-peer interactions, rather than through a group-wide coordinated effort [43]. However, this way of time synchronization can take a long time to converge. For example, Werner-Allen et al. [43] report that the convergence time for a group of 24 nodes is 284 seconds.

To speed up convergence rate, we use a more coordinated approach. Specifically, a node will synchronize towards the smallest ID it hears. Of course, this has an issue due to the one-way discovery mechanism. Since only one of a pair of nodes can discover the other, a group can bifurcate to two synchronized groups: each occupies one half of listening period p and none can hear the other group. To overcome this problem, we take advantage of the data frames in the beacons and perform on-demand beacon extensions. Each data frame contains the group membership information, i.e. a list of sorted node IDs. While each node always synchronize its listening time towards the smallest ID it can hear, it also monitors the other IDs it receives that are not in the group. If any nodes with a higher

ID is discovered, the node will temporarily extends the beaconing to a full period p. The receiving node can then synchronize to this node, and through that, synchronize to the node with the smallest ID. The implementation details, including establishing on-demand bidirectional links, are described in Section 4.3

4.2.2 Evenly-Spaced Transmitting

The primary purpose of evenly spaced transmission is to collaboratively improve the group formation latency of a new node joining an already established group or a transient group. As an illustration, take an example of an established 2-node group, with each beacon having a beacon period of 100 time "slots" (each time slot refer to a listening duration). If the nodes are allowed to beacon randomly in the 100 slots, a new node wishing to join the node may have to wait for anywhere between 0 slots to 98 slots before it hears a beacon. However, if the two beacons are evenly spaced, i.e. each is 50-slots apart, an incoming node will now have to wait only between 0 and 50 slots to hear a beacon. Thus, a group can reduce discovery latency through collaboratively spacing out beacons.

Additionally, if a dense group with large number of nodes transmits beacons in an uncoordinated fashion, there can be packet collisions. Evenly spaced transmissions can minimize the number of collisions and hence allow the protocol to scale well to large number of nodes.

The mechanism for collaboratively spacing out beaconing times is as follows. The node with the smallest ID picks an arbitrary time slot as slot number zero. Each node maintains the group membership table, sorted by the node ID. Each node maintains a *current slot* counter, which is computed based on the information it has received so far. Using this current slot counter, the node transmits the slot number of the beaconing slot with its beacon

message. It also updates the *current slot* counter based on the slot counter information received with a beacon containing a node id that is equal or less than the smallest node id it has received so far.

Since there are $2q/p$ beacon slots in each beaconing period q, each node i uses its position within the sorted group membership table l_i and the size of the membership table s to compute its transmit slot $s_T(i)$ as follows:

$$s_T(i) = \frac{p \times l_i}{2s}.$$

The node transmits its beacon when its current slot counter reaches this transmit slot value. However, transmitting on a fixed slot can cause collisions, especially during the early stages of building the membership tables. This is because each node computes transmit slots assuming there are only a small number of nodes in the group.

To overcome such persistent collisions, we introduce jitters in the node transmission schedule. Instead of selecting the its computed slot number, the node selects a random slot number uniformly distributed within a small range of slot numbers centered around its computed transmit slot number. The range for random jitter is computed from the number of nodes in the group and the beaconing period. The slot number transmitted with the beacon is the slot number of the actual beacon slot, rather than the computed slot number.

As group membership tables are propagated using beacons, each node merges tables from its neighbors with its own table while keeping the table sorted. Under stable group membership, the membership tables at the nodes rapidly converge. The *current slot* counter value also converge to a consistent value across nodes, since this is always computed with respect to the smallest ID node in the group. Hence, under steady membership, the beacon transmission schedule converges such that node beacons are evenly spaced.

We note that, since a group can span over multiple communication hops, the transmission schedule computed by WiFlock is more relaxed compared to schedule computed by graph coloring using two-hop connectivity [44]. However, when the diameter of the group is small, the evenly spaced schedule would be close to a schedule computed by graph coloring.

We need to be careful again here with the short beacon size of $p/2$. Due to one-way node discovery, if all nodes beacon in the first half of their p length beacon slot, the entire group acts as a single phase node. Thus, a new node joining in either would be able to hear all nodes or would rely on its own beacon to make its presence known to the group. While, subsequent discovery still takes place with synchronization, the advantage of evenly spacing out beacons is reduced.

Thus, we allow nodes part of a synchronized group to randomly pick whether they transmit in the first half of their p length beacon slot or the second. Due to listen time synchronization and slight overlap between beacons, nodes within the group can still all hear each other. This mechanism was shown to improve the average latency of incoming nodes to discover the group in our experiments.

4.2.3 Handling Node Departure

To handle nodes leaving the group, each node entry in the group membership table contains a time-to-live (TTL) value. Each node initializes this value when advertising itself in the beacon message. The initial TTL value is set based on the freshness requirements (how quickly the node departure should be detected), and the maximum number of hops expected in a group.

The TTL value at each node is updated as follows. Whenever a node receives a mem-

bership table from a neighbor, it decrements the received TTL value of each node, and then compares these with the values stored locally. If the local neighbor table does not contain a received node ID, the new ID is added to the table with the updated TTL. If the node already exists, and the received TTL is greater than that in the local table, the local TTL is updated. After every beacon transmit, the node decrements all the local TTL values by 1. Any entry with a TTL equal to 0 is removed from the group table. For example, if we use a value of 7, then with a 3 hop network and a persistence time of the node is 3 time periods at the last hop.

Since the TTL value of a node is decremented at each hop, the initial TTL value determines how far a given node ID is propagated within the group. Since all the nodes need a uniform view of the minimum node ID, it is important that the initial TTL correctly reflects the maximum hop count within the group for correct operation of the synchronized listening and evenly spaced transmissions. On the other hand, a large initial TTL will require a longer time for the node information to be removed once the node leaves the group.

4.3 IMPLEMENTATION

In this section we present several implementation details of WiFlock, focusing on the hardware platform, the techniques to reduce wakeup and listening period, and the implementation of SLEST.

4.3.1 Hardware Platform

We use the TI EZ430-RF2500 development tool as the design and test hardware [45] (Figure 4.5). This module has a CC2500 (802.15.4 radio similar to that available on the IV generation SensorFly platform) radio and an MSP430F2274 microcontroller with 32kB

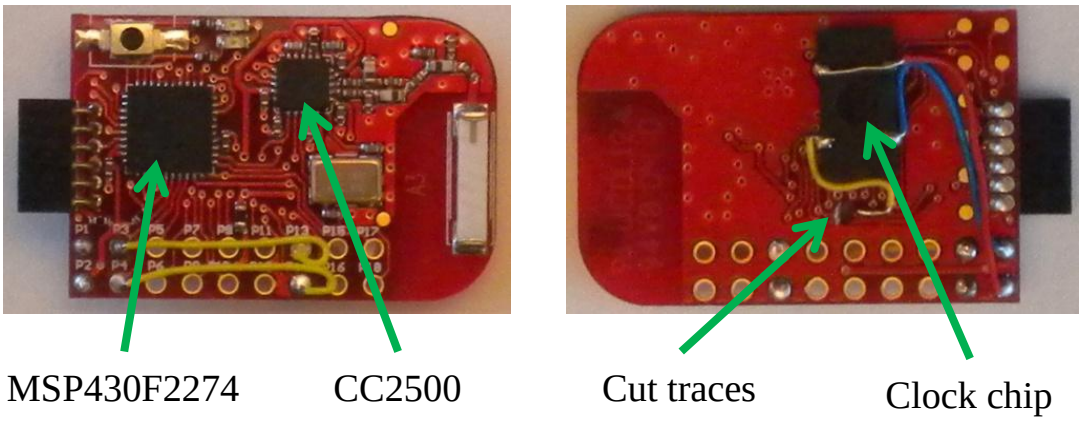


Figure 4.5: The modified EZ430-RF2500 development platform used for evaluating WiFlock. The figure shows the HW modifications for attaching the 32kHz clock chip.

Flash memory and 1kB RAM. It can be interfaced to a PC using a USB connector, which provides both a RS232 communication link and the firmware programming support.

Nodes running WiFlock use low power sleep to save energy. However, the TI evaluation module does not have a real time clock crystal that enables low power sleep with a relatively accurate wakeup timer. To overcome this, we added a 32kHz clock chip [46]. This clock chip consumes $\simeq 2\mu$A current, which is similar to the power overhead of the MSP430F2274 when driving a 32kHz clock crystal. The clock chip is introduced because it is difficult to modify the hardware module to properly attach a crystal.

4.3.2 Minimizing Discovery Latency

In Section 4.1, we observe that reducing the wakeup duration D is the most effective way to achieve a reasonable latency under a low duty cycle. The purpose of this wakeup duration is for a node to periodically wake up and detect any ongoing beacon transmissions by detecting the presence of a busy RF carrier. So our aim is to minimize D, without

Mode	Time	Frequency	Current
Idle-to-Rx	88.4μs	1	7.4mA
Carrier Sense	$\sim$80μs	1	18.8mA
Rx-to-Idle	0.1μs	1	7.4mA
Calibration	721μs	Variable	7.4mA

Table 4.1: Breakdown of radio current consumption on every wakeup.

adversely affecting the node's carrier sensing performance.

Radio listen time refers to the duration for which the radio is in receive mode for either carrier sensing or packet receptions, as opposed to the low power SLEEP mode. Table 4.1 gives the current consumption and time for operations that must be performed by the radio for every wake up. The VCO characteristics of the radio vary with temperature and supply voltage changes. Frequency synthesizer self-calibration can be performed manually when node operating conditions change. The current consumption for the radio Rx-mode, which includes carrier sense and radio packet reception states, is 18.8 mA for the CC2500 [41]. Thus, apart from beacon transmissions, the Rx-mode dominates the power budget of the radio.

Parameters Affecting Carrier Sense Times

Carrier sensing is done by measuring the Receive Signal Strength Indicator (RSSI) value, and comparing it against a threshold to determine the presence of the RF carrier. The duration to perform a valid carrier sense depends on the time required by the radio to provide a valid value for RSSI, i.e. the RSSI response time. The time to obtain a valid RSSI value depends on a number of configurable parameters of the radio. For example, in CC2500, the RSSI response time depends on receiver filter bandwidth, data rate, modulation format, and AGC(Analog Gain Control) module parameters [47]. Specifically, the

following factors impose the lower bound on the RSSI measurement time:

AGC Filter Length. The RSSI value is a byproduct of the AGC module, which uses the estimated RSSI to keep the signal level input at the demodulator constant, regardless of the signal level at the antenna. Since the RSSI signal is used to set the gain of an internal amplifier, it is generated by low-pass filtering the input signal to prevent frequent AGC gain switching. The length of this low pass filter affects the time required for the AGC to report a valid RSSI value.

AGC Wait Time. The AGC module has a configurable wait time after each gain adjustment to stabilize the control loop. A very short wait time may cause the AGC module to report unstable gain values, prolonging the final settling time. While a long wait time takes longer to generate a valid RSSI signal since the AGC ignores all samples during the wait time.

Signal Power. AGC settling time and hence RSSI response time depends on the power of the input signal. A stronger input signal can lead to a longer AGC settling time than a weaker input signal, since a strong signal may require multiple gain changes. While the theoretical minimum response time can be computed from the equations in [47], the stochastic nature of received power makes empirical evaluations necessary.

In general, the larger filter bandwidth, the higher the data rate, and lower the AGC filter length, the lower is the theoretical minimum RSSI response time. Figure 4.6 shows the measured RSSI response time as a function of the AGC filter length and filter bandwidth.

Empirical Evaluation

We empirically evaluate the effect of RSSI response time setting, obtained by varying AGC parameters, on the accuracy of carrier sensing. Carrier sensing is performed through

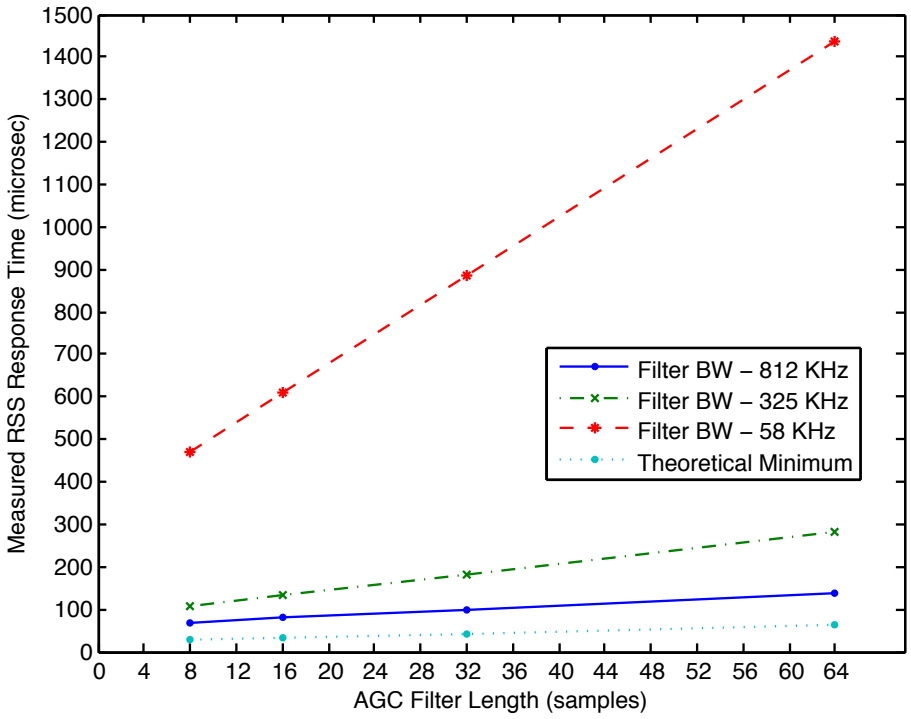


Figure 4.6: Measured RSSI response time as a function of filter length and filter bandwidth. A higher filter bandwidth reduces the RSSI response time. A lower filter length corresponds to lower RSSI response times.

comparing the detected energy (RSSI value) to a fixed threshold. We establish this threshold as the 95th percentile RSSI value observed by a node in a quite environment with no valid transmissions. Carrier sensing accuracy is measured in terms of percentage error, defined as the number of valid transmissions missed by a receiver node to the total number of transmissions by the sender.

Figures 4.8 & 4.7, show the percentage carrier sensing error for nodes with different RSSI response time settings. The experimental setup included a pair of nodes, configured as a receiver and a sender. The sender was set to continuously transmit data at maximum power of 0dBm. The receiver was programmed to wakeup every 0.5 seconds and stay awake until a valid RSSI reading was obtained, i.e. the radio was on for the RSSI response

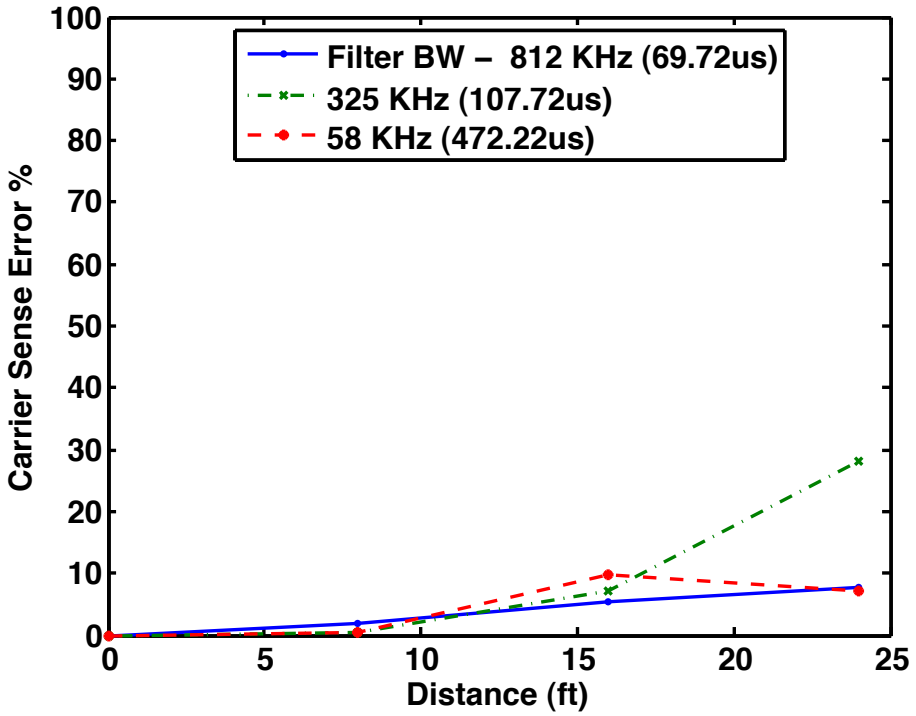


Figure 4.7: Error rate in detecting a transmitting carrier at different filter bandwidths as a function of distance, with a filter length of 8 samples with CS threshold of 95%tile. Filter bandwidth does not show a clear effect on carrier sensing error rates.

time. The receiver node logged the RSSI value to the PC through a USB port. The distance between the sender and receiver was varied in order to change the signal strength at the receiver node. The experiment was repeated with 12 different combinations of AGC filter length and wait time. 1000 readings were obtained for each setting and distance. The carrier sense threshold was established by obtaining the 95%tile value of 1000 RSSI readings obtained at the particular settings, with the sender node turned off.

Every RSSI value below the threshold was considered as correctly detected carrier, while the number of missed detection contributed to the percentage carrier sensing error. We observed that selecting a wider bandwidth did not have a pronounced effect on carrier sense errors, all other factors being constant. However, a smaller filter length (8 samples)

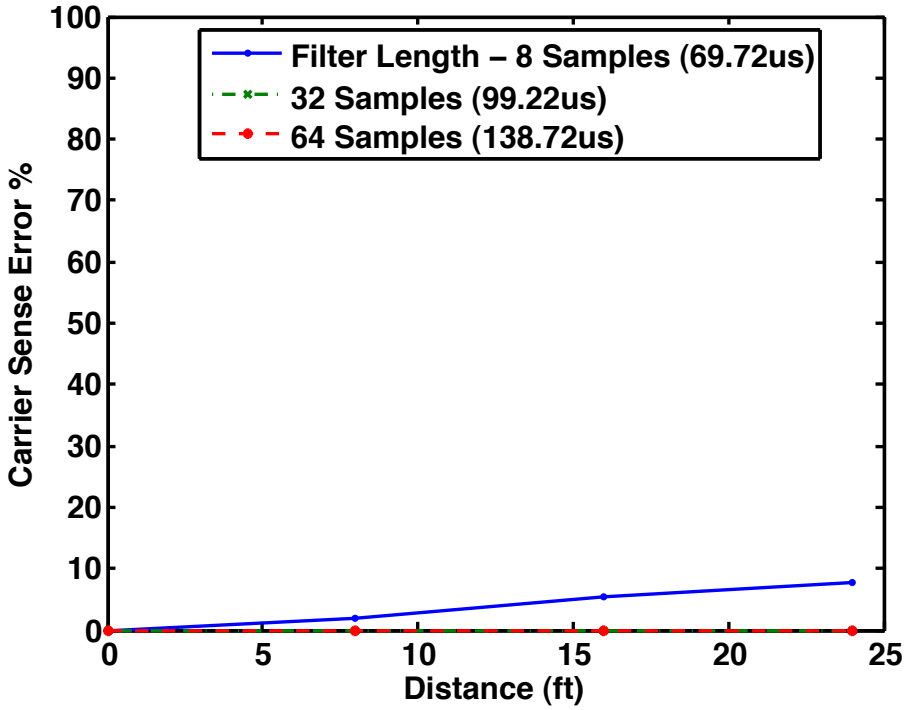


Figure 4.8: Error rate in detecting a transmitting carrier at different filter lengths as a function of distance, with filter bandwidth 812kHz and CS threshold 95%tile. 8 sample filter length provides very short carrier sense times of ¡ 70μs, but can cause more errors as signal strength drops.

caused about 10% missed detections when the sender was moved 25 feet away resulting in a weak signal at the receiver. Thus, there is a trade-off between energy consumption cost due to a higher CS time vs. the latency cost of not reliably detecting some valid transmissions.

Our experiments show that with a AGC filter bandwidth of 812kHz and filter length of 8 samples, a carrier sense time of 70μs is achievable. This allows us to use small listen slots of 80μs duration, with some guard against interference sources. This is a 3X improvement over prior neighbor discovery protocols [39], providing the corresponding improvement in latency for similar duty cycles. This allows us to achieve reasonable discovery latencies even with very low duty cycles such as 0.2%. We note that these parameter settings should

only be used for carrier sensing. They may not be optimal for data communication. WiFlock nodes update these parameters with a different set of values when transmitting or receiving data.

Node Listen-Wakeup Scheme

Energy detection based carrier sensing suffers from the possibility of *false wake ups*, due to interference from other sources such as WiFi, which may occupy overlapping channels. Depending on the nature of interference, the radio may detect energy and stay in receive mode for a long time in the absence of valid transmissions. Thus, we employ a tiered scheme for countering this effect and minimizing node listen durations.

First, each node uses the short duration energy detection (80μs) to detect the presence of an ongoing transmission. Second, if an ongoing transmission is detected, the node keeps the radio in the receive mode for the duration of a single frame, while continuously searching for the sync word; if the carrier becomes free during this search, the radio and the node goes back to sleep. Third, if a sync word is detected the radio remains in the receive mode until an entire packet is received or until the carrier becomes free.

In Section 4.4, we evaluate the robustness of this tiered scheme against overlapping WiFi interference, and obtain a typical carrier sensing duration of 80μs under WiFi interference.

4.3.3 Multi-Frame Beacons

At very low duty cycle (for example,$\sim$0.2%) operation, which is desired by the WiFlock protocol, the advertising beacon needs a very long preamble of length ($P/2$ or $40ms$). A listening node using a busy carrier preamble such as in U-Connect [39], on average, would

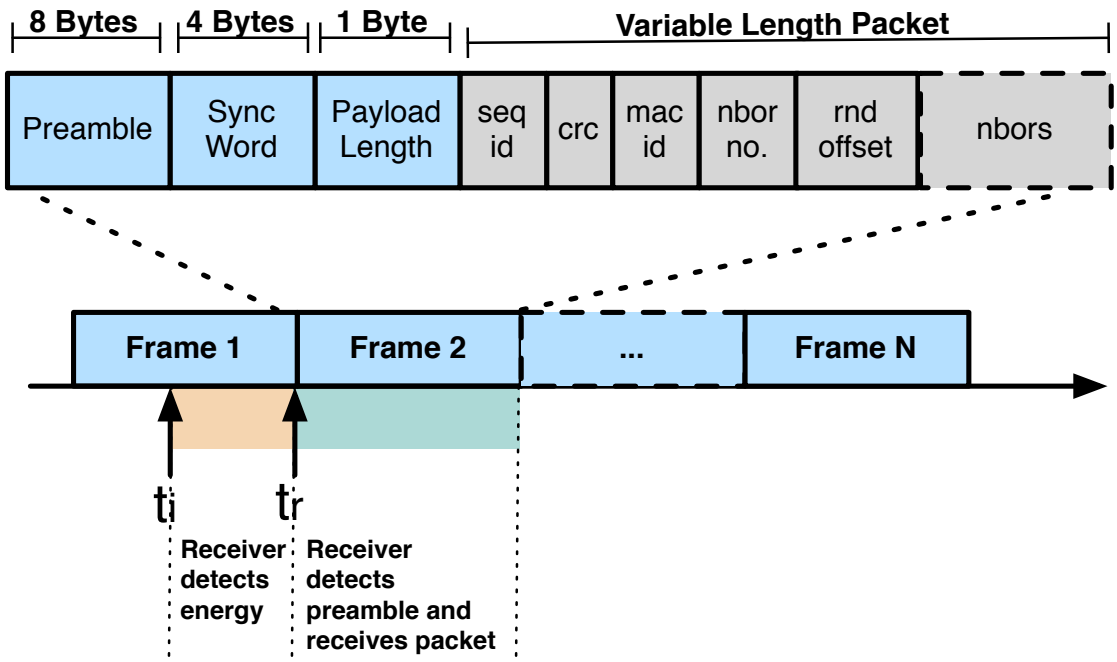


Figure 4.9: Format of the WiFlock beacon with multiple contiguous frames with their contained fields. A receiver waking up at time t_i tracks the beacon frame until the next frame boundary at time t_r, after which it can detect the preamble and eventually receive the packet.

track (radio remains in receive mode) the preamble for $20ms$ for every packet reception. This is equivalent to receiving 20 128-byte packets at a baud rate of 256Kbps. With special preamble architecture, consisting of back-to-back frames with group advertisement packets, we reduce radio on times as well as provide a mechanism for implicit listen phase synchronization.

The problem of long preambles has been mentioned and solutions proposed in MAC schemes like X-MAC [42]. X-MAC takes the approach of sending a strobed preamble with encoded destination addresses. The strobed preamble allows nodes to minimize the preamble tracking time while the encoded destination addresses allow nodes to avoid receiving unintended packets.

In contrast, WiFlock beacon architecture addresses different objectives and has following features –

- It has a predictable data transmission requirement, where nodes advertise group information, that are contained in relatively short frames, periodically. It operates at a very low duty cycle, which allows sending multiple data packets (frames) instead of just packet header information. This reduces the maximum preamble tracking time of listening nodes to the length of a single frame.

- The WiFlock beacon consists of multiple frames with no gap between them. This enables the nodes to use a short listening period to detect an on going transmission, since there is no danger of the listening interval falling in to a carrier free gap.

The beacon is implemented as a very long continuous transmission that includes multiple frames, where each frame is a qualified CC2500 packet preceded by a preamble (hence, a receiver waking up in the middle of this transmission can receive the next frame). The beacon is 10% longer than the required $p/2$ beacon length. The additional length ensures that a node that wakes up at the very end of the $p/2$ period can still receive a valid frame. The additional length also helps mitigate effects due to clock drift.

The long gap-less beacon is realized through the radio's [41] infinite length packet mode, which allows arbitrary length packets. The infinite length packet mode is a feature of a range of newer Texas Instrument's RF transceiver chips including CC1100, CC1100E, CC1101, CC1150, CC2500, and CC2550. Infinite packet length mode can be used to transmit and/or receive until TX or RX mode is turned off manually.

The nodes employ a random back-off mechanism before beginning transmission if another transmission is detected. In addition, a random jitter is introduced in beacon trans-

missions to account for exposed terminal and hidden node issues. The back off and jitter moves the beacon transmission by multiples of p long beacon slots.

Figure 4.9, shows the structure of a WiFlock beacon. Each beacon consists of multiple frames. Each frame, similar to a single CC2500 packet transmission, has a preamble, a sync word and payload length field, followed by the variable length packet, with a frame sequence id and group information. Since each frame acts as a fully qualified CC2500 packet, a receiver node radio can detect and receive the packet embedded in the frame as per its normal link layer protocol. The sequence id field in each packet, embedded in each frame, specifies the corresponding frame's position in the beacon. The receiver uses this sequence id for listen synchronization. The length of the fields in each frame can be configured by the user as per application requirements. The experimental setup used for this chapter was configured to use a 1-byte sequence id, 2-byte CRC, 1-byte random offset, 1-byte neighbor number, and 1-byte neighbor mac-id fields.

For large groups where the group information does not fit in one frame length (256 Bytes), the nodes encode frames with portions of the group information in a cyclical fashion. For example, the first 243 bytes of a 486-byte group table is encoded in frame-1 of the beacon, the second half is encoded in frame-2, frame-3 encodes the first half again, and so on. A receiver after receiving a particular frame figures out that it has received a part of the group table and must stay on to receive subsequent frames. The receiver may need multiple beacons to obtain the entire group table from a node.

4.3.4 Propagating Group Information

WiFlock uses synchronized listening to enable fast group information propagation. Listing 4.1 shows the pseudo code the nodes use for propagating the neighbor table and

Listing 4.1: Pseudocode for phase sync algorithm

```
// Synchronize if hear a sending node that is sync'd to a lower id node
if (urSyncNeighbor < mySyncNeighbor) {
        syncListen(rxFrameSequenceId);
} else {
        // Extend beacon if the sender must sync
        ExtendBeaconFlag = TRUE;
}

// Compute offset and advance next wakeup
void syncListen(rxFrameSequenceId) {
        del = rxFrameSequenceId * BEACON_FRAME_TIME;
        if ( del >= BEACON_OFFSET_TIME && del <= BEACON_TIME )
                del = del - BEACON_OFFSET_TIME;
        else
                del = 0;
        if (del == 0)
                return;
        advanceNextWakeup(del);
}
```

achieving synchronized listening. The synchronized listening attempts to synchronize to the smallest id node. However, the unidirectional discovery may prevent the smallest beacon information to propagate from node to node. The top half of the code shows the logic used to get around this. If a node i receives a beacon form a node j with a smallest id (the first node of the frame) that is large than its own smallest id, the node i sets its `ExtendBeaconFlag`. The next time the node i beacons, it uses a beacon of duration p, which lets node j hear that beacon, hence receive the current smallest node information at i. The bottom half of the code shows how nodes use listen time offset to synchronize to the smallest ID node.

We used a 6 node test bed to evaluate the synchronization performance under 3 network topologies - all-to-all connected topology, a grid topology where a node could communicate with 2 or 3 other nodes depending on its position, and a chain topology. Each node was programmed to set a GPIO pin to high when its radio was in receiving mode. Using a logic analyzer connected to all 6 nodes, we observed the timing of node listen intervals. We determine the degree of synchronization within the group through *time spread*, which is

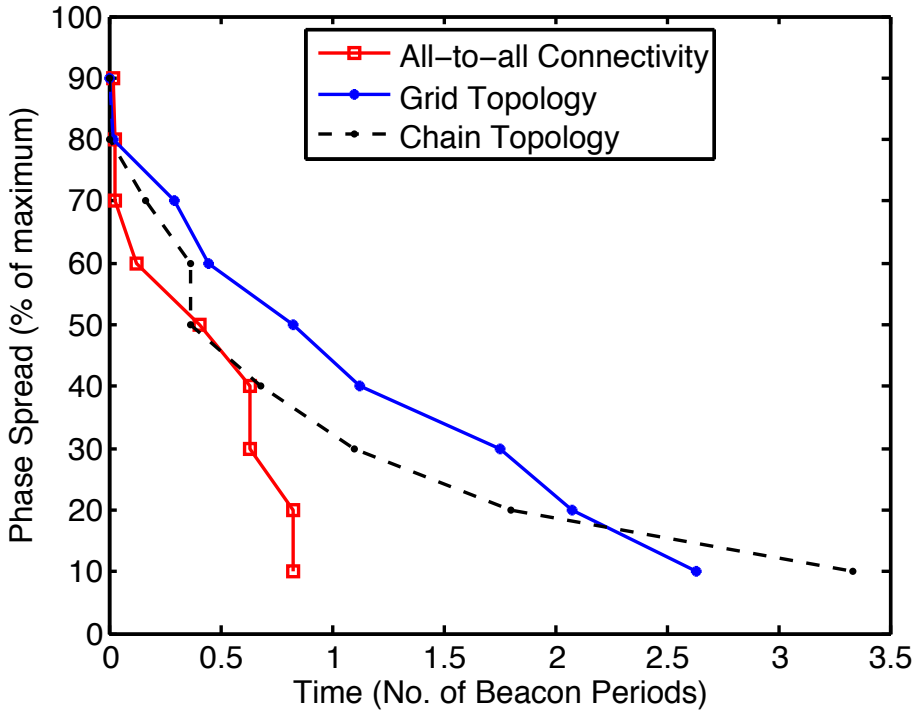


Figure 4.10: Percentage time spread achieved vs. beacon periods for group of 6 nodes under three network topologies - all-to-all connectivity, grid and chain.

defined as the ratio of the maximum phase difference between any two nodes in the group, to the listen period. As WiFlock has a 10% overlap in transmit beacons, we assume that nodes are synchronized when the time spread is ¡10%. From Figure 4.10, synchronization was achieved within 1 beacon period for the all-connected case, while around 3 beacon periods were required for the chain topology.

4.3.5 Sorted Group Tables

WiFlock uses the ranking based on node mac-id's to evenly spread the node beaconing within the node beacon period. To minimize the processing overhead, nodes maintain a sorted group table locally as well as in the beacons. Starting from the node's own ID, when-

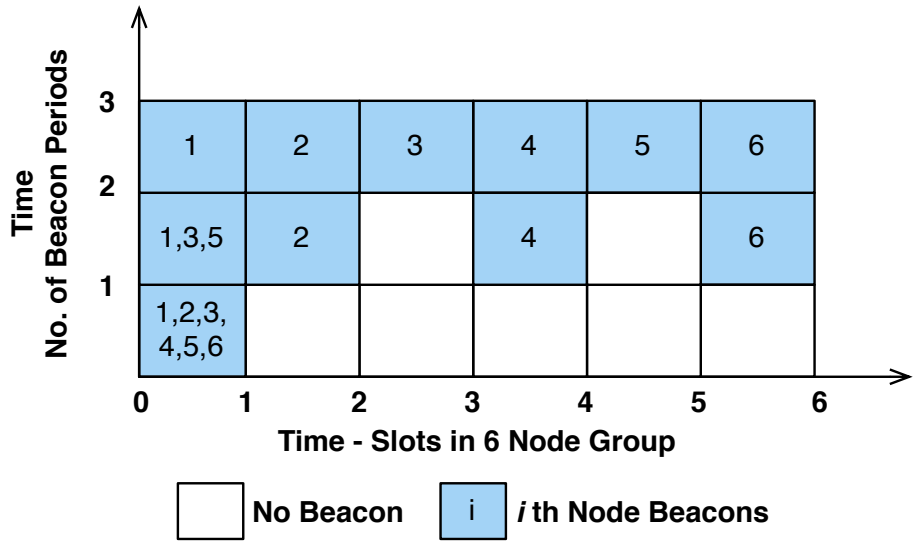


Figure 4.11: Example of nodes assuming evenly spaced beacon positions from a 6 node experiment with all-to-all topology.

ever a node obtains a group table from an advertised beacon, it performs a computationally inexpensive ($O(n)$) merge operation to keep the table sorted. The number of comparisons performed is nearly optimal in this algorithm, thereby minimizing processing overhead on receiving every group table update.

Figure 4.11 visualizes how the sorted tables progress among a group of 6 nodes. The nodes were instrumented with a logic analyzer to obtain the timing. Here, the beacon period is divided to 6 equal sized slots for easy visualization (each slot shown contains multiple beacon slots). Initially, with no information about the other nodes, all the nodes transmit in the first slot. The random jitter introduced during the beacon transmissions prevent persistent collisions and allows the group information to propagate. Nodes 5 and 3 beacon before node 1 in the second period, and therefore do not shift to their respective slots. Nodes 2 hears three other nodes beaconing ,including the lowest-id node 1. As node 4 has not beaconed yet, node 2 computes its position relative to a group membership of five

nodes only. Eventually, node 4 beacons in its correct position since it has heard beacons from all other nodes in the group. By the third beacon period, the group table is consistent among all nodes and the beacons are evenly spread out.

4.4 PERFORMANCE EVALUATION

In this section, we empirically evaluate the performance our WiFlock implementation. Through a series of experiments, we study the energy consumption, group formation latencies, and scalability of our solution in realistic deployment conditions.

4.4.1 Radio Listen Time

Radio listen time duration D is a critical parameter in WiFlock, which not only determines the energy spent on the listening model, but also affects the beacon length. While in Section 4.3.2 we show that the carrier sensing time can be made as low as 70μs, we must evaluate the choice in realistic environments, especially with WiFi interference. WiFi transmissions can cause the radio to detect energy and stay awake unnecessarily.

To assess the effect we setup a 802.11 network consisting of a802.11 b/g access point and a laptop equipped with an 802.11 wireless radio in infrastructure mode. This laptop was used to generate a constant stream of 1,500-byte UDP packets using the *iperf* tool. The *iperf* tool was used to vary the bandwidth utilized by the 802.11 transmissions. Another laptop, connected to the access point through an Ethernet cable, acted as the traffic sink for the WiFi network.

A WiFlock node was placed adjacent to the transmitting laptop. The node radio was set to the same frequency band as that of the 802.11 channel to force maximum interference. The node was instrumented with a logic analyzer to obtain accurate timing when the radio

changed modes. The listen time was averaged over 1000 wake up's of the radio. Each experiment was performed 10 times at different times over the course of a day.

Figure 4.12 shows the average time the radio spent in receive mode for idle listening, i.e. with no valid WiFlock beacons, as 802.11 traffic is increased for 0 to 10 Mbytes/second. We observe that the radio idle listening time increases with increasing 802.11 traffic.

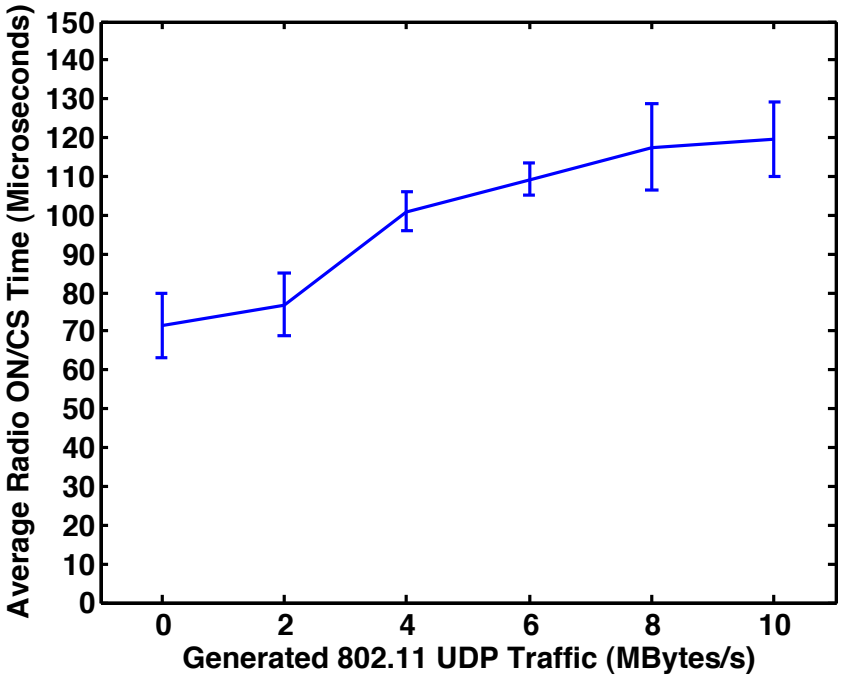


Figure 4.12: The effect of interfering WiFi traffic on energy-detection based carrier sensing scheme. The average radio idle listen times increase as generated 802.11 UDP traffic is increased.

However, Figure 4.13 shows the average radio idle listening time in typical environments where 802.11 traffic is not generally constant. We perform the experiment at 3 different location and in 2 channels with and without overlapping WiFi channels.

The channel without Wi-Fi overlap had nearly constant average ON times of 70μs, the ON time increased to 84μs at the location with maximum activity. Based on these observations, we select 80μs duration as the *expected* listen slot size to accommodate possible WiFi

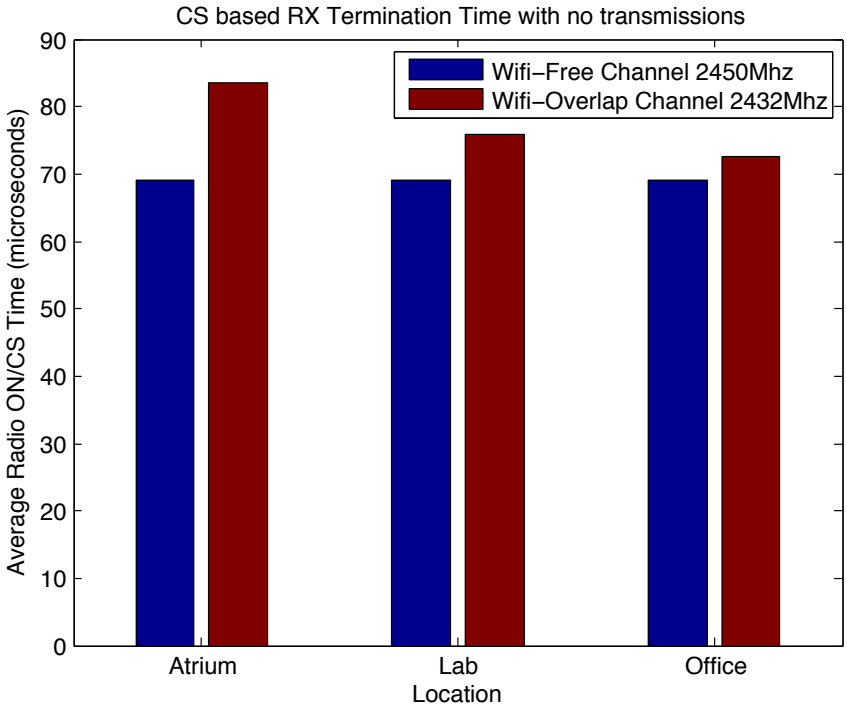


Figure 4.13: The energy detection based carrier sensing performance in common interfering environments. The presence of WiFi in real environments increases the average idle listen time by 10-20%.

interference. The 80μs is $\simeq$3X smaller than U-Connect, the previous state-of-art neighbor discovery protocol[39].

Figure 4.14 shows the current consumption during every node wakeup. The MSP430 micro-controller consumes $\sim$2.8 mA when operating at 8 MHz. The CC2500 radio consumes 1.5 mA when in *idle* mode. The radio transition from *idle* mode to *Rx* mode takes about 100 μs drawing $\sim$8 mA of current. Carrier sensing takes $\sim$80 μs drawing 19 mA of current.

We also measure the average radio listen time for a packet reception to evaluate the benefit of including data frames in the advertisement beacons. The time is averaged over 1000 periodically received packets by a listening node from a corresponding number of

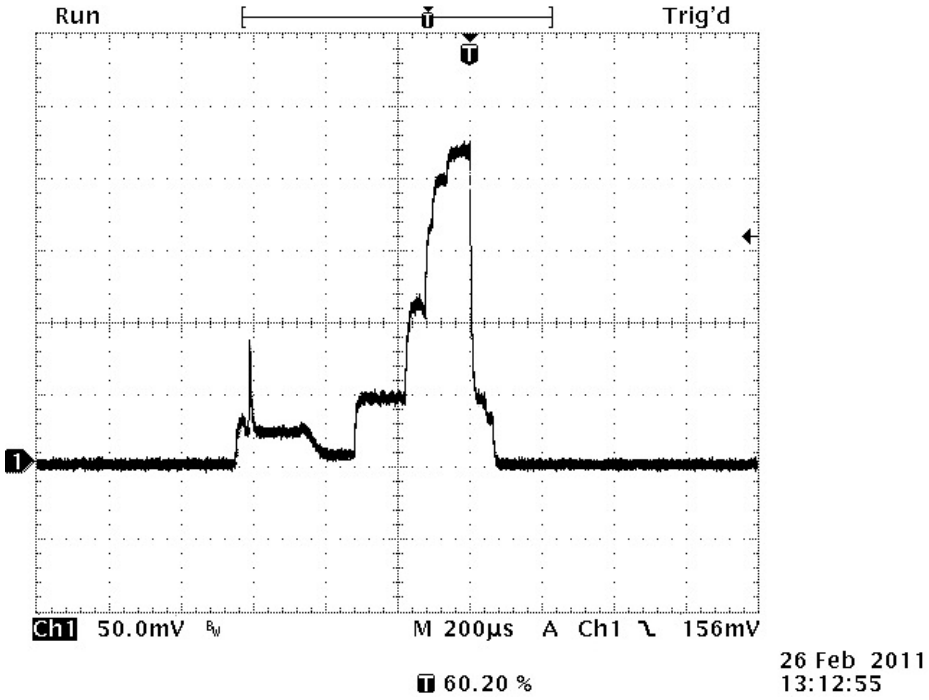


Figure 4.14: Oscilloscope trace showing current consumption (in mV using a 10.3 Ω resistor in series with power supply) during every node wakeup for listening.

transmitted beacons by a transmitter. Each advertisement beacon contains 20 packets, and each packet is 128 bytes long. The packets are transmitted at a baud rate of 250Kbps. Compared against the average theoretical radio listen time for LPL-like beacons employed by U-Connect, the WiFlock beacon structure reduces the time for which the radio must track the beacon before successfully receiving data contents. Average beacon decoding time is about 7.4ms, a 5X improvement over LPL.

4.4.2 Group Discovery Latencies

To evaluate group discovery and formation latency, we setup 50 nodes with listening duty cycle of 0.2% and listen slot durations of 80μs. All the nodes were located within a single hop from each other. Of these 50 nodes, 20 nodes were connected to a PC via USB.

Group Size	Avg. Latency	Std. Dev.
5	113s	35s
10	114s	30s
15	119s	12s
20	120s	22s
50	168s	16s

Table 4.2: Group formation latencies for groups of varying sizes, with nodes operating at 0.2% duty cycle.

The PC can turn on and off a group of selected size from the twenty nodes (the remaining 30 nodes were manually switched for the 50 node experiment). Each active node logs updates to its group table to the PC, where they are time stamped. Node groups of different sizes are turned on with random offsets over a 40 second period. The one beacon period interval for nodes joining is to ensure nodes are not synchronized in the beginning. The experiment provides a snapshot of every node's group table at any given time. We repeat the experiment 20 times for each group size.

Table 4.2 shows average and standard deviation for group formation latencies, i.e. the time for the group information to be propagated to every node in the group, with various group sizes. Since, every member of the group must discover every other member of the group, this corresponds to the worst latency observed by any node during an instance of the experiment.

Figure 4.15 shows the average rate of discovery, i.e. time nodes to discover a fraction of the entire group.

From Figure 4.15 we can see that the average time it takes for a set of nodes to discover and form a group stays rather constant when the group size increases. For less than 20 nodes, at 0.2% duty cycle nodes consistently discovered the entire group within an average of 120 seconds. Even with 50 nodes, the average discovery latency is less than 3 minutes.

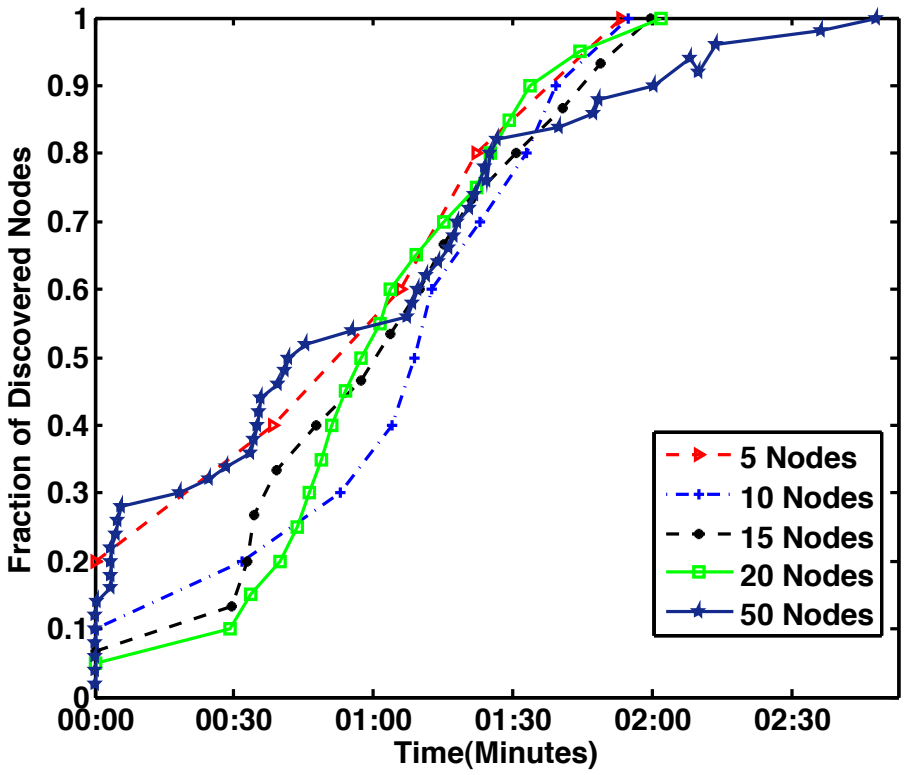


Figure 4.15: The average time for a number of nodes, operating at 0.2% duty cycle, to discover a fraction of the group. The evenly spaced beaconing reduces packet collisions and scales well with increased group sizes.

The results show that the synchronized listening and evenly spaced beaconing allows the protocol to scale to a large number of nodes without affecting latencies.

Neighbor discovery protocols such as U-Connect and Disco provide theoretical worst-case bounds on discovery latencies for a pair of nodes. While they support discovery in clusters of nodes, they do not optimize for or address problems of group maintenance such as absence of all-to-all connectivity or the high probability of collisions as group sizes become large. In contrast, WiFlock goes beyond neighbor discovery to provide a service for discovery and maintenance of groups of mobile wireless nodes. It uses synchronization and sharing of neighbor tables to make continuous neighbor discovery efficient and scalable

to large clusters of nodes.

Nevertheless, for pair-wise discovery, WiFlock provides a 3X improvement in theoretical worst-case latency over U-Connect due to shorter carrier sense times. For example, at 0.2% duty cycle the theoretical worst-case latency for a pair of U-Connect nodes to discover each other is 250 seconds, while that for WiFlock nodes is 80 seconds. U-connect experimental results suggest the latencies may exceed the theoretical maximum when the number of nodes in the same collision domain is high. In practice, in a large group of 50 nodes with collisions and varying network topologies, WiFlock achieves performance well within the worst-case for U-Connect.

4.4.3 Group Maintenance Latencies

An already established group can collaborate on their beaconing to improve the efficiency for the group to react to changes such as discovering new nodes.

A new coming node, depending on its distance from the group and location, may have connectivity with only a fraction of the established group. In static networks, the network topology may be known and can be used by the network to predict the connectivity of an incoming node[48]. However, no such link quality assumptions can be made in flocking sensor networks where all nodes are mobile. Therefore, we empirically evaluate the average case and worst case latencies for a node approaching a group when only a fraction of the group nodes are in communication range of the new node.

Figure 4.16, shows the average latency for an incoming node to discover an established 20 node group, when it has connectivity with only a fraction of group nodes. The experimental setup is similar to the one used in section 4.4.2. A group of 20 nodes is given enough time to to form an established group. A new node is then turned on and the radio

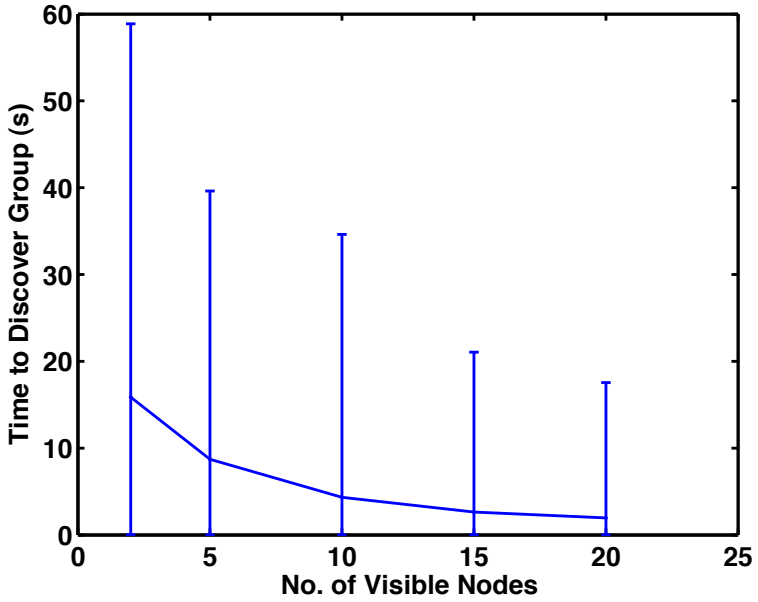


Figure 4.16: Average discovery latencies when a mobile node joins an established group of 20 nodes, when only a fraction of the group has connectivity to the incoming node. The error bars show the maximum and minimum latency observed.

connectivity is simulated by a ID filter. That is, the exact node id's that the new node is "supposed" to hear is provided to the node by the PC before each experiment. For example, to simulate a situation that the new coming node can hear n nodes in the group, a random n id's is picked and sent to the new coming node. The time for the new node to discover the group is logged. We repeat 1000 runs for each n. Obviously, greater exposure to the group shortens the latency of new discovery.

4.5 RELATED WORK

Continuous neighborhood management for devices not subject to energy constraints can be attained simply by periodic transmission of advertisement beacons. Neighboring devices with always-on receivers can periodically update their neighborhood tables to the

presence of co-located devices.

If an out-of-band time synchronization mechanism exists such as GPS[49], FM transmitters [44], neighbor discovery can be achieved with higher efficiency. However, for applications such as in asset tracking, where *tags* are constrained due to both cost and energy, such out-of-band mechanism lose their appeal.

For energy constrained but static networks neighborhood management can be attained as two-phase process, where larger latency and energy consumption may be tolerated for initial neighbor discovery while switching to a more energy efficient common mode of maintaining neighborhood information. In some cases, for mobile nodes the initial neighbor discovery phase can be periodically repeated. Its relatively higher latency and energy consumption may be acceptable if node lifetime requirements are relatively less strict. As an illustration, neighborhood maintenance in mobile ad hoc networks is generally a function of the routing protocol such as AODV [50]. Nodes usually operate in the always listening mode. When a node needs to communicate with another node, a RREQ (route request) packet is flooded to other nodes. This causes high control traffic overhead and delay when initiating communication for the first time. An established route maintenance procedure is subsequently adopted using the RERR (route error) message to notify other nodes of the loss of neighbors. In mobile networks with highly transient neighborhoods such protocols have a high energy cost, having to constantly initiate route requests.

Quorum-based protocols [51] or variants there-of such as BMAC [40], SMAC [52] provide useful primitives for communication between a pair of devices. In Quorum, a sequence of beacon intervals is divided into sets starting from the first interval such the each continuous n^2 is called a group, where n is a global parameter. In each group, intervals are arranged in a $n \times n$ array in a row-major manner. Nodes arbitrarily pick a row and

a column, and any pair of nodes will have two intersections allowing them to discover each other. While these protocols address the basic primitive behind discovery they do not concern themselves with the very low-energy operation as desired by our motivating application. Also, for a given duty cycle their implementations tend to provide order of magnitude higher latencies than our proposed solution. As they are essentially based on nodes beaconing at random times, they do not address network effects like collisions which intensify in large groups. This is also the case for ”birthday protocols” for static ad-hoc networks, proposed in [53], which also have long tails on discovery probabilities.

Dutta et. al. propose Disco, a low-power asynchronous neighbor discovery protocol [38]. The authors address the rendezvous of two mobile nodes with support for nodes operating at dissimilar duty cycles. Nodes select a pair of primes such that the sum of their reciprocals is equal to their desired duty cycle. Nodes wakeup at multiples of the prime numbers. The worst case latency is the product of the minimum of the primes picked by the nodes in operation. Disco also analyzes discovery latencies in clusters and suggests extending the work by using gossip. We adopt some of these ideas in our protocol. However, Disco requires large wake-up slots in the order of milliseconds due to the need for bidirectional communication and susceptibility to clock drift. This results in large latencies for our desired duty cycle of operation.

Kandhalu et al. propose U-Connect, a protocol for asynchronous neighbor discovery for both symmetric as well as asymmetric duty cycle nodes [39]. They establish that U-Connect is an 1.5-approximation algorithm for the symmetric asynchronous neighbor discovery problem, whereas existing protocols like Quorum and Disco are 2-approximation algorithms. We use a modification of U-Connect as the basic neighbor discovery primitive. The flip side to the U-Connect protocol is the creation of asymmetric links where only

one node can hear the other depending on the phase of its beacon. To discover each other, nodes must perform a pair-wise id exchange every time a node hears another node, creating message traffic in $O(n^2)$. This must be frequently repeated to maintain a continuously up-to-date neighborhood group. Disco also uses a pair-wise message exchange for discovery making it less attractive for scalable, low-latency and energy efficient neighborhood management.

Continuous neighbor discovery is mentioned in [48], in the context of maintaining node connectivity information in face of disruptions in wireless communication, synchronization or change in transmission power for established static networks. The authors propose an algorithm for continuous neighbor discovery after an initial neighborhood has been established using a broadcast SYNC message which is heard by all nodes. This assumption of all nodes being either synchronized or awake at the same time when initially deployed, is not applicable to a network of mobile nodes with transient neighborhoods.

4.6 CONCLUSION

This chapter presents WiFlock, a collaborative group formation, and group maintenance protocol for controlled-mobile sensor networks. In this chapter, we show that the neighbor discovery process and the group maintenance process must be designed together to achieve better energy-efficiency, reactiveness, and scalability. Since a sensor node can spend most of its time alone, optimizing beaconing and listening periods and durations is critical. Through analysis and implementation, we show how to achieve ultra low duty cycles ($\sim$0.2%) without sacrificing discovery latency.

CHAPTER 5

COVERAGE

In this chapter, we present SugarMap – a system that enables resource-constrained controlled-mobile sensor swarms to collaboratively cover an area. In a majority of the proposed applications for controlled-mobile sensing systems, such as surveillance or survivor search, the sensor network is tasked with moving and covering a target space (single-room). These networks must rely on collaboration to quickly and efficiently achieve their system-wide sensing objectives despite the limitations of individual nodes.

SugarMap coordinates node movements to reduce the amount of sensing overlap between controlled-mobile sensor nodes and increase the efficiency and speed of coverage. Most importantly, the system does not rely on external location infrastructure, bulky or sophisticated sensors, computation intensive algorithms or high-bandwidth communication.

In the SugarMap system, radio RF-signatures are obtained by *explorer* sensor nodes through querying a set of *anchor* nodes. The anchor nodes are controlled-mobile nodes that SugarMap deploys (lands) in the area, through a dispersion algorithm, at initialization. The system does not require a location information for deploying anchor nodes and is thus self-establishing. Consequently, SugarMap uses the radio measurements from the anchors as a

common spatial frame of reference to coordinate node motion and collaboratively cover an unknown area. The algorithm uses particle filters to account for the actuation uncertainty of low-cost controlled-mobile nodes and uses redundancy in node paths to guarantee coverage with the desired degree of confidence.

We evaluate the performance of SugarMap through large-scale simulation and validate through a real-system implementation on the SensorFly [54] controlled-mobile sensor platform. We compare the performance of SugarMap to the state-of-the-art in existing online coverage algorithms for swarms. We show that SugarMap provides faster coverage than existing approaches in absence of location information.

5.1 SYSTEM OVERVIEW

This section gives an overview of the different aspects of the SugarMap system, including the capabilities of nodes and the typical operating scenario.

Figure 5.1 gives a flow diagram of data and commands between the major components of the SugarMap system. At system initialization, nodes are sent into the environment and landed in a constrained dispersion manner. The SugarMap system has a base station *brain* running the SugarMap algorithm. The base station gives commands to the mobile nodes (explorers) to move in a coordinated fashion. The explorers collect RF-signatures from stationary nodes (anchors) and relay it to the base station. The base station uses particle filters and RF-signatures obtained from the explorer nodes to update a global coverage map for the swarm. The global coverage map in turn helps the base station recursively plan the next movement of explorer nodes.

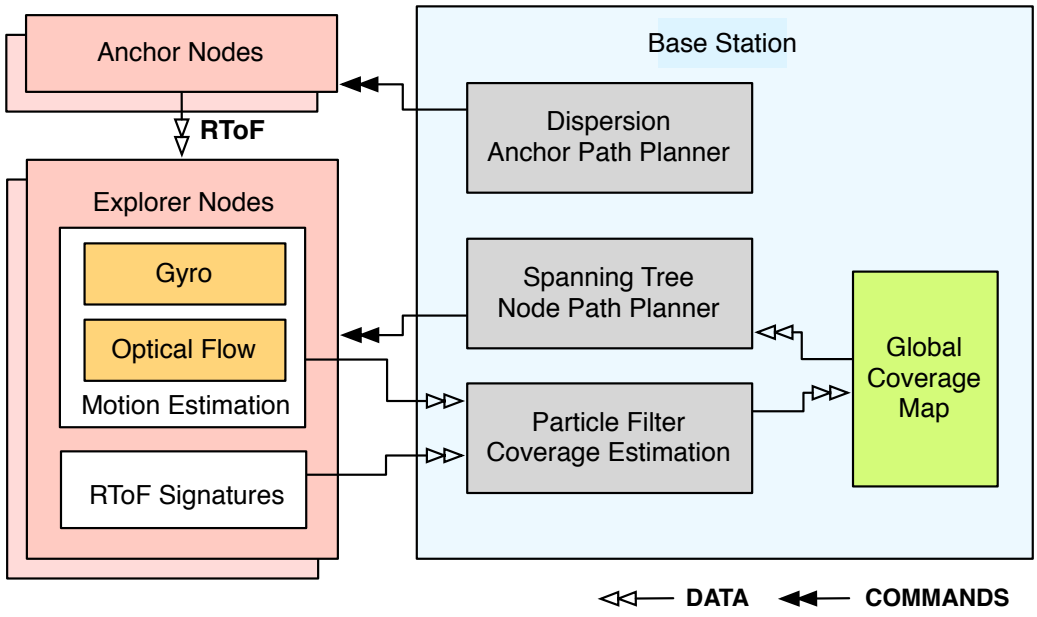


Figure 5.1: An overview of the SugarMap system.

5.1.1 Controlled-mobile Sensor Nodes

As per our design, we contend controlled-mobile nodes to be weight-limited resource-constrained platforms. They have limited on-board computation capability and light-weight components such as MEMS-based inertial motion sensors, an altitude measurement sensor, a sensor to estimate velocity, and a radio for communication and RF-signature estimation.

For our prototype evaluation, we implement SugarMap on the SensorFly [54] controlled-mobile swarm platform. The SensorFly is a controlled-mobile platform with a 8-bit AVR AtMega128 microcontroller, inertial motion sensors – 3-axis accelerometer and 3-axis gyro, an ultrasonic ranger for altitude estimation, an optical flow sensor for velocity estimation, and a 802.15.4a compatible radio with Round-trip time-of-flight (RToF) measurement capability. The entire platform is under $30g$ in weight striking a careful balance between weight and sensing capability as is typical of controlled-mobile sensing platforms.

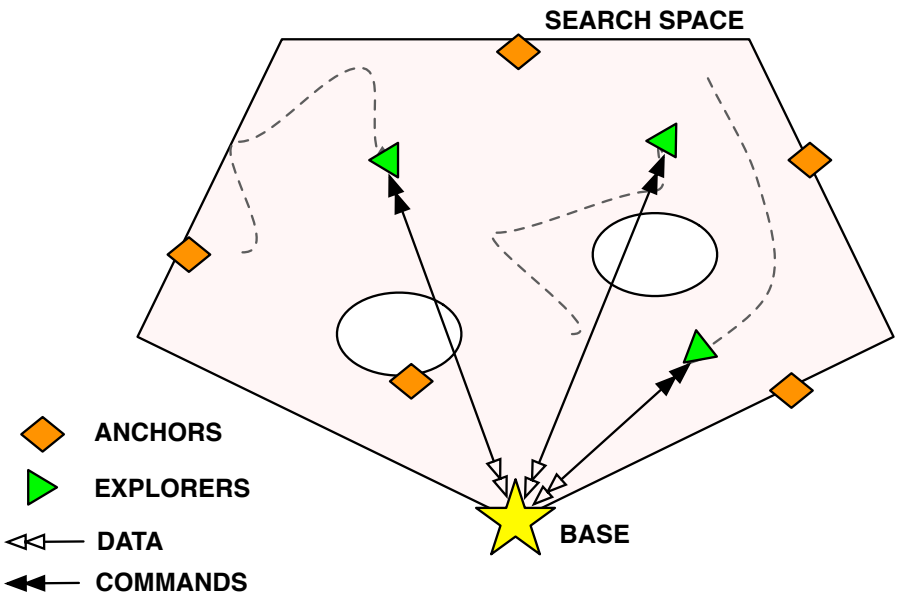


Figure 5.2: An illustration of the SugarMap system operating environment.

5.1.2 Operating Environment

The predominant application of proposed controlled-mobile swarms is in attaining sensing coverage of unknown environments that are inaccessible or hazardous for humans to enter. For example, in applications such as urban surveillance, survivor search after disasters, or nuclear radiation monitoring.

Figure 5.2 gives an illustration of the operating environment. The anchors are deployed by dispersion at the boundaries, explorers move and sense the area, and the base station received data from explorers and transmits commands. We make the following assumptions about the operating environment from the application scenario:

- The initial position and pose of swarm nodes is known. The swarm nodes are introduced into the operating environment manually. For example, a firefighter introduces nodes into an indoor structure damaged by an earthquake through an accessible open-

ing.

- The search space for a single group of swarm nodes is continuous. The swarm nodes do not have the capability to break through obstructing structures.

5.1.3 System Components

The SugarMap system has 3 major components:

- **Anchors** – Anchors are controlled-mobile nodes that the system lands at initialization in the environment. The nodes land using a simple dispersion algorithm where they seek to approximately spread out from each other but at the same time maintain radio connectivity with all other anchors. The dispersion does not use any location information but relies on the approximate proximity information provided by the radio time-of-flight measurements.

- **Explorers** – Explorers are controlled-mobile nodes that move and sense the environment. The nodes collect radio signatures from the anchor nodes, execute commanded motion with feedback from their motion sensors, sense the environment using application specific sensors and relay this data to a coordinating base station. The nodes receive high-level commands from the base station to follow a movement path.

- **Base Station** – The base station is a node at a safer location with access to higher computing power than the swarm nodes. The nodes relay information to the base station. The base station computes the probabilistic coverage from obtained data and directs the motion of all the explorer nodes as per the coverage algorithm. The base station provides a real-time coverage status to the users of the swarm.

5.1.4 SugarMap Coverage Algorithm

The base station runs the SugarMap online coverage algorithm. Every node in SugarMap starts with an empty grid map of the world with a known initial position. The size of a grid cell is equal to the sensing radius of each sensor. As explorer nodes move, they approximately measure their motion using sensors (optical flow and inertial), and collect RToF signatures from landed anchor nodes, and relay this data to the base station.

The SugarMap algorithm directs explorer nodes to perform a depth first search (spanning tree coverage [55]) of the environment but to avoid areas already covered by other nodes. This minimizes sensing overlap and speeds up coverage. To achieve this, the algorithm constructs a common grid map for areas covered by all nodes.

However, the motion estimation of controlled-mobile nodes is approximate and exact coverage area is hard to determine. The algorithm uses a particle filter to model the coverage uncertainty (due to motion and sensing uncertainty) of each SensorFly node. Every node is represented by n particles that track the coverage path of the SensorFly on the grid, where the weight of each particle gives the probability of each cell on its path being covered. The algorithm detects revisits in node paths by matching RToF signatures and uses this to update particle weights. The algorithm combines the probabilistic coverage map of all particles by summing and normalizing the cell coverage probabilities to arrive at a coverage map for each SensorFly.

Thus, the SugarMap algorithm constructs a probabilistic spanning tree for each node's coverage. Each node of the spanning tree corresponds to a unique RToF signature of a covered area, while the edges are defined through the relative motion estimates. Using the common initial position of all nodes, individual maps are overlaid on a common grid map by again combining the probabilities of individual cells to arrive at a global coverage map

of the swarm. The algorithm uses this global coverage map to direct nodes to cells with a lower probability of having been covered.

5.2 SYSTEM DESCRIPTION

In this section, we describe the details of the different components and algorithms used in SugarMap. We discuss the deployment of anchors, the path planner that commands explorer nodes, and the particle filter based probabilistic coverage map estimation.

5.2.1 Deployment of Anchors

The SugarMap system initializes by deploying a subset of controlled-mobile nodes in the environment to act as radio anchors. The absolute position of the anchor nodes is not critical. However, it is desirable for the anchors to be dispersed over the search space to provide robust RToF signatures. Dispersion algorithms for constrained robots have been studied in prior work [56, 57, 58] and many potential approaches can be employed. In SugarMap, the anchors deploy using a *fiducial dispersion* algorithm [56] based on the nodes' ability to obtain RToF measurements from other nodes. The objective of the dispersion algorithm is for nodes to spread away from each other till they encounter obstacles or lose radio range with other anchors.

Figure 5.3 shows a flowchart of the algorithm used to deploy anchor nodes. The base station commands a sub-set of controlled-mobile nodes (designated as anchors) to move in randomly chosen directions. Every node periodically attempts to obtain radio RToF measurements from other anchors and relay it to the base station. If any anchor node is out of radio range and does not respond, the node attempts to retrace its path by turning $180°$. Conversely, if all nodes are in range, each RToF measurement (d_i) is compared to

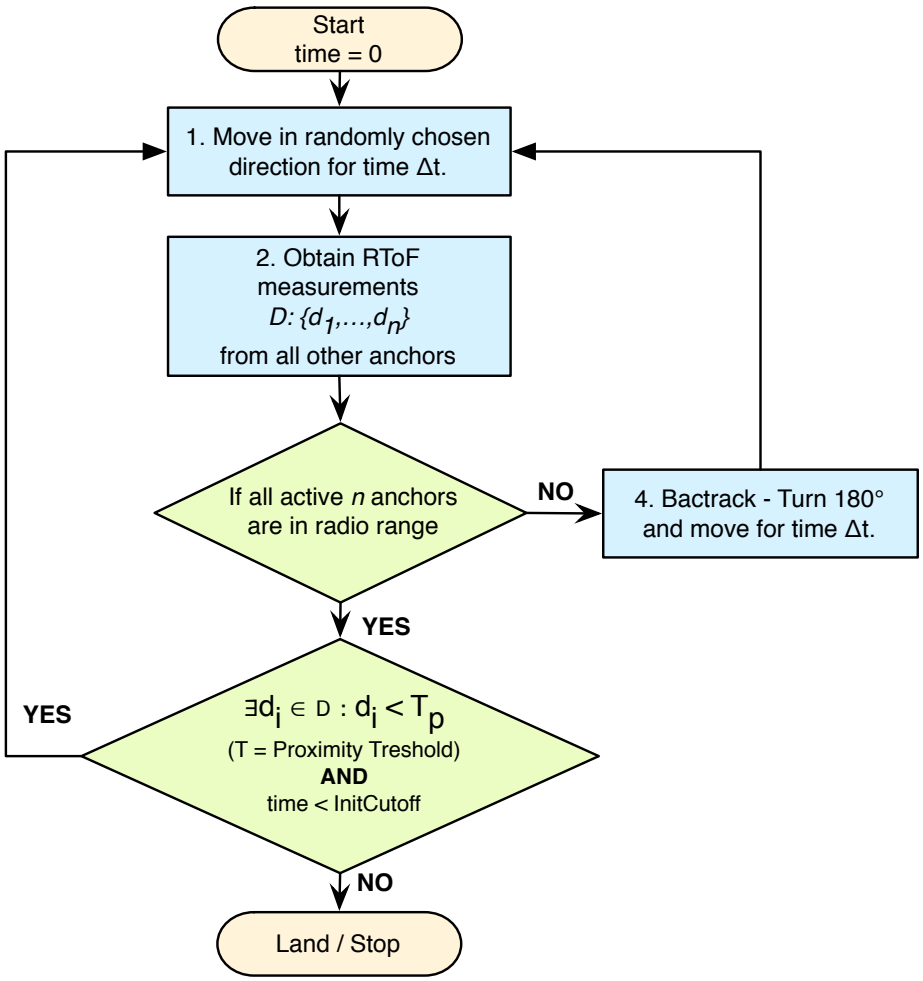


Figure 5.3: Figure shows a flowchart of the constrained dispersion algorithm used to deploy anchor nodes at initialization of SugarMap.

a proximity threshold (T_p). If no other anchor is within the proximity threshold, the node lands and deploys. Otherwise, the node moves in a random direction and recursively repeats the above sequence of operations.

In addition, a maximum cutoff time for anchor deployment ($InitCutoff$) is defined. If an equilibrium is not reached within the cutoff time, the nodes still land and deploy.

It must be noted that the explorer nodes use RToF measurements from the anchor nodes as a signature and not as a measure of distance. The uniqueness of the signature is a feature of the position of the anchors and multi-path radio propagation characteristics of the physical environment. The system relies on measurements from a relatively large number of anchor nodes (at least greater than 4) to improve signature uniqueness. The system does not assume any particular anchor node topology, such as non-linear, as would be the case for computing location coordinates from deployed anchors.

5.2.2 Coverage Path Planner

The central idea of the SugarMap coverage algorithm is to coordinate explorer node movements so as to cover the environment with a certain measure of confidence.

In the application scenario, the explorer controlled-mobile nodes have no a priori knowledge of the search space. However, due to manual placement, the relative pose and position of the explorer nodes is known. SugarMap approximates the world with a grid with a cell size of D, where D is determined by the effective radius of the application specific sensor deployed on the controlled-mobile node.

A controlled-mobile node is commanded to move along 4 basic direction relative to itself – N, S, W and E, and must be located to within the D-size cell. For simplicity, we first describe the algorithm assuming nodes can be localized to the D-size cell with a certain

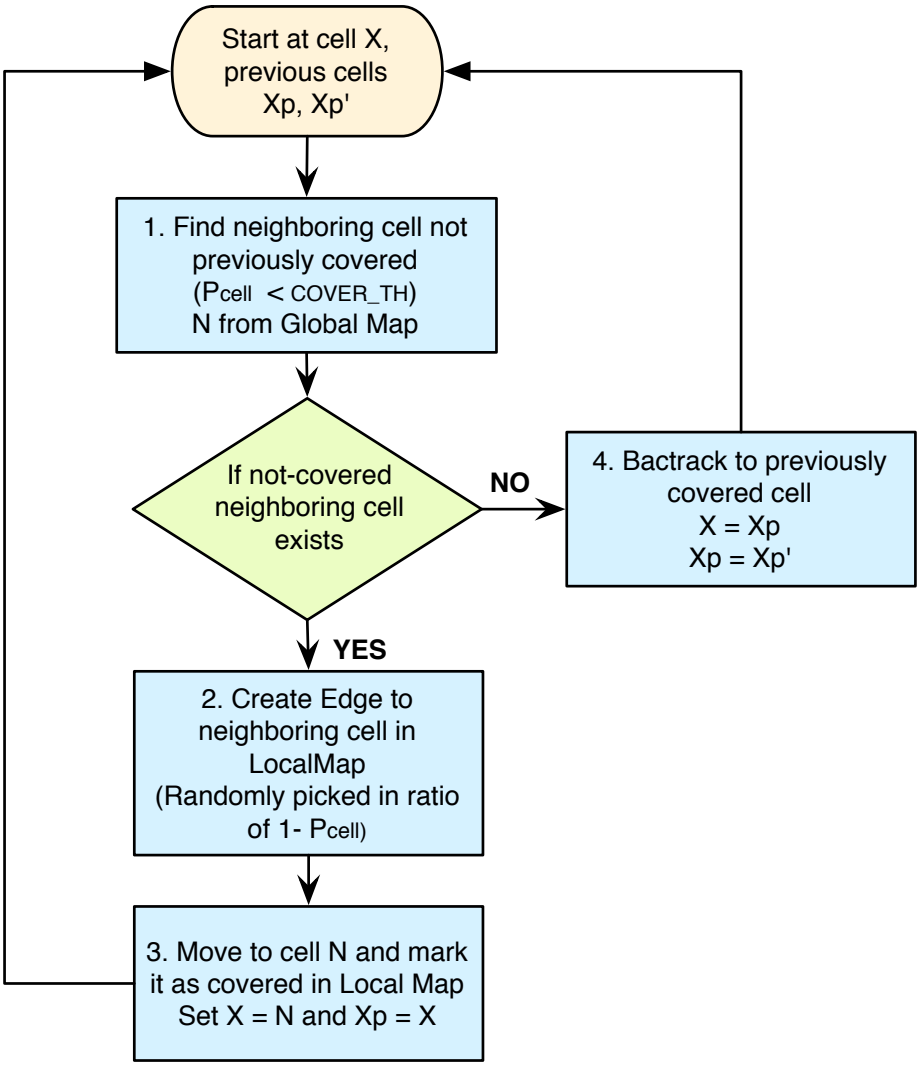


Figure 5.4: The figure shows a flow chart of the SugarMap coverage path planner algorithm. The distributed algorithm runs on every node and builds a spanning tree of covered nodes using a depth first exploration approach.

probability. In section 5.2.3, we explain how this is achieved by the SugarMap system in practice.

Each controlled-mobile node builds a local spanning tree of cells that it discovers, while tracking the portion of the map covered by other nodes through a global map. The global map is obtained by aggregating the local maps from all nodes in the swarm. The spanning tree is built using a depth-first approach – 1) find a neighboring cell that hasn't been covered(by itself or by any node in the swarm); 2) create a tree edge to the neighboring cell; 3) move to the cell and mark it as covered in its local map; recursively repeat steps 1,2,3 with this cell; 4) if all the neighbors are covered or blocked (obstacles), the node backtracks along its local spanning tree to the previously covered cell.

To account for the uncertainty in sensing and position of node's, the coverage of a cell is not designated by a binary value but a real number between 0 and 1, representing the probability that a cell is covered. The algorithm considers a cell as not covered if the coverage probability of a cell (P_{cell}) is less than a desired coverage confidence threshold (*COVER_TH*). The algorithm chooses the next cell from neighboring non-visited cells randomly, where the probability of picking a cell is inverse of its confidence of coverage. When all neighboring cells are above the coverage confidence threshold, the algorithm increments the probability of picking the last covered cell biasing the node towards backtracking on its traversed path.

Figure 5.4 shows a flowchart of the coverage path planner algorithm. The algorithm is distributed and runs on every node. The nodes are coordinated using the global map that is aggregated from each node's local coverage estimates.

5.2.3 Probabilistic Coverage Maps

The path planner requires each node to mark coverage in a grid map of the search space. In addition, the planner requires a global coverage map that is obtained by aggregating the coverage maps of individual swarm nodes.

As the initial location of all nodes is known, the system can potentially track the cells covered by nodes keeping track of the commanded motion of the controlled-mobile nodes. However, controlled-mobile nodes have low-quality inertial sensors and imperfect actuators. This makes it impossible for controlled-mobile nodes to accurately execute the commanded path. SugarMap uses a particle filter based probabilistic approach that combines approximate motion noise models of the explorer nodes with radio round-trip time-of-flight (RToF) position signatures, to create a common probabilistic coverage map for the swarm.

5.2.3.1 Particle Filter (PF)

A particle filter (PF) [59] is a Bayesian estimation method used to estimate a system's state based on noisy sensor information. In a particle filter, a probability distribution $p(x)$ is represented by a number of N weighted samples or *particles* $x^{[i]}, i = 1..N$, with weights $w^{[i]}$ as:

$$p(x) = \sum_i w^{[i]} \delta(x^{[i]} - x) \tag{5.1}$$

With a initial probability $p(x_0)$, which is represented as equally distributed samples with equal weights, a recursive update at time t_k to estimate system state x_k is performed in 3 steps:

1. **Prediction** – Every particle $(x_{k-1}^{[i]}, w_{k-1}^{[i]})$ of the *a posteriori* distribution $p(x_{k-1}|z_0, \ldots, z_{k-1})$, where $z_0, \ldots, z_k$ are measurements about the system up to time t_k, is replaced accord-

ing to a process model $p(x_k|x_{k-1})$. The process model incorporates knowledge of the evolution of the system over time. In coverage estimation, we use an empirically obtained actuation noise profile from the controlled-mobile nodes' motion sensors to construct this model. Thus a new set of particles $(\tilde{x}_k^{[i]}, \tilde{w}^{[i]})$ is obtained representing the *a priori* distribution.

2. **Correction** – The weight $w^{[i]}$ of every sample of the *a priori* distribution, is updated according to a measurement model as:

$$w^{[i]} = \tilde{w}^{[i]} \cdot p(z_k|\tilde{x}_k^{[i]}), \sum_i w^{[i]} = 1 \tag{5.2}$$

With the weight update, the prior particles now approximate the *a posteriori* probability. In coverage estimation, we use the radio RToF signatures obtained by the explorer nodes from the anchor nodes to compute the term $p(z_k|\tilde{x}_k^{[i]})$, which relates the coverage state of the system to its observation.

3. **Resampling** – A new set of particles is drawn with replacement from the prior set with probability of a particle being drawn given by its weight. The samples are weighted equally. The resampling step prunes the less likely state estimates.

5.2.3.2 Applying PF To Coverage Estimation

In this section, we describe the application of a particle filter to estimate coverage in SugarMap. We first consider a single controlled-mobile node. Each controlled-mobile node is represented by N particles. Each particle holds an array ($LocalMap$) representing the particle's coverage on a local grid map of the search space. Cells not covered are set to 0 in the array, while covered cells are set to 1. The state of a particle $x_k^{[i]}$ at time t_k is given by this $LocalMap_k^{[i]}$.

Prediction: In the prediction step, the $LocalMap_k^{[i]}$ is updated according to the commanded motion of the controlled-mobile node and a actuation noise model. The controlled-mobile node executes a motion command – turn and velocity, using feedback from its inertial sensor (gyro) and optical flow velocity sensor. Therefore, if c_x and c_y are coordinates of the last cell covered by a particle, v_k is the velocity, ϕ_k is the change in pose, the $LocalMap_k^{[i]}$ is calculated as:

$$\begin{pmatrix} c_x \\ c_y \end{pmatrix}_k^{[i]} = \begin{pmatrix} c_x \\ c_y \end{pmatrix}_{k-1}^{[i]} + (v_k^{[i]} \cdot \delta t) \begin{pmatrix} sin(\phi_k^{[i]}) \\ cos(\phi_k^{[i]}) \end{pmatrix} \tag{5.3}$$

$$LocalMap_k^{[i]} \left[c_{x_k}^{[i]}\right] \left[c_{y_k}^{[i]}\right] = 1 \tag{5.4}$$

Noise is added to the velocity and turn commands as per the empirically obtained actuation noise models $p(n_v)$ and $p(n_\phi)$. This is based on the actuation mechanism of the controlled-mobile platform used. Thus, $v_k^{[i]}$ and $\phi_k^{[i]}$ are obtained as:

$$v_k^{[i]} = \quad v_k + n_v^{[i]} \ , \ n_v^{[i]} \text{ is drawn from } p(n_v) \tag{5.5}$$

$$\phi_k^{[i]} = \quad \phi_k + n_\phi^{[i]} \ , \ n_\phi^{[i]} \text{ is drawn from } p(n_\phi) \tag{5.6}$$

Correction: In the correction step, the weight of the particles is updated using the radio RToF area signatures obtained by the controlled-mobile node. A detailed explanation of the radio RToF area signatures is given in Section 5.3.4.

The central idea of the correction step is to compare the radio RToF signature s^k obtained at time t_k to a set of known area signatures $S : \{s^1, \ldots, s^j\}$. The set S is empty at initialization. A list of location estimates $(c_x^{[i]}, c_y^{[i]})$ from every particle is stored for every signature in set S.

A distance function $f(s^k, s^j)$ gives the distance between the currently obtained signature and previously known signatures in set S. If distance given by $f(s^k, s^i)$ is more than a threshold (SIG_TH) for all signatures in set S i.e. the obtained signature is of an unexplored area, signature s^k is added to S. Conversely, if the distance is less than the threshold, the signatures are considered similar and the area is designated as a previously covered area.

On identifying a matching previously visited signature s^j, the corresponding distance in location estimates for the current signature s^k and known signature s^j for each particle is computed as:

$$d^{[i]}_{s^j,s^k} = EuclideanDist\left\{(c_x, c_y)^{[i]}_{s^j}, (c_x, c_y)^{[i]}_{s^k}\right\} \tag{5.7}$$

Consequently, the weights of the particles are updated as a function of the distance between the two estimates as:

$$w^{[i]} = \frac{\left\{\sum_i^N d^{[i]}_{s^j,s^k}\right\} - d^{[i]}_{s^j,s^k}}{\sum_i^N d^{[i]}_{s^j,s^k}} \tag{5.8}$$

Resampling: In the resampling step, a new set of particles is drawn with replacement using the weights as the probability of drawing. The weights are reset to their initialization values.

Merging: Finally, the local maps of each particle are combined to compute a node coverage map at time t_k as:

$$NodeMap^j_k = \frac{\sum_{i=1}^N LocalMap^{[i]}_k}{N} \tag{5.9}$$

Similarly, the central base station combines the NodeMaps from all controlled-mobile nodes to arrive at the GlobalMap for the swarm:

$$GlobalMap_k = 1 - \prod_{j=1}^{n} \left(1 - NodeMap_k^j\right) \tag{5.10}$$

where n is the number of nodes in the swarm.

5.3 EVALUATION

In this section we evaluate the capability of SugarMap to command a swarm of node's collaboratively cover a search space in realistic large-scale simulations and in a real controlled-mobile testbed. We characterize the performance of our system with respect to the percentage of coverage achieved by a fixed number of nodes as a function of time. Percentage coverage as a function of time is an essential metric in applications such as disaster response, where the speed of covering an area and identifying survivors is critical to the success of the operation. Even random walk algorithms eventually attain complete coverage of an area, however we seek to minimize the time for coverage through coordinated node movement approaching that of an algorithm with accurate location measurements.

Due to the limited suitable controlled-mobile coverage algorithms available in literature, we compare SugarMap to currently available online coverage algorithms for controlled-mobile nodes namely random walk coverage [60], and Online Multi-robot Spanning Tree Coverage (OMSTC) [61] that assumes accurate node location is available.

5.3.1 Simulation Setup

We utilize our simulation environment for the SensorFly controlled-mobile indoor sensor swarm to evaluate our coverage algorithm at scale in a realistic scenario. The simulator

incorporates a realistic physical arena, controlled-mobile node virtual sensors and sensor noise models, controlled-mobile node mobility models, wireless communication and radio path loss model, and application specific sensing models. These models are generated through data collected from controlled-mobile nodes in indoor environments. In addition, the simulator allows users to program the logic for actuation of node's and implement coverage algorithms such as SugarMap or Random Walk. The simulator, now ported to Python, extends previous work [62] to include additional application scenarios and adds the ability to interface with actual hardware and run hardware-in-loop simulations.

To simulate SugarMap we configure the different aspects of the simulator as follows:

- **Arena** – We assume an indoor search and rescue scenario, where nodes are required to move and cover a continuous indoor space such as a room. The simulation arena constitutes the search space and we evaluate SugarMap in a $20m \times 20m$ square arena, but with multiple configurations of boundary walls and obstacles.

- **Node Sensors** – We model the controlled-mobile node based on the SensorFly [54] controlled-mobile platform used for our real implementation. The node is equipped with 3 virtual sensors: an inertial gyroscope sensor, an optical flow velocity sensor, an ultrasonic altitude measurement sensor, and a radio with round-trip time-of-flight measurement capability. The application sensor has a sensing footprint equal to a square cell of $0.5m \times 0.5m$. This corresponds to the cell size of the grid map used by SugarMap to compute coverage.

- **Node Mobility** – Like the SensorFly nodes, the simulated nodes can be commanded to turn by a desired angle and move forward for a designated time. The nodes execute the commands with feedback from their virtual sensors, incorporating the errors from

their associated noise models. The nodes operate at speeds of $0.25m/s$ to $0.45m/s$.

- **Anchor Nodes** – We model the deployment of anchor nodes as described in Section 5.2.1. The simulation clock starts after initialization of the system and anchor node deployment.

- **Radio Model** – Shadowing with a path loss exponent of 3 is used as the radio link model, which is an estimate for an indoor single-floor scenario [37]. The movement algorithm ensures that nodes maintain connectivity.

- **Simulation Time-steps** – The simulation time-step is configurable and determines the resolution of node movement that can be recorded by the simulator. The nodes execute their movement based on their velocity, specified in meters per simulation-seconds. Therefore, the minimum distance that can be achieved by the node in one command is determined by the duration of the simulation time-step. For the purpose of the evaluation, we selected a time step of $1sec$ that enables nodes to cover a distance of $0.25m$ to $0.45m$ in a single simulation tick. In terms, of sensing footprint ($0.5m \times 0.5m$), a node can travel from one sensing cell to another in a single tick.

The coverage algorithm or the base-station *brain* of the swarm does not receive the real physical locations of the nodes but only their presumed locations based on movement commands. The nodes are also not aware of the map and are assumed to be at the center of an infinite space at initialization. However, nodes are aware of their relative initial positions with respect to each other in accordance with the manual deployment at the entrance to the search space.

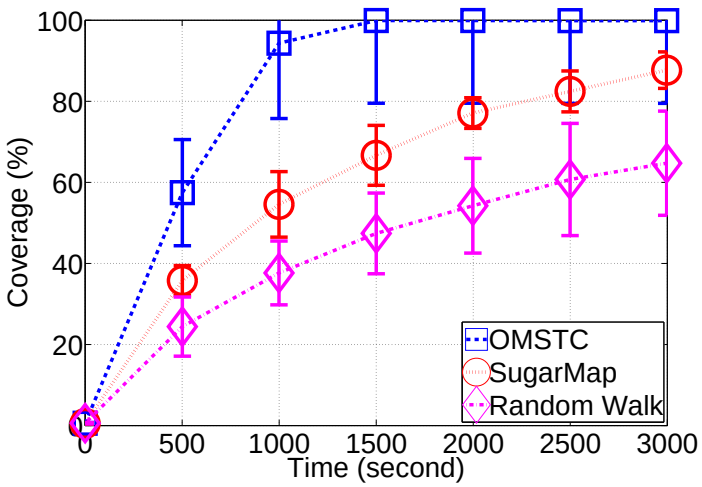


Figure 5.5: The figure shows the percentage coverage of two nodes as a function of time for the SugarMap algorithm, Random-Walk, and a spanning tree coverage algorithm – OMSTC that assumes locations of nodes are known. The error bars show the standard deviation over 10 runs.

5.3.2 Comparing Coverage

We evaluate the performance of SugarMap by comparing the percentage coverage achieved as a function of time to state-of-the-art existing online coverage approaches applicable to controlled-mobile swarms.

We implement two approaches – random walk and online multi-robot spanning tree coverage (OMSTC) (assumes locations of nodes is known with high accuracy and precision). Random walk [60] is the most popular approach used by resource-constrained nodes when no location information or prior knowledge of the environment is available. This provides us a baseline for comparison. On the other hand, OMSTC [61] is a guaranteed multi-node coverage algorithm that assumes that nodes can locate themselves in the space. Although, perfect location is unattainable, this presents us with an ideal system for comparison.

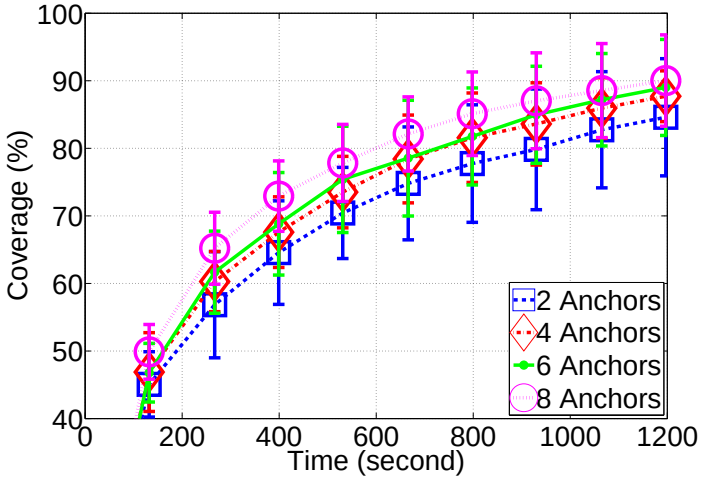


Figure 5.6: The figure shows the evolution of coverage for a varying number of deployed anchor nodes with 4 explorer nodes. The error bars show the standard deviation over 10 runs.

Figure 5.5 shows the percentage coverage as a function of time for SugarMap, Random-Walk (baseline) and OMSTC (ideal). The simulation uses 4 explorer nodes for all algorithms and runs for 3000 simulation seconds. SugarMap performs better than random walk achieving a faster rate of coverage in the simulation scenarios.

5.3.3 Analyzing SugarMap

In addition to comparing coverage with other approaches, we analyze the trade-offs involved in selecting parameters for a SugarMap deployment.

5.3.3.1 Impact of Number of Anchors

The SugarMap system uses anchor nodes to obtain a radio signature for various covered areas. Individual radio measurements (round-trip time-of-flight) are subject to variability as shown in Figure 5.14. However, a signature combining measurements from multiple

anchors is more stable.

Figure 5.6 shows the coverage achieved as a function of time while the increasing number of anchors deployed by the system. The number of explorers is set at 4 and the number of particles used for each SugarMap node is 10. A stable area signature enables SugarMap to determine visited locations better and compute better trajectories for the swarm nodes. Therefore, coverage improves with an increase in the number of anchors. However, depending on the variance in individual measurements, the benefit of increasing the number of anchors diminishes after a point. The number of anchors also depends on the size of the area.

With fewer anchor nodes (2 nodes in Figure 5.6), the radio location signatures are not sufficiently unique and the accuracy of coverage estimation suffers. Without estimation of coverage, the path planning algorithm cannot efficiently compute motion and coverage achieved shows high variance similar to a random walk algorithm.

5.3.3.2 Impact of Number of Explorers

Figure 5.7 shows the coverage achieved as a function of time for a varying number of explorer nodes. 6 anchor nodes are used for the simulation. As nodes explore the environment in parallel, the larger the number of explorer nodes the greater is the coverage. Moreover, unlike heuristic algorithms, SugarMap coordinates node movements to reduce the overlap in area coverage. Thus, increasing the number of nodes shows an almost linear increase in speed of coverage. However, due to noise in sensor and actuation, some overlap in individual node coverage exists in SugarMap. Therefore, the benefit of increasing explorer nodes diminishes after a point.

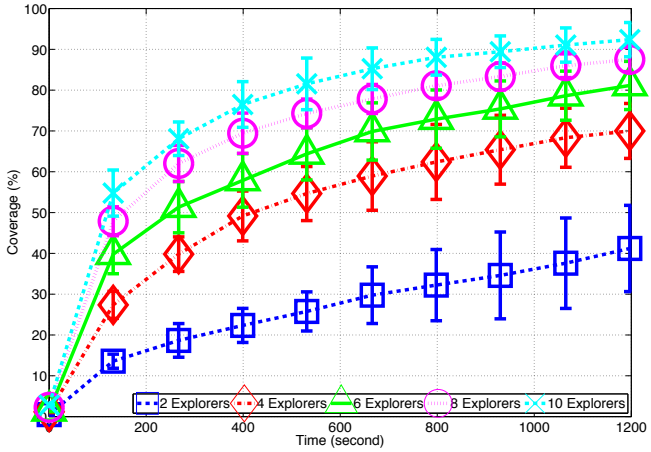


Figure 5.7: The figure shows the evolution of coverage for varying number of explorer nodes with 6 anchor nodes.The error bars show the standard deviation over 10 runs.

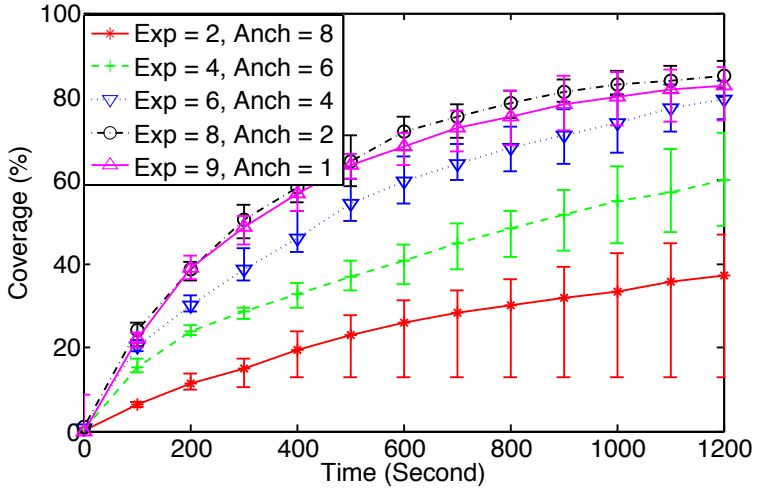


Figure 5.8: The figure shows the percentage coverage for fixed size 10-node deployment with varying number of anchors and explorers.

5.3.3.3 *Anchor-Explorer Trade-off*

Figure 5.8 shows the rate of coverage for varying ratio of anchor nodes to explorer nodes for a fixed-size (10-node) deployment in a $20m \times 20m$ arena. The plot shows the trade-off between anchor nodes and explorer nodes. It is evident that for a given size deployment, the coverage rate is determined largely by the number of explorer nodes. The larger the number of explorer nodes the faster the coverage attained. However, the incremental performance gain of introducing additional explorer nodes diminishes when the number of anchors is reduced below 4. With a small number of anchor nodes, the radio location signatures are not sufficiently unique and the accuracy of SugarMap coverage estimation decreases with time. Consequently, the nodes cannot coordinate efficiently and performance drops.

5.3.3.4 *Impact of Number of Particles*

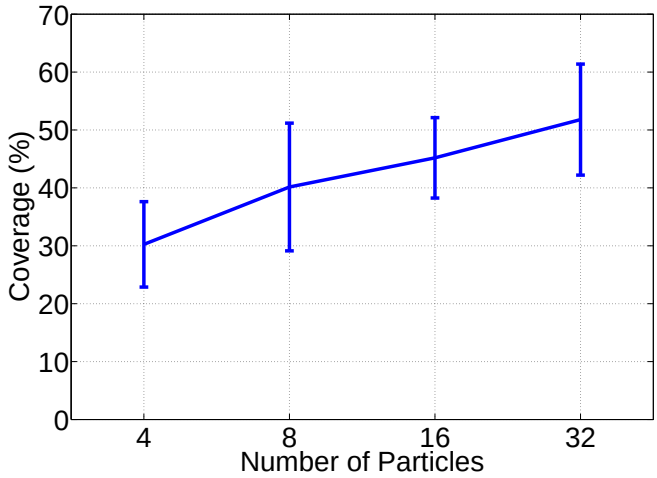
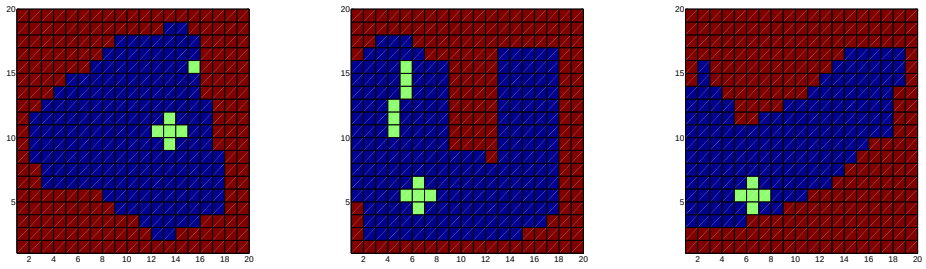


Figure 5.9: The figure shows the coverage achieved in 400 simulation ticks as a function of the number of particles used in the SugarMap algorithm. The number of explorer nodes is fixed at 2 and number of anchors nodes is 6. The error bars show the standard deviation over 10 runs.



(a) (b) (c)

Figure 5.10: The figure shows 3 geometries of the simulation arena with different obstacle configurations used to evaluate the robustness of the SugarMap algorithm. The red cells are walls, the blue cells represent navigable space, and the green cells designate obstacles.

SugarMap uses a particle filter to model the uncertainty in actuation of controlled-mobile nodes. Each node is represented by a number of particles, each of which holds an estimate of the node's coverage map. Figure 5.9 shows the percentage coverage achieved by a deployment of 2 explorer nodes and 6 anchor nodes in 400 ticks of the simulation, while the number of particles used is varied. The greater the number of particles, the better is the estimation of the area covered by SensorFly's. This translates to a lower overlap in sensed area and a higher speed of coverage. As a trade-off, the larger number of particles require higher memory and computation at the base station. In addition, the choice of particles depends on the sensor and actuation noise of the controlled-mobile node. Larger sensor noise requires higher number of particles.

5.3.3.5 Impact of Search Space Geometry

We evaluate the performance of SugarMap in multiple space configurations. The simulation environment enables the programmer to specify walls (red cells), open navigable area (blue cells) and obstacles (green cells) by providing a bitmap image. Figure 5.10 shows 3 different geometries used to evaluate SugarMap. Geometry (a) is an almost circular open

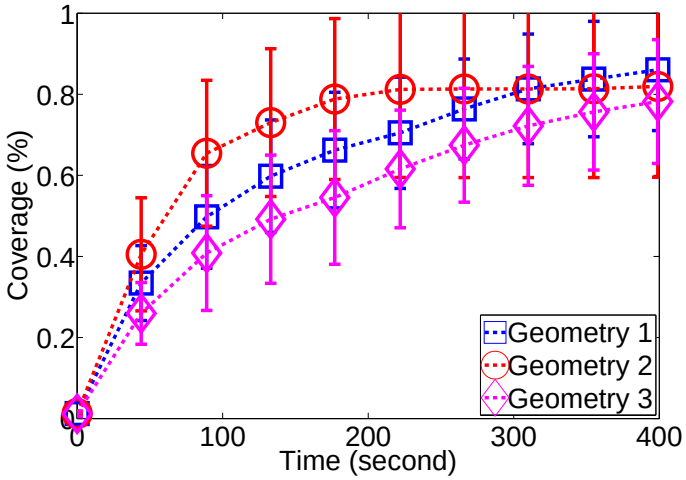


Figure 5.11: Figure shows the coverage as a function of time for the SugarMap algorithm in 3 different geometries (Figure 5.10) of the simulation arena. The error bars show the standard deviation over 10 runs.

space with a few obstacles. Geometry (b) is a U-shaped region with a narrow leg similar to a room with a connecting hallway. While, geometry (c) models a narrow corridor.

Figure 5.11 shows the coverage as a function of time for the 3 different arena geometries. The curves for the 3 spaces follow each other closely showing that SugarMap is robust to variations in space configuration.

5.3.3.6 Algorithm Parameters

The SugarMap algorithm has two principal parameters – (1) a threshold to match radio signatures (SIG_TH), for determining if an observed signature is similar to a previously observed one; and (2) a threshold to select the next location for movement ($COVER\_TH$), for determining the confidence of coverage for a neighboring location as per the global coverage map.

The SIG_TH is determined empirically based on the radio feature and metric used to

compute the similarity of signatures. The SugarMap system implementation uses a vector of RToF (Round-trip Time-of-Flight) measurements from a set of anchor nodes as the radio signature. Purohit et. al. [54] provide a more detailed evaluation of RToF measurements and their correlation with distance. We use Euclidean distance as a metric to measure similarity between N-dimensional RToF signatures, given by,

$$d_{i,j} = \frac{\| \overrightarrow{s^i} - \overrightarrow{s^j} \|}{\#\text{ of visible common anchors}} \tag{5.11}$$

For the purpose of our evaluation, we selected a SIG_TH of 1.2, obtained by measuring the average RToF signature distance for 20 signatures in a 2-meter radius circle, over 5 distinct locations in our lab.

The $COVER\_TH$ is defined as an exit condition for the path planning algorithm. Since, the coverage map represents confidence probabilistically it does not achieve 100% coverage confidence for a location in finite time. The parameter determines the coverage confidence that is desired by the application for the path planner to avoid revisiting a location. We chose 95% as the coverage threshold for our evaluation signifying a high degree of certainty that the area is covered.

5.3.3.7 Actuation Noise Model

The SugarMap algorithm uses particle filters to account for the uncertainty in the movement actually executed by controlled-mobile nodes on a given command. A model of the actuation noise is required in the prediction step of the particle filter as described in Section 5.2.3.2. The actuation uncertainty depends on the sensors and control algorithm for the specific controlled-mobile platform. For the purposes of the evaluation, we use the SensorFly platform to empirically determine an approximate noise model. The Sensor-

Figure 5.12: The figure shows a SensorFly node used to implement SugarMap on the prototype testbed.

Fly platform uses PID control with feedback from an optical flow sensor (velocity) and a gyro (turn) to execute the commanded motion. We measured the standard deviation in turn executed by the platform using a vision-based ground truth measurement relative to the commanded turn value to derive a model. For the evaluation, we used a normal distribution with a standard deviation of 20% of the commanded turn value to predict turn noise. Similarly, a normal distribution with a standard deviation of 15% of the commanded velocity value was used as the velocity noise model.

5.3.4 Controlled-mobile Testbed

In addition to the simulation experiments, we evaluate SugarMap in our controlled-mobile swarm testbed. We implement SugarMap on the SensorFly [54] controlled-mobile platform.

The SensorFly platform is equipped with a 8-bit 16Mhz AVR AtMega128rfa1 microcontroller, inertial motion sensors – 3-axis accelerometer and 3-axis gyro, an ultrasonic ranger for altitude estimation, an optical flow sensor for velocity estimation, and a 802.15.4a compatible radio with Round-trip time-of-flight (RToF) measurement capability. The en-

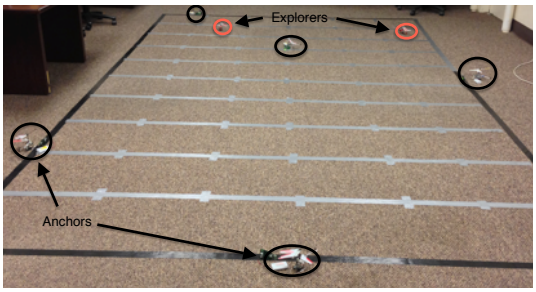


Figure 5.13: The figure shows the arena for the controlled-mobile swarm testbed with deployed SensorFly nodes.

tire platform is under $30g$ in weight and has a flight time of 6-8 minutes. The SensorFly nodes are capable of receiving high-level movement commands such as *"Turn X degrees"* and *"Move forward X seconds"*. The nodes execute the movement in accordance with on-board PID control algorithms utilizing angular and translational velocity feedback from the nodes' gyro and optical-flow sensors. Figure 5.12 shows a SensorFly node with the battery and basic set of sensors.

The testbed consists of a $5m \times 3m$ arena where the SensorFly nodes move. A grid is painted on the arena dividing it into cells of $0.5m \times 0.5m$. A camera is deployed on the ceiling to capture the entire arena in its field of view. A workstation running a color blob detection algorithm uses the feed from the camera to compute the *ground-truth* location of all controlled-mobile nodes on the grid. The cell size of the grid is chosen to reflect the sensing radius of the controlled-mobile node. The blades of the SensorFly nodes are affixed with red-tape so as to be easily detectable by the vision-based ground-truth tracking system. Figure 5.13 shows the arena of the testbed with deployed SensorFly anchors and explorers.

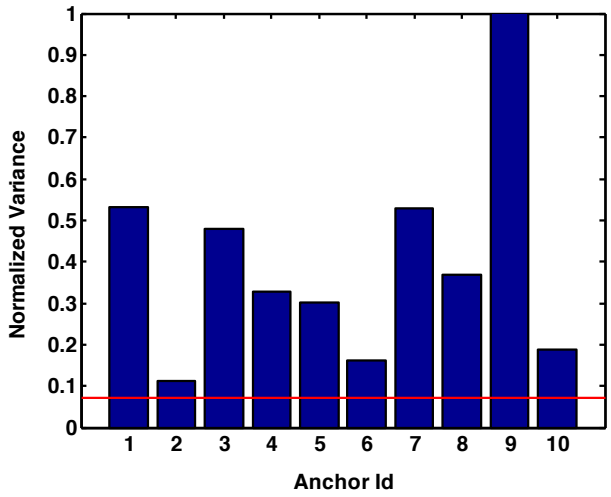


Figure 5.14: Figure shows the normalized variance in RToF measurements from a explorer node and 10 individual anchors (bars) over a 3 day period. The variance of the signature (combined set of 10 individual measurements) is shown by the horizontal line. The low variance of the combined vector makes it suitable as an area signature.

RToF Area Signature Stability

The explorer SensorFly nodes use a set of round-trip time-of-flight (RToF) measurements from stationary anchor nodes as a signature of area covered. The signature enables explorer nodes to determine if a covered area is being revisited. This information is used for by the coverage path planner for coordination and by the probabilistic coverage estimation algorithm for updating particle weights.

The round-trip time-of-flight method measures the elapsed time between the host node sending a data signal to the remote node, and receiving an acknowledgment from it. Our implementation, based on the SensorFly platform, uses physical layer timestamps and hardware-generated acknowledgments to compute a RToF measurements [54, 63].

We perform an experiment to measure the repeatability of a set of RToF measurements and hence validate its suitability as a area signature. RToF measurements from 10 nearby anchors were collected at a mobile node location over a 3 day period, in a lab environment.

The lab is subject to frequent environment changes due to movement and activity of people. Figure 5.14 shows the variance in RToF measurements from each anchor. We observe a large variation in individual RToF measurements over the 3-day period. However, despite these environmental variations, we observe that the overall ten-dimension signature vector of the RToF measurements remains consistent (red-line), showing very low variance over the entire test period.

Testbed Results

In our testbed experiment, we use a swarm of 7 controlled-mobile nodes running the SugarMap algorithm. 5 nodes are allowed to disperse and deploy as anchors at initialization. 2 nodes are used as explorers. The two explorer nodes start at grid locations [1,1] and [2,1], respectively, and move as per commands given by the base-station. The explorer nodes obtain RToF readings from anchor nodes and relay it to the base-station at every step. The actual coverage achieved is computed for evaluation purposes using the ground-truth vision-based tracking system of the testbed. The testbed experiment runs for 3-minutes of node flying time which enables us to execute multiple runs on a single charge.

Figure 5.15 shows the evolution of coverage attained by the 7-node swarm running SugarMap (solid-line). The swarm is able to cover 85% of the arena in the 3-minute experiment time. For comparison, we also plot a simulation result (dotted-line) using a similar size arena, 5 anchor nodes, 2 explorer nodes, and the actuation noise model for SensorFly nodes. The results from the simulation and real experiment closely follow each other as time progresses, validating the simulation setup and methodology.

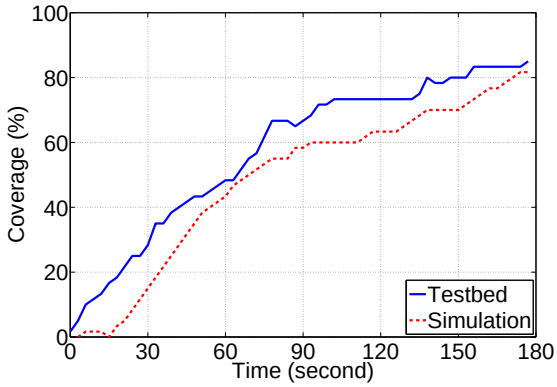


Figure 5.15: The figure shows the evolution of coverage for the 7-node controlled-mobile swarm testbed experiment (solid-line). The result is in agreement with our simulation (dotted-line) using similar arena-size, number of anchors, number of explorers, and the actuation noise model.

5.4 RELATED WORK

The coverage problem has been addressed in the past by research in multi-robot coverage algorithms that focus on the space swept by the robot's sensor. Choset [64] provides a survey of early coverage algorithms and classifies them into *off-line*, in which a map of the area is known beforehand, and *on-line* algorithms, in which the area is unknown. The controlled-mobile swarm applications demand an on-line multi-node coverage approach. In this section, we discuss some of the existing work in online coverage that is applicable to controlled-mobile swarms.

Gage [60] analyzes randomized robot coverage for robots without costly localization sensors or valuable computational resources for calculating their position. A random search does not guarantee coverage but may be the only approach with very low capability sensor nodes in absence of motion sensing.

Wagner et al. [65] propose a pheromone based stigmergic algorithm, where nodes co-

ordinate by leaving markers in the environment. The algorithm is heuristic in nature and requires nodes to be capable of leaving and detecting physical markers in the environment. This is not practical for lightweight controlled-mobile nodes.

A series of simultaneous localization and mapping (SLAM) approaches [66] exist for multi-robot systems that use laser range-finders or computer vision algorithms (to extract visual features) in conjunction with noisy odometry to construct maps of the area and obtain coverage. This approach is very promising for ground-based robots or larger aerial nodes. Currently available laser rangers tend to be too bulky for deployment on node's, while vision-based approaches require better computation and communication capabilities.

Howard et al. [67] considers the problem of deploying a mobile sensor network in an unknown environment. The approach assumes that each node is equipped with a sensor that allows it to determine the range and bearing of both nearby nodes and obstacles, such as scanning laser range-finders or omni-cameras. Using these, the system constructs fields such that each node is repelled by both obstacles and by other nodes, thereby forcing the network to spread itself throughout the environment. The algorithm has the advantage of requiring no communication between nodes but does not guarantee completeness. In Spreading-Out [68], Batalin et al. present an algorithm for robot teams without access to maps or a Global Positioning System (location). The robots are assumed to be equipped with planar laser range-finders, color camera and vision beacons, and robots select a direction away from all their immediate sensed neighbors and move in that direction. The approach is heuristic with the premise that robots must 'spread out' over the environment in order to achieve good coverage.

Hazon et al. [61] present a guaranteed robust multi-robot coverage algorithm based on spanning tree coverage paths. Each robot works within an assigned portion of the work

area, constructing a local spanning-tree covering this portion, as it moves. It coordinates movement with other robots to minimize overlap in coverage. However, the algorithm assumes the robots have access to relative location of all nodes in the area. SugarMap employs the concept of spanning trees to guarantee completeness of coverage but further extends it to remove the requirement for node location. Moreover, SugarMap introduces probabilities into the coverage map to account for and overcome the sensor and actuation uncertainty of controlled-mobile nodes.

Related work in robotics and sensor networks [69, 70, 71] uses particle filters for monitoring robot or human position in indoor environments. Their primary focus is to localize nodes using measurement from motion sensors and observed environmental landmarks. Our approach seeks to guarantee coverage of an unknown space by a swarm of nodes based on the coordinated motion commands, using particle filters to account for the uncertainty in controlled-mobile actuation.

5.5 CONCLUSION

This chapter presents SugarMap, a location-less coverage system that allows a swarm to collaboratively and efficiently attain sensing coverage of a target area with a controlled-mobile swarm. SugarMap does not require sophisticated or bulky sensors, advanced on-device processing abilities, or a pre-existing location infrastructure as do many state-of-the-art prior approaches. We provide a comprehensive evaluation of the SugarMap system through large-scale simulations and a real implementation on the SensorFly [54] platform. Our experimental evaluations show that SugarMap performs significantly better than existing coverage approaches for resource-limited (in terms of sensors, energy, location, processing power) mobile sensing nodes such as random-walk especially in larger locations

with fewer nodes. This algorithm will enable swarms of the new class of realistic resource constrained micro-aerial vehicles in new applications.

CHAPTER 6

DEPLOYMENT

This chapter presents DrunkWalk, a technique for the cooperative deployment of swarms of controlled-mobile sensors in multi-room environments not formerly preconditioned for their operation. The key focus behind this networked controlled-mobile swarm research is to rely on collaboration to overcome the limitations of individual nodes and efficiently achieve system-wide sensing objectives.

DrunkWalk, builds on SugarMap described in Chapter 5, to deploy nodes in multi-room scenarios. In DrunkWalk, similar to SugarMap, the controlled-mobile swarm self-establishes a temporary infrastructure of a few landed controlled-mobile nodes acting as radio beacons. Using radio signature or fingerprints from beacon nodes, the algorithm detects intersections in trajectories of exploring mobile controlled-mobile nodes. The algorithm combines noisy dead-reckoning measurements from multiple controlled-mobile nodes at the detected intersections to improve the accuracy of the nodes location estimates. Most importantly, the algorithm adaptively plans trajectories of controlled-mobile nodes according to the certainty of their location estimates – directing movement to improve location estimates when certainty is low, and directing them to follow a map bias when cer-

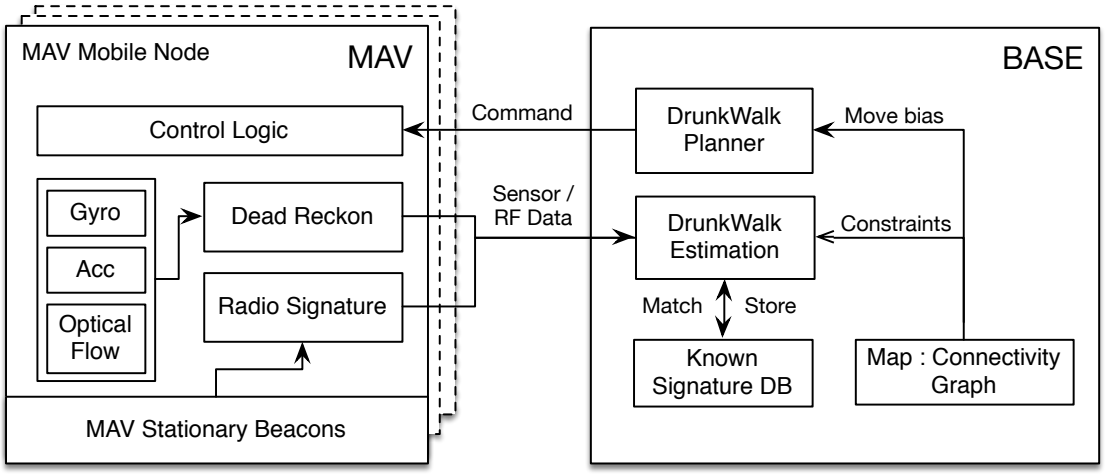


Figure 6.1: The figure shows architecture of our deployment system. The mobile controlled-mobile nodes send dead-reckoning sensor data and radio signatures to a base station. The base station runs the DrunkWalk estimation and planning algorithm and issues movement commands to individual controlled-mobile nodes.

tainty of location estimates is high. The adaptive strategy enables DrunkWalk to improve the speed and accuracy of deployment.

6.1 OVERVIEW

Potential controlled-mobile swarm sensing applications will require mobile sensors to autonomously deploy in multi-room operating environments with no localization infrastructure. In this chapter, we address the problem of how a network of mobile sensors can be deployed to pre-determined deployment positions under time and accuracy constraints.

6.1.1 Operation & Architecture

The system begins operation with a swarm of node's being introduced into a multi-room connected space through an opening. We make the assumption that a coarse map of the building is available and can be utilized by domain experts to pre-determine suitable placement of sensors. This is a valid assumption in most scenarios, as emergency response personnel have access to the rough floor-plans of buildings through city registries and thanks to increased availability of indoor maps tailored to location based services (e.g., indoor Google maps). The system uses the rough map to extract a connectivity graph of the various spaces (rooms) in the deployment environment and determine deployment locations.

The proposed system has 3 major operational phases, setup, estimation and planning (the latter two proceed in conjunction) –

- **Setup:** The system autonomously establishes a transient infrastructure of stationary controlled-mobile nodes acting as wireless beacons. These nodes land on being introduced into the area and remain stationary during the deployment process. The objective of the stationary nodes is to enable mobile controlled-mobile nodes to obtain radio signatures or fingerprints of location traversed on their paths. These nodes use a simple dispersion algorithm [56, 72] that lets them spread out in the environment *without any estimation of their location.*

- **Estimation:** The system then desires to estimate the locations of nodes in order to guide them to their deployment locations. To realize this, the system first uses dead reckoning sensors such as an optical flow velocity sensor and gyroscope (in our test controlled-mobile platform) to get a rough estimate of the motion path of mobile

nodes. Second, the system uses radio fingerprints, collected by mobile nodes from the self-established wireless beacons, to determine *rendezvous points*, i.e. points where nodes visit locations already visited by other nodes or by themselves. Finally, the system uses the rendezvous points to combine location estimates from multiple nodes and collaboratively improve location estimates of the entire swarm.

- **Planning:** Having estimated locations, the system commands the nodes to follow a path to subsequent deployment positions. However, the quality of the planned path depends greatly on the accuracy of the initial location estimate of nodes. A bad location estimate will render any attempt to plan a deterministic path useless – *when the nodes don't know where they are, they cannot plan a correct path to their destination*.

  The novel aspect of our system is that it considers the quality of location estimates in planning node paths. The path planner commands node movement such that they increase rendezvous points and potentially improve location estimates when the quality of their estimates is likely to be low. On the other hand, when the location estimates are likely to be more accurate, the planner uses the map to direct them to their designated deployment locations.

Figure 6.1 shows the architecture of the system. The system deploys **Stationary controlled-mobile Nodes** through dispersion that act as wireless beacons. **Mobile controlled-mobile Nodes** explore, obtaining dead-reckoning measurements from their on-board sensors and radio RF-signatures from the stationary beacons. The mobile nodes relay this to a **Base**. The Base stores a database of known radio signatures (**Signature DB**) that is used to determine rendezvous in node paths and apply corrections to their dead-reckoning estimates.

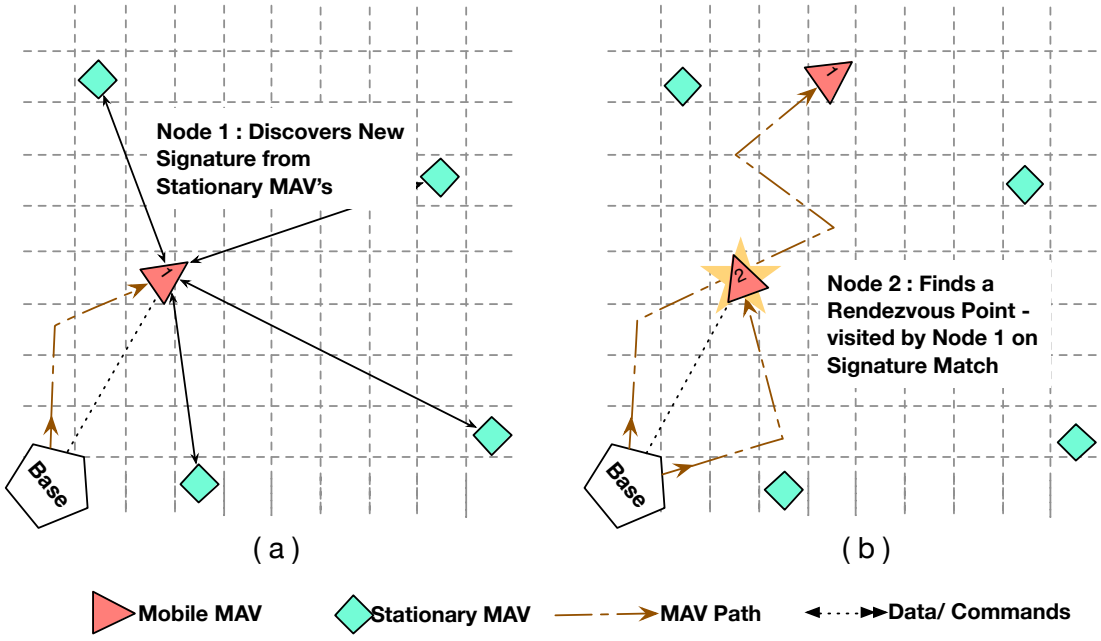


Figure 6.2: The figure shows the process of determining rendezvous points. (a) Node 1 moves and obtains a radio signature from stationary controlled-mobile beacons. This is entered in to the the Base Signature DB as new signature. (b) When Node 2 visits the same location, its collected radio signature matches existing signature and a correction can be performed at the Base.

The corrected location estimates are used by the Base in conjunction with a **Connectivity Graph** (extracted from the coarse map) of the environment to command the subsequent movements of controlled-mobile nodes.

6.1.2 Improving Location Through Swarms

The core idea behind our estimation approach is to use the relatively large number of mobile sensors in the swarm to collaboratively reduce the error. This is achieved by detecting when nodes move over the same space in the environment and combining their individual location estimates at these points. Errors in dead reckoning measurements are

mainly due to noise in inertial sensors that are independent across nodes and time [73]. Thus, combining estimates from multiple nodes and propagating corrections to them, improves their location estimates. Figure 6.2 illustrates the on-line process of determining rendezvous from radio measurements.

6.1.2.1 Determining Rendezvous Points

The location estimation requires a node to be able to determine when it visits a location visited previously by itself or by another node - *a rendezvous*. The rendezvous point provides the opportunity to combine estimates from multiple independent mobile nodes and improve location estimates.

The system determines rendezvous using radio fingerprints collected by mobile nodes from the self-established beacon nodes. The radio fingerprints are collected in an *online* fashion, i.e., the nodes discover fingerprints as they explore. These fingerprints are sent to the Base and matched with a database of previously discovered signatures. If the signature matches an existing signature in the database (decided by a distance metric), the point is classified as a rendezvous, and a correction can be applied. If the signature does not match any existing signature, it is added to the database as a new entry.

6.1.2.2 Combining Estimates at Rendezvous Points

The process of combining location estimates must be performed carefully. The naive approach would be to take the average of all location estimates for a particular rendezvous point. However, this approach does not consider the nature of the underlying distribution of noise in location estimates that often do not follow a normal distribution.

Combining estimates is a chicken and egg problem that requires a rendezvous point to estimate and update its own location from visiting mobile nodes, and subsequently, use the

updated location to correct the estimates of the visiting mobile nodes.

To achieve this, we employ a particle filter based approach. A particle filter [74] is a Bayesian estimation method used to estimate system state based on multiple noisy sensor measurements. We use a particle filter to track the position and orientation of each mobile node. Similarly, we use a particle filter to track the position of each rendezvous point as it is discovered and visited by the controlled-mobile nodes. Every visit to a rendezvous point by a mobile node, results in the the mobile node *correcting* the estimates of the particles of the rendezvous point, which in turn *corrects* the estimates of the particles of the mobile node. The various estimation algorithms are described in detail in Section 6.2.

6.1.3 Adaptive Path Planning

We described how a rendezvous between the paths of nodes can be utilized to improve their location estimates. Planning paths is thus the second chicken and egg problem encountered in deployment – better location estimates are needed by nodes to deploy at predetermined regions quickly, while achieving better location estimates may require nodes to take detours (to find rendezvous points) costing time and energy. The planning component of our system seeks to make a suitable trade-off between these aspects of deployment.

6.1.3.1 DrunkWalk

In order to reach the deployment regions, we use the floor plan to produce a graph of connected regions of the environment. Each room or space is a node in the graph and the edges represent the connecting openings between them. It should be noted that the algorithm does not require high quality maps with information of the position of obstacles. Such rough maps are generally available or easy to obtain in most application scenarios,

and provide great benefit in planning the motion of the swarm.

The graph enables us to bias the direction of movement of nodes towards predetermined deployment regions, if the current location of the node in the map can be reasonably determined. However, due to noisy sensors, the location of individual nodes cannot always be estimated correctly making it difficult to consistently plan correct paths. The system attempts to solve this by operating in two modes –

- **Exploration:** In this mode, the controlled-mobile node attempts to seek rendezvous points that can potentially improve the location estimates of the controlled-mobile node. This is executed when the quality of location estimates (determined by the entropy of the tracking particle filter distribution) is low.

- **Navigation:** In this mode, the controlled-mobile node attempts to follow the direction of the bias from the deployment graph using the estimated location from the DrunkWalk algorithm. This is executed when the quality of location estimates is high.

It is easy to see that the performance of the navigation step depends on the outcome of exploration step. However, the exploration step requires extra use of resources that increase the time of deployment. Therefore, the DrunkWalk algorithm seeks to optimize this trade-off by adaptively switching between these two modes.

6.2 DESCRIPTION

This section provides a detailed description of the technical aspects of our proposed system. First, this section describes how the pose of the nodes and the positions of the signatures are estimated over time using a set of particle filters. Separate particle filters

are associated to each controlled-mobile in the team and each RF-signature being localized in space. Therefore, particles estimating the pose of the nodes include the components c_x, c_y, c_ϕ for position and orientation, whereas particles to estimate the location of the signatures include components s_x, s_y for the position. As described in Section 6.1, a base station exchanges information with the nodes (commands and measurements) and maintains a database of known signatures (see Figure 6.1). Due to the limited controlled-mobile on-board computational power, all computations happen in the base.

6.2.1 Particle Filter Basics

A particle filter is a Bayesian estimation method using a finite number of elements (so called *particles*) to represent a non-parametric probability density. Introduced already in the fifties [75], particle filters became enormously popular in robotics in the last two decades [74]. The reasons for this growth are numerous, and we here mention just two. First, there are numerous applications that cannot be tackled with classical estimation methods like (Extended) Kalman Filters. This is the case when the posterior is known to be poorly modeled by a Gaussian, for example when it needs to be multimodal to track various hypotheses. This situation emerges often in robotics and sensor networks applications, so there has been a push towards methods that can overcome these limitations. The other reason is the continuous improvement in computational power that makes now possible to efficiently carry out the computations entailed by a particle filter. Particle filters however do not come without drawbacks. Besides their computational cost, a successful implementation depends on various parameters and distributions whose tuning must often be completed by hand. For example picking the "right" number of particles is often achieved via trial and error.

A particle filter is a specific implementation of the more general recursive Bayes filter under the Markov assumption. Therefore, one needs to assume the availability of two probabilistic models, namely the state evolution model (often called motion model in mobile or robotic applications) and the measurement model. Assuming the unknown state to be estimated at time t is indicated by x_t, the state evolution model provides

$$p(x_t|x_{t-1}, u_t) \tag{6.1}$$

where u_t is the known command given to the system at time t. The measurement model, instead, is given by

$$p(z_t|x_t) \tag{6.2}$$

where z_t is the measurement at time t. Due to the Markov assumption, x_t is conditionally independent from x_k with $k < t - 1$ once x_{t-1} is known. Similarly, given x_t the measurement z_t is conditionally independent from any other variable. Note that one needs not to commmit to specific distributions in Eq. 6.1 and Eq. 6.2, e.g., they do not have to be Gaussian distributions. The generic algorithm to propagate a posterior using a particle filter is given in Algorithm 6.2.1, where we mostly follow the notation presented in [74]. The algorithm starts with a set of M particles $\mathcal{X}$ estimating the posterior of x_{t-1}, i.e., the state x at time $t - 1$. Given the latest command u_t and measurement z_t, it produces a new set of M particles providing an updated posterior estimate for x at time t. The ith particle in $\mathcal{X}_t$, $x_t^{[i]}$, represents the ith possible hypothesis about the state at time t.

The first *for* loop creates a new set of M particles sampling the motion model from the set of existing particles. Each new particle $x_t^{[i]}$ is then associated with a weight $w_t^{[i]}$ scoring how well this new sample matches the most recent measurement z_t. The second for loop implements the so-called *importance resampling*. In this step M particles are sampled

Algorithm 6.2.1 Generic particle filter algorithm

Data: $\mathcal{X}_{t-1}, u_t, z_t$
Result: $\mathcal{X}_t$
$\mathcal{X} \leftarrow \emptyset$ $\mathcal{X}_t \leftarrow \emptyset$ **for** $i \leftarrow 1$ **to** M **do**
  $x_t^{[i]} \leftarrow$ sample $\sim p(x_t|x_{t-1}^{[i]}, u_t)$ $w_t^{[i]} \leftarrow p(z_t|x_t^{[i]})$ $\mathcal{X} \leftarrow \mathcal{X} \cup \{< x_t^{[i]}, w_t^{[i]} >\}$
end
for $i \leftarrow 1$ **to** M **do**
  draw j with probability $\alpha\, w_t^{[j]}$ $\mathcal{X}_t \leftarrow \mathcal{X}_t \cup \{x_t^{[j]}\}$
end

from the intermediate set $\mathcal{X}$, each with a probability proportional to its weight. In this way, particles with a higher weight, i.e., those that better match the measurement z_t, have a higher probability of being selected. The important part in resampling is that it is done *with replacement*, i.e., particles with high weight can be selected multiple times and then appear more than once in the new set of particles $\mathcal{X}_t$.

The set of particles provides a discrete approximation for the posterior. In many applications, including ours, it is often necessary to convert this discrete representation back into a continuous probability density. Different methods have been proposed and the reader is referred to the above references for details. We use an approach based on kernel density estimation where each particle is associated with the center of a kernel function that in our case is the Gaussian distribution. The sum of Gaussian distributions[1] is then normalized and leads to a Gaussian Mixture Model (GMM). Further details are given in the remaining part of this section.

Another problem often arising with particle filters is related to particle depletion, i.e., the problem that the diversity in the set of particles may eventually be lost. When this happens the filter may be unable to correct large errors in the state evolution or measurement,

[1] Kernels are not explicitly scaled by the weight of the particles because particles with higher weights will appear multiple times in the particle set. Multiple occurrences of the same particle give then the weighting.

or may converge to a wrong estimate from which it will not recover. To counter this problem it is necessary to ensure diversity in the particles, for example *injecting* new particles in the filter at every iteration. As detailed in the following, our implementation includes this step exploiting the spatial model of the environment given by the graph (see section 6.2.2.3 for details).

6.2.2 Particle Filter for controlled-mobile

In this section we show how the generic particle filter estimator can be specialized to estimate the location of the nodes. To reduce the computational complexity, rather than implementing a centralized particle filter jointly estimating the location of all the nodes, we associate a particle filter to each controlled-mobile. Assuming there are N_M nodes involved in the deployment task, the system then creates and updates N_M particle filters. Each filter is initialized with $M = 100$ particles uniformly distributed in the area associated with the graph vertex corresponding to the deployment site. All computations take place on the base station.

6.2.2.1 Prediction from Motion Models

For the prediction step it is necessary to use a generative law to implement the particle creation in line 2 of Algorithm 6.2.1. To this end, we use equations similar to the ones given in [72]. Let the command at time t be $u_t = (v_t, \omega_t)$, where v_t is the translational velocity and ω_t is the rotational velocity. Note that the control system always generates commands in which only one of the two components is different from 0, i.e., the controlled-mobile either translates or rotates, but does not make both movements at the same time. Then, a new particle is generated as

$$\begin{pmatrix} c_x \\ c_y \\ c_\phi \end{pmatrix}_t^{[i]} = \begin{pmatrix} c_x \\ c_y \\ c_\phi \end{pmatrix}_{t-1}^{[i]} + \delta t \begin{pmatrix} v_t \cos(c_{\phi t-1}^{[i]}) \\ v_t \sin(c_{\phi t-1}^{[i]}) \\ \omega_t \end{pmatrix}$$

where δt is the time interval between two commands, and the correctness of the equation follows form the assumption that only one between v_t and ω_t can be different from 0. Noise is added to the translational and rotational velocities as per the empirically obtained actuation noise models $p(n_v)$ and $p(n_\omega)$ from our test controlled-mobile platform, but can be specified as per the specific sensor or controlled-mobile platform used. Thus, $v_t^{[i]}$ and $\omega_t^{[i]}$ are obtained as:

$$v_t^{[i]} = v_t + n_v^{[i]} \ , \ n_v^{[i]} \text{ is drawn from } p(n_v)$$

$$\omega_t^{[i]} = \omega_t + n_\omega^{[i]} \ , \ n_\omega^{[i]} \text{ is drawn from } p(n_\omega)$$

where v_t and ω_t are the nominal commands. In our simulations, $p(n_v)$ and $p(n_\omega)$ are specified as normal distributions with $\mu = 0$ and σ is expressed as a percentage of the value of v_t or ω_t.

6.2.2.2 Correction from Measurements

The correction step hinges on the weights assigned to the particles (line 2 in Algorithms 6.2.1). Each controlled-mobile is equipped with a sensor returning a measurement for its heading. Moreover, RF-signature rendezvous provide another measurement. These two measurements are asynchronous, in the sense that while the onboard heading sensor can be queried after each command is executed, signature matching occurs only when revisiting

a location associated with a known signature. In the following we therefore separately describe how the two different weights are computed, given that they are generated and used (via resampling) in separate stages.

The heading measurement is straightforward to integrate. According to former experimental measures [76], the nominal heading returned by the sensor is affected by Gaussian noise with a known variance σ^2 ($\sigma = 40$ degrees to be precise). Therefore, for the heading weight we set

$$p(z_t|x_t) = f_{\mathcal{N},\sigma^2}(z_t - c_{\phi t}^{[i]})$$

where $f_{\mathcal{N},\sigma^2}$ is the density probability of a Gaussian with 0 mean and variance σ^2, and the argument $z_t - c_{\phi t}^{[i]}$ is normalized to account for the 2π period.

The process is substantially different for what concerns RF-signature redezvous and in this case rather than computing $p(z_t|x_t)$ we determine $w_t^{[i]}$ through a two steps process. When a signature is measured, the first step is to communicate with the known signatures database to determine whether the signature is new or has been encountered already (either by the same nodes or a different one). If the database determines the signature is new, the controlled-mobile does not perform the second step and does not compute weights (however, the signature is stored in the database and a new particle filter is created; see section 6.2.3 for details.) If on the contrary the database determines that a signature rendezvous is taking place, the second step starts. First, on the database side, the particle filter estimating the position of the signature being revisited is updated (see section 6.2.3 for details.) After the RF-signature particle filter has been updated, each particle in the controlled-mobile particle filter is assigned a weight as follows. A GMM is created starting from the particles in the signature being matched. Such GMM is a bidimensional probability density function

with the following density function:

$$f_{GMM}(x,y) = \frac{1}{M}\sum_{i=1}^{M} f^i_{\mathcal{N},\Sigma}(x,y)$$

where $f^i_{\mathcal{N},\Sigma}$ is a bidimensional Gaussian distribution with mean $\mu = [s_x^{[i]}\ s_y^{[i]}]^T$ and covariance matrix Σ (a diagonal matrix with value 2 on the main diagonal). Then, each particle is assigned the weight

$$w_t^{[i]} \leftarrow f_{GMM}(c_{xt}^{[i]}, c_{yt}^{[i]}).$$

After all weights have been computed, then resampling can take place as described in Algorithm 6.2.1.

6.2.2.3 Adding Particles Using Connectivity Graph

Due to the unavoidable errors in the estimation process, we implemented an additional step to counter the formerly mentioned particle depletion problem. After the new set $\mathcal{X}_t$ has been created, we determine the graph node with the highest number of particles. Let v_d be this node, and let N be the set of neighbor nodes according to the given graph. Then, the 25 particles with the lowest weight are discarded and replaced by an equal number of particles generated using a random distribution over the space associated with the nodes in N. The rationale behind this step is to generate particles to recover errors due to the erroneous determination that a transition from a room to the next effectively took place.

6.2.3 Particle Filter for Rendezvous Points

We now describe how the spatial location of the signatures can be estimated using a set of particle filters. As for the nodes case, we do not compute a centralized estimation, but we rather associate a filter with each signature to be tracked. This estimation process has

two main differences with the pose estimation for the nodes. First, the number of signature locations to be estimated is not known upfront, so new filters need to be created on-the-fly when a new signature to be localized is identified. Second, signatures do not move. Therefore the estimation process does not include a prediction step but only a correction step. As for the nodes, each filter includes 100 particles.

6.2.3.1 Initialization from controlled-mobile Particles

As described in the previous section, a new signature is generated when the known signatures database receives a query from one of the nodes with an RF-signature that cannot be matched to any of the formerly discovered ones. In this case, a new entry in the database is created and a new particles filter is instantiated. The initial set of particles for this new filter is copied from the particles of the vehicle that discovered the feature, obviously discarding the component related to heading because it is irrelevant for the signature estimation process.

6.2.3.2 Correction from controlled-mobile Particle Filter

Correction happens when a controlled-mobile queries the signature database with a signature that can be matched with one of the entries already discovered. In this case Algorithm 6.2.1 is executed for the signature filter, with the exception of line 2, because no prediction takes place. The weight for $w_t^{[i]}$ for the ith particle is computed as follows. First, the pose (discarding the orientation) of the controlled-mobile that generated the rendezvous is determined taking the average of its particles. Note that this average is implicitly weighted, because through the resampling process particles with higher weight will be included more often in the particle set (see line 5 in Algorithm 6.2.1), so they will be counted multiple times when computing the average. Let $\overline{x}$ be the computed average pose of the

controlled-mobile generating the match, let $s_{t-1}^{[i]}$ be the pose of the ith particle in the signature particle filter at time $t-1$, and let $d_i = ||\overline{x} - s_{t-1}^{[i]}||_2$ be the Euclidean distance between the expected pose of the controlled-mobile and the particle. The weight of each particle at time t is then defined as

$$w_t^{[i]} = F_{d,\delta}(d+K) - F_{d,\delta}(d-K)$$

where $F_{d,\delta}$ is the cumulative density function of a Gaussian distribution with mean d and variance δ. This formula is based on our experimental testbed showing that revisits are correctly detected when the displacement between the original and the new position is within K meters. The specific values for δ and K depend on the number of anchors and are further described in Section 6.3.3.2. Once weights have been computed, correction for the estimate of the signature particle filter can then take place through resampling, as formerly described in Algorithm 6.2.1.

6.2.4 DrunkWalk Planning

In this section, we describe how the system plans the paths of controlled-mobile nodes in order to deploy quickly with location estimates of varying quality. Figure 6.3 shows a flowchart of the planning algorithm.

6.2.4.1 Connectivity Graph from Map

The system uses the map of the environment to extract a connectivity graph. The rooms or empty spaces in the map make up the nodes of the graph, while the doors or other connecting openings represent the edges of the graph. The nodes contain the locations within the rooms while the edges encapsulate the direction nodes must take to move to subsequent rooms.

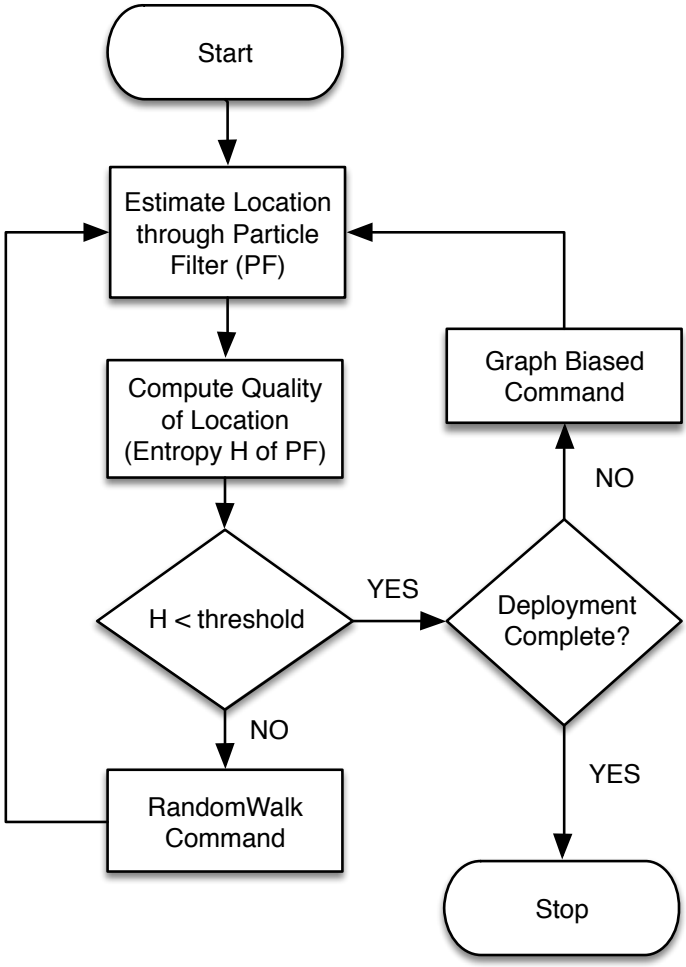


Figure 6.3: The figure shows a flowchart of the DrunkWalk planning algorithm. The planner adaptively changes between random walk and graph biased movement based on the entropy of particle filters tracking respective controlled-mobile nodes.

The graph makes very little assumptions about the quality of the map, but provides a way to bias the motion of controlled-mobile nodes towards designated deployed locations.

6.2.4.2 *Entropy as Quality of Location Estimates*

The entropy of a random variable x can be defined as the expected information that the value of x carries. In the discrete case, it is given by

$$H(x) = E[-\log_2 p(x)]$$

which represents the number of bits required to encode using an optimal encoding, assuming that $p(x)$ is the probability of observing x. The entropy can therefore be used as an indication of the uncertainty of the estimate of a particle filter. The lower the entropy the better is the certainty of the location estimate, whereas, a higher value of entropy indicates a poor estimate of location. For the particle filter, we make a discrete approximation [74] of the entropy of the weights at time t, given by

$$H_t = -\sum_{i=1}^{M} w_t^{[i]} \log_2 w_t^{[i]}.$$

6.2.4.3 *Exploration*

When the entropy of the particle filter is high (¿ threshold T_H), the system seeks to primarily improve the estimates of location. The intuition here is that with an incorrect estimation of current location, using the bias from the graph is likely to be incorrect. This also results in cases where the controlled-mobile may get stuck and perform even more poorly than purely random deployment strategy.

With this in mind, the planner employs a random walk strategy to direct the motion of controlled-mobile nodes. With random walk, the likelihood of nodes discovering rendezvous points increases and so does the likelihood of improving their location estimates. This is referred to as the exploration step.

6.2.4.4 Navigation

Correspondingly, when the uncertainty of location estimates is low ($H_t < T_H$), the planner commands the nodes to follow the bias indicated by the connectivity graph. With a more accurate location, there is an increased likelihood that nodes following the bias reach the intended destination.

A key point in choosing the directional bias from the graph is that it is sampled based on the distribution of particles in the node's particle filter. For example, consider a node with 20 particles indicating its position as room 1 and therefore requiring the node to go north-west to exit the room, while 80 particles indicate the node is in room 2 and must move south. In this case, the planner samples the movement direction according to the distribution of particles over the nodes of the graph, i.e. the node has a 20% chance of being commanded to move north-west and a 80% chance of receiving a south command.

6.2.4.5 Collision Recovery Strategy

controlled-mobile platforms have very limited sensing capability and often do not employ sophisticated obstacle detection sensors. Proposed controlled-mobile platforms [54] rely on their low weight and often use collisions themselves to discover obstacles. However, a strategy is needed in dealing with collision so as to prevent controlled-mobile nodes from being stuck and enable them to back off from corners and crevices and seek out openings. This is especially useful when location estimates are inaccurate. The planner employs

a random exponential back-off strategy, where nodes move in randomly chosen direction (uniformly from a discrete number of directions) for a time duration that increases exponentially with the number of recent collisions. This is implemented by keeping a counter for collisions in a certain time-window. The counter is decremented with time if no new collisions are encountered.

6.3 EVALUATION

In this section we evaluate the performance of our system in deploying in multi-room operating environments through realistic large-scale simulations and experiments on an controlled-mobile testbed. The evaluation focuses on the following aspects –

- Characterizing the performance of the system in terms of time to complete deployment and average accuracy of deployment in comparison to existing deployment approaches.
- The robustness of the system with changing parameters such as sensor noise, radio fingerprint accuracy, and structure of deployment space.
- Validation of the assumptions of the simulation experiments through real controlled-mobile testbed experiments.

6.3.1 Simulation Environment

We extend a controlled-mobile simulation environment [62] for the SensorFly controlled-mobile indoor sensor swarm to evaluate our deployment algorithms at scale in multiple realistic scenarios. The simulator supports inclusion of realistic physical arenas, sensors with configurable sensor noise models, controlled-mobile nodes with mobility models, indoor

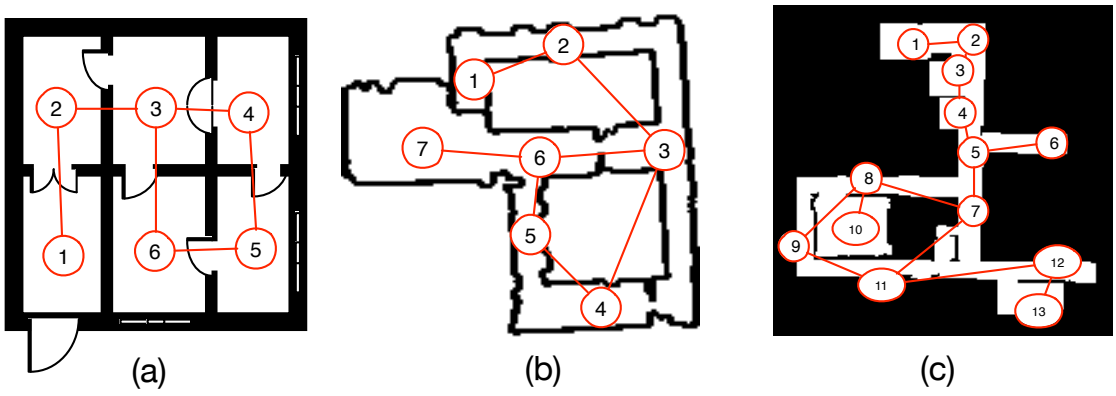


Figure 6.4: The figure shows the floor plans used for simulation with the extracted connectivity graph. (a) A $50m \times 50m$ 6-room map representing a house floor plan with rooms connected through doorways. (b) The $50m \times 50m$ 7-room map of a office environment obtained by a mapping robot. (c) 13-room $200m \times 200m$ map of the floor of a university building to evaluate at a large scale.

radio propagation models, and environment sensing. The simulator allows users to program the logic for actuation of node's and implement control and planning algorithms. Finally, the simulator can be interfaced with actual hardware and run hardware-in-loop simulations in a controlled-mobile testbed.

For our evaluations we configure the simulator as follows:

- **Arena** – We assume a multi-room indoor scenario, where nodes are required to autonomously deploy over all rooms. We use maps of multiple sizes and structure, including two obtained by a mobile robot equipped with laser scanners, to evaluate our system. Figure 6.4, shows the 3 distinct maps. Figure 6.4a represents a typical indoor apartment scenario where such systems may be deployed in search and rescue applications. This map is used for extensive simulations, parameter selection and analysis of the algorithm. In addition, we use maps of increasing complexity

, Figure 6.4b is a 7-room map of an office building built by a robot and therefore presenting challenges due to uneven walls and narrow navigable paths. Finally, to test the algorithm at a larger scale we use a 13-room $200m \times 200m$ floor plan of an university building (Figure 6.4c). [2] Note that given a map produced by a SLAM algorithm, graphs like those displayed in the figures can be automatically extracted and do not have to be produced by hand [77].

- **Node Sensors** – The sensor nodes in the simulation are modelled after the SensorFly [54] controlled-mobile platform, which is also used in our testbed experiment. Each node has a 802.15.4 radio and dead-reckoning sensors – gyroscope, an optical flow velocity sensor, an ultrasonic altitude measurement sensor.

- **Node Mobility** – The controlled-mobile nodes can turn by a commanded angle and move for a commanded time and velocity. We set the velocity to 0.25 m/s in agreement with the testbed controlled-mobile parameters. The velocity of course varies in accordance with the noisiness of the optical flow sensor, that provides feedback to each node's control algorithm.

- **Simulation Time-steps** – The simulation time-step is chosen as $1sec$ that enables nodes to cover a distance of $0.25m$ to $0.45m$ in one simulation tick.

- **Radio** – The simulation supports estimating received signal strength (RSS) and round-trip time-of-flight (RToF) measurements between two nodes. The RSS is computed using shadowing with a path loss exponent of 3, which is an estimate for an indoor single-floor scenario [37]. The RToF is modeled as a function of distance with added Gaussian noise of standard deviation $3m$, from experimental measurements in [54].

[2] This map is obtained from the sdr40 dataset available on `http://radish.sourceforge.net`.

6.3.2 Simulation Results

First, we evaluate our system by comparing the percentage of deployment completed in a 6-room scenario as a function of time.

We compare our DrunkWalk algorithm to two other deployment strategies that do not require any location infrastructure. They are –

- **Random Walk:** This is a popular strategy for simple robots with few sensors and has been extensively researched for use in scenarios where no location infrastructure exists [60]. We use this as a baseline for comparison.

- **Dead-Reckoning with Map Bias:** Dead-reckoning is another infrastructure-free technique used to estimate a node's location in unknown environments [78]. The method uses measurements from motion sensors – optical flow and gyroscope to estimate the change in position of the node. Having an estimate of location we use the map to bias the direction of the node's movement similar to DrunkWalk.

All experiments were performed 25 times and the error bars show the standard deviation of the measured values.

Figure 6.5 shows the percentage of deployment completed as a function of time using DrunkWalk, Dead-Reckoning with map bias, and Random Walk. The simulation uses 10 nodes for each strategy in the 6-room map (Figure 6.4a). The nodes are introduced into room 1 and the objective is to deploy at least one node in each of the 6-rooms of the map. We run the simulation for a time period of 1000 seconds ($\tilde{1}5$ minutes) corresponding to the typical battery life of current generation controlled-mobile nodes. The dead-reckoning sensor noise models are set $\sigma = 20\%$ and radio fingerprint accuracy is $1m$.

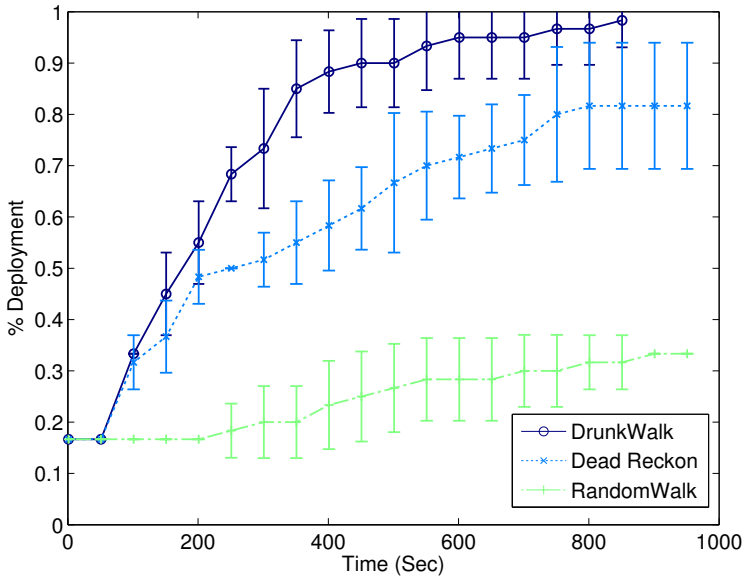


Figure 6.5: The figure shows the % deployment completed over time (6-room map) for 10 nodes using 3 different strategies - DrunkWalk, dead-reckoning based biased walk, and random walk. DrunkWalk strategy achieves complete coverage in lesser time. The error bars show the standard deviation over 25 runs.

DrunkWalk is significantly faster at deployment and also manages to complete the deployment before other strategies. Dead-reckoning achieved 80% deployment at the end of the node lifetime. while Random Walk managed 30%. The poor performance of Random Walk is expected in multi-room scenarios with small openings between rooms since the probability of robots making it to subsequent rooms is low. The better planning and location accuracy of DrunkWalk enables it to perform better than dead-reckoning, especially later in the deployment when dead-reckoning error becomes extremely large.

Figure 6.7 shows the ratio of dead-reckoning error to DrunkWalk location estimation error over time, in the 3 different floor plan (Figure 6.4). The plot indicates that DrunkWalk provides a significant improvement in estimation error over dead-reckoning, irrespective of the size and structure of the environment.

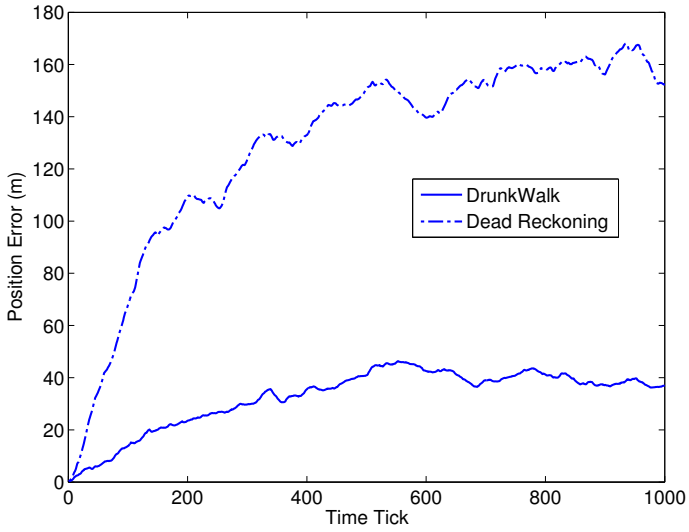


Figure 6.6: The figure shows the location error as a function of time using DrunkWalk estimation and dead reckoning alone, using 10 nodes in the 6-room map scenario.

6.3.2.1 Number of Nodes

Figure 6.8 shows the percentage deployment over time for varying number of mobile controlled-mobile nodes. The larger number of nodes complete deployment faster. This can be attributed to both the slight improvement in accuracy due to a higher chance of rendezvous between node paths and the larger number of nodes exploring the space. For the experiment, each dead-reckoning sensor noise is set to $\sigma = 20\%$ and radio fingerprint accuracy is $1m$.

Another interesting metric is the ratio of dead-reckoning error to DrunkWalk estimates as the number of nodes are varied. Figure 6.9 shows the error ratio as a function of time. We observe that DrunkWalk provides more than 3X improvement in location accuracy even when the number of nodes increase.

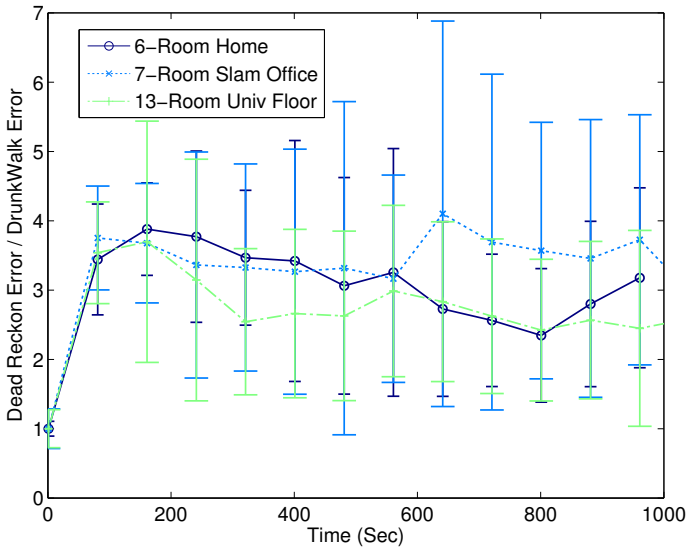


Figure 6.7: The figure shows the ratio of dead-reckoning error to DrunkWalk location estimation error over time in the 3 different floor plans. The plot indicates that DrunkWalk provides a significant improvement over dead-reckoning irrespective of the size and structure of the floor plan.

6.3.2.2 Dead-reckoning Sensor Noise

The noise in motion measurements due to the dead-reckoning sensors is an important parameter in determining the eventual performance of the algorithm. Different controlled-mobile platforms and operating environments might have different degree of noise in their motion measurements, making it useful to analyze the performance of the algorithm for varying degree of sensor noise. For our simulations, in agreement with empirical measurements on our test controlled-mobile platform, we model noise as a normal distribution with a standard deviation proportional to the sensor measurement. For example, for the optical flow velocity sensor, a 5% noise corresponds to a normal distribution with 0 mean and standard deviation of 5% of the measured velocity value.

In our simulations, we keep the value of noise for both the velocity and turn sensors equal. The resultant noise in dead-reckoning location can be computed as per the motion

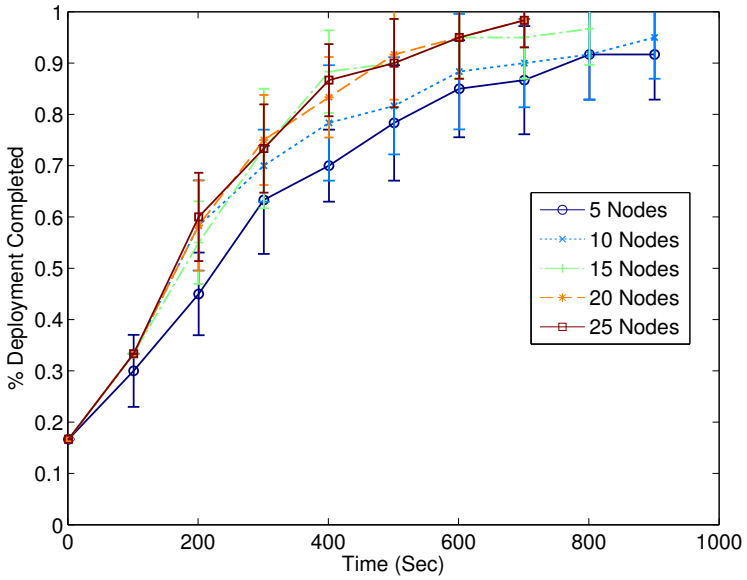


Figure 6.8: The figure shows the % deployment completed over time with varying number of nodes using DrunkWalk. The deployment time reduces as the number of nodes is increased.

update equation.

The sensor noise affects dead-reckoning estimates as well as DrunkWalk estimates because it uses the dead-reckoning estimates as a baseline. Figure 6.10 shows the ratio of error in dead-reckoning to DrunkWalk for 10 nodes in the 6-room map scenario at various sensor noise levels. We observe that for very low noise levels (5%), dead reckoning performs well and the relative improvement is low. However, as the noise level increases, the advantage of DrunkWalk becomes apparent. For more realistic noise values ($15\% - 20\%$), DrunkWalk provides $3\times$ improvement.

6.3.2.3 Radio Fingerprint Accuracy

DrunkWalk depends on being able to identify rendezvous on node paths. This it accomplishes using radio fingerprints from stationary controlled-mobile nodes that it deploys at the beginning. When a radio fingerprint collected by a node at a certain location is sim-

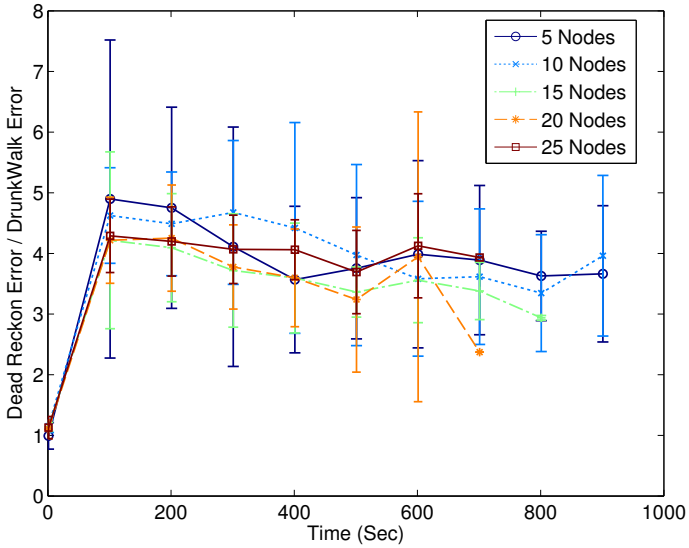


Figure 6.9: The figure shows the ratio of dead-reckoning error to DrunkWalk location estimation error over time with varying number of nodes. DrunkWalk provides ¿ 3X improvement in location accuracy over dead-reckoning alone.

ilar to a fingerprint in the database, the system classifies it as rendezvous and performs the correction of location estimates. Thus, the performance of DrunkWalk depends on the accuracy of the fingerprints, i.e. the distance within which two radio fingerprints can be reliably classified as being in the same location.

The fingerprint accuracy depends on the nature of radio measurements, number of anchors, and the structure of the indoor environment, as has been shown in existing wireless fingerprinting based localization approaches. Therefore, we seek to evaluate how DrunkWalk performs in a range of accuracies that maybe encountered in real-life deployments with different radio measurement capabilities. For example, a Wi-Fi RSS based approach with few anchors might have an accuracy of 6m [79], whereas a system with more accurate round-trip time-of-flight (RToF) measurements and numerous anchor nodes might have under $1m$ accuracy [80].

Figure 6.11 shows the ratio of dead-reckoning error to DrunkWalk error for 10 nodes for

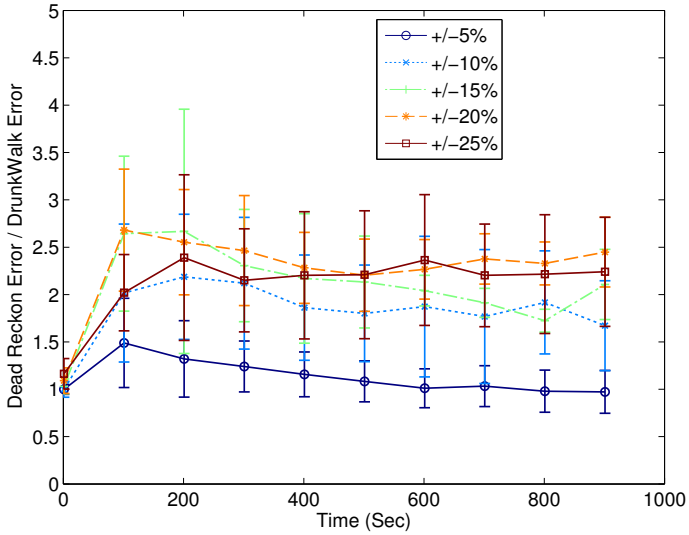


Figure 6.10: The figure shows the ratio of dead-reckoning error to DrunkWalk location estimation error over time with varying sensor noise.The noise per sensor is modeled as a normal distribution with varying standard deviation. The plot shows that DrunkWalk estimation is able to correct the dead-reckoning error especially as the sensor noise increases and raw dead-reckoning results deteriorate.

different fingerprint accuracy values. We observe that DrunkWalk offers an improvement of over $2\times$ over dead-reckoning even for low accuracy of $5m - 7m$. In case of controlled-mobile deployments, where the low-cost of nodes makes it possible to deploy a relatively large number of beacon nodes and attain better fingerprint accuracies$<$ $3m$, DrunkWalk provides a much larger reduction in error.

The initial increase in accuracy can be attributed the fact that controlled-mobile nodes start close to each other with a very accurate initial estimate of location. Therefore, the number of rendezvous points close to starting point is higher and DrunkWalk performance is better. As nodes move further away, the number of rendezvous reduce and so does the performance of DrunkWalk. However, the planning algorithm later succeeds in stabilizing the performance by ensuring nodes seek out rendezvous points through the exploration strategy.

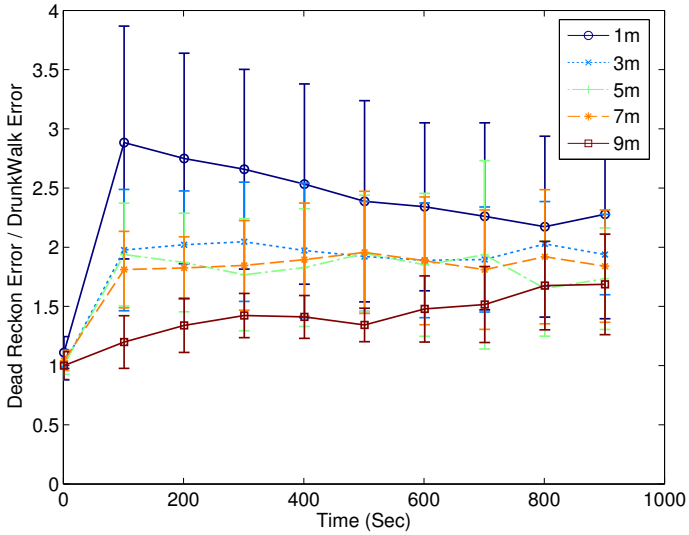


Figure 6.11: The figure shows the ratio of ratio of dead-reckoning error to DrunkWalk location estimation error over time with varying accuracy of radio fingerprints.

6.3.3 controlled-mobile Swarm Testbed

We validate our system in a controlled-mobile swarm testbed experiment. We implement our algorithm on a base station and the SensorFly [54] controlled-mobile testbed.

The SensorFly platform used in our test has a 8-bit 16Mhz AVR AtMega128rfa1 microcontroller, motion sensors – 3-axis accelerometer, 3-axis gyroscope, optical flow velocity sensor, ultrasonic altitude sensor and a XBee 802.15.4 radio [15]. The entire platform is under $30g$ in weight and has a flight time of 6-8 minutes. The SensorFly nodes are capable of translational and rotation motion from on-board PID control algorithms utilizing feedback from the gyroscope and optical-flow sensors.

The testbed consists of a $5m \times 8m$ arena with a grid dividing it into cells of $1m \times 1m$. A ceiling-mounted camera feeding into a color blob detection algorithm is used to track the *ground-truth* location of all controlled-mobile nodes on the grid. The blades of the SensorFly nodes are affixed with red-tape so as to be easily detectable by the vision-based

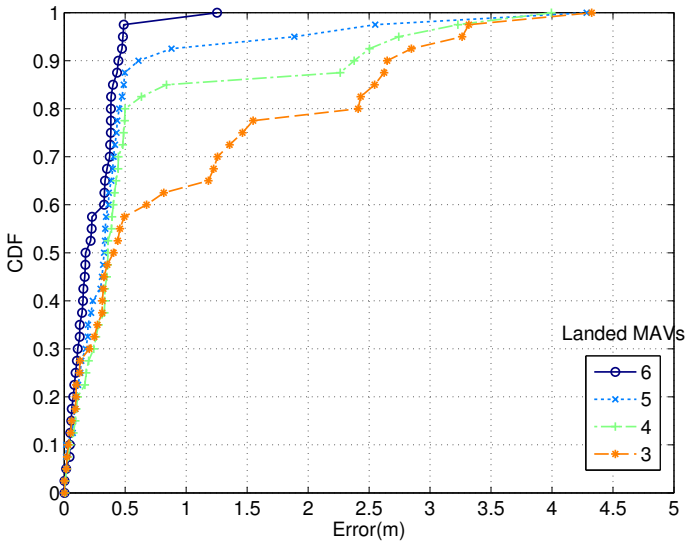


Figure 6.12: The figure shows the fingerprint accuracy for the testbed experiment in comparison to the ground-truth measurements with varying number of stationary controlled-mobile beacons. With 6 nodes the $90\%ile$ fingerprint accuracy is under $0.5m$ for the testbed arena over the experiment duration of 6-minutes.

ground-truth tracking system.

6.3.3.1 Testbed Results

Our testbed experiment employs a swarm of 9 controlled-mobile nodes. 5 nodes are allowed to disperse and deploy as beacons at initialization, while 4 nodes are mobile to seek out the deployment location. The nodes are introduced at grid location [1,1] and follow a random path in the arena. The objective of the testbed experiment is to validate the simulation of the DrunkWalk system under real radio measurements and dead-reckoning sensor noise. Currently, the testbed experiment cannot fully evaluate the planning due to unavailability of multi-room controlled-mobile testbeds with accurate ground-truth measurements.

The mobile nodes obtain RSS measurements from landed controlled-mobile nodes and

compute dead-reckoning estimates on-board (rotation obtained from integrating gyroscope and velocity from optical flow sensor). This data is relayed to the base periodically at an interval of 2 seconds. The location estimation error is computed using the ground-truth vision-based tracking system of the testbed. The testbed experiment runs for 6-minutes of node flying time which enables us to execute multiple runs on a single charge.

6.3.3.2 RSS Fingerprint Accuracy

As the mobile nodes move through the arena they create an on-line database of radio signatures from the deployed beacons. One of the key points to validate in the real experiment is the accuracy of radio fingerprints that can be attained in a realistic setting with radio multi-path effects and noise. Therefore, using the ground-truth location, we compute the error (distance) for every signature that is matched against a previously collected signature, over the duration of the experiment. In order to analyze the effect of the number of beacons (landed nodes) on fingerprint accuracy, we post-process the data to omit readings from a subset of beacons, and replay the matching process.

Figure 6.12 shows the cumulative distribution of the error for fingerprint matches. We observe that a relatively high accuracy can be obtained with 6 nodes ($90\%ile$ under $0.5m$) even with noisy RSS measurements. This maybe attributed to the fact that the initial signature discovery and subsequent matching occur over a short duration of time (in the order of minutes), thereby avoiding the effects of longer-term RSS variations that are prevalent with traditional wireless fingerprinting based systems.

This experiment can also aid us in setting parameters of the DrunkWalk algorithm, specifically, the value K (Section 6.2.3.2) that represents the distance under which revisits can be detected correctly by matching signatures.

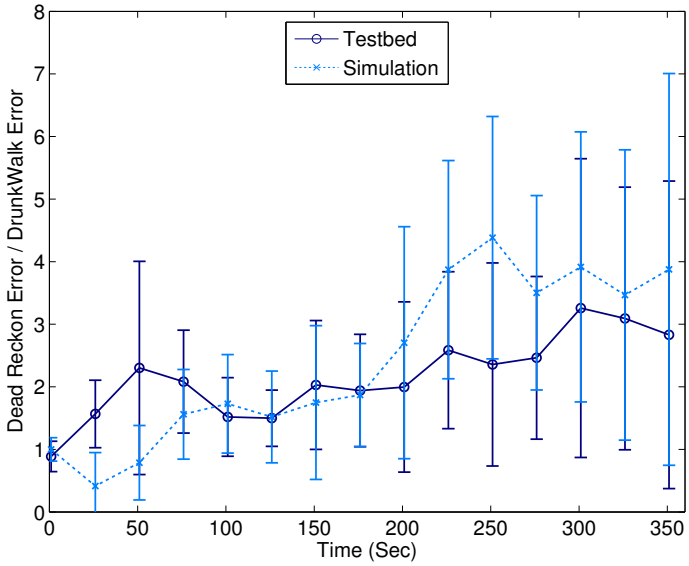


Figure 6.13: The figure shows the ratio of Dead-Reckoning error to DrunkWalk location estimation error for 4 mobile node's, 5 stationary node's, over a duration of 6 minutes in our controlled-mobile testbed. DrunkWalk estimates were up to 3X more accurate than dead-reckoning. The result follows closely to a simulation with similar number of nodes, arena configuration and a radio fingerprint accuracy of $1m$, validating our assumptions.

6.3.3.3 DrunkWalk Performance

We evaluate the performance of DrunkWalk in terms of the ratio of dead-reckoning error to the DrunkWalk location estimation error, similar to the metric used in our simulation experiments.

Figure 6.13 shows the ratio of dead-reckoning to DrunkWalk location estimation error as a function of time for the real testbed experiment and for a simulation under similar arena, node, and noise configurations ($5m \times 8m$, 4 mobile nodes, 5 stationary nodes, $1m$ fingerprint accuracy, and $\sigma = 20\%$ dead-reckoning sensor noise). We observe up to $3\times$ reduction in error with DrunkWalk. The results from the simulation and real experiment closely follow each other as time progresses, validating the simulation setup and methodology.

6.4 RELATED WORK

Howard et al. [81] present techniques for mobile sensor network deployment in an unknown environment. Their approach constructs fields such that each node is repelled by both obstacles and by other nodes, enabling the network to spread itself throughout the environment. Similarly, Batalin et al. [68] present a deployment algorithm for robot teams without access to maps or location. The robots are assumed to be equipped with vision sensors and range finders and select a direction away from all their immediate sensed neighbors and move in that direction. The algorithm does not require communication between nodes but also does not allow nodes to be deployed at designated locations. The domain experts have no control over the emergent deployment locations of the nodes.

The problem addressed in this chapter can be seen as an instance of the Simultaneous Localization And Mapping (SLAM) problem that has been extensively studied in robotics [74]. In fact, in the system we described multiple nodes try to localize themselves while at the same time trying to acquire a representation of the spatial distribution of the radio signatures. In recent years there has been copious research in SLAM using either methods based on Kalman filters [82] or particle filters [83]. Both approaches, however, have been mostly applied to solve instance of the SLAM problem where mobile agents are equipped with sensors returning distances (e.g., laser range finders, or sonars) or cameras (either monocular or stereo). Therefore, the ultimate objective of these solutions to the SLAM problem was to map physical entities located in the environment, like walls, obstacles, etc. Methods based on Kalman filters are not applicable for the scenario we consider because we are dealing with multimodal, nonparametric probability distributions. Therefore, we opt for a solution based on particle filters.

Approaches based on explicit perception and processing of radio signals have been

mostly aimed at implementing localization systems with the underlying assumption that radio signals were preliminarily collected off line to build so-called *map signals* [79, 84]. A recent paper by Twigg et al. [85] discusses a system where a robot autonomously discovers the area within which connectivity with an assigned WiFi base station is ensured. Their solution, however, solves only the mapping side of the problem because the robot is equipped with a laser range finder solving the localization problem. In other words, RSS readings are mapped to the physical space exploiting the availability of a different sensor providing reliable localization.

For people carrying mobile devices, SLAM-like approaches have recently been proposed that fuse WiFi-based RSS and motion sensor data to simultaneously build a sensor map of the environment and locate the user within this map. e.g. radio fingerprint maps [86, 87], or organic landmark maps [88]. These approaches focus on location estimation part for an orthogonal problem, where the motion of users cannot be controlled and hence does not involve motion planning or deployment. Purohit et al. [72] present a system for infrastructure-free single room *sweep* coverage with controlled-mobile sensor swarms. Their approach however does not support deploying nodes to pre-assigned destinations.

To the best of our knowledge, this chapter presents the first attempt to solve a SLAM problem using a swarm of nodes that combines location estimation and planning to improve the speed and accuracy of deployment.

6.5 CONCLUSION

This chapter presents a system for collaborative deployment of resource-constrained controlled-mobile sensing swarms to quickly and efficiently deploy at preassigned locations. The system uses collaboration between nodes of the swarm to overcome the sensing

and computational limitations of controlled-mobile nodes, and challenging operating environments. We comprehensively evaluate the system through large-scale simulations and real controlled-mobile testbed experiments showing that DrunkWalk performs up to $3\times$ better than existing deployment strategies.

CHAPTER 7

NAVIGATION

Fine-grained indoor navigation can enable controlled-mobile nodes to achieve tracking and guiding capabilities in indoor environments. For example, using indoor navigation system, a sensor node can guide firefighters to a victim. The application of such navigation systems depends on their accuracy in face of the constraints of controlled-mobile systems.

In this chapter, we present SugarTrail, a system for indoor navigation for controlled-mobile sensors in structured environments without existing maps or prior surveys. By leveraging the structured movement pathways in indoor environments such as retail stores, homes or offices, the system enables controlled-mobile sensor nodes to provide high accuracy navigation assistance to users.

Similar to SugarMap (Chapter 5) and DrunkWalk (Chapter 6), the system deploys low-cost sensor nodes equipped with off-the-shelf radios without location measurement. To acquire knowledge of the physical environment *without* a map, the system collects radio and compass signatures to continually record paths traversed by exploring nodes. Using this information, paths are automatically aggregated into path-clusters and a navigable *Virtual Roadmap* (VRM) of the indoor environment is built by the system. Using this roadmap,

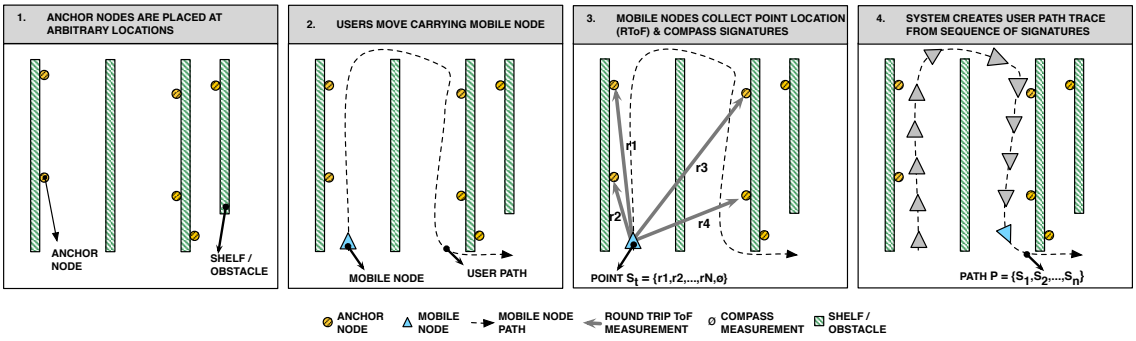


Figure 7.1: The figure shows the process of recording node path traces in a structured retail store indoor environment. 1) Anchor nodes are distributed in the environment without the need to measure their location. 2) Mobile nodes move through the environment and 3) Mobile nodes periodically collect magnetometer measurements and radio measurements from anchor nodes that are in range. 4) The sequence of signatures define a path and are transmitted to a base station.

the system guides users to a point of interest in the explored map.

To fully evaluate our system in a realistic usage scenarios, we developed a SensorFy node based system and deployed it, on campus, and in the New Wing supermarket in Santa Clara, California (over a 30-day period) during business hours. Based on this data, results show that SugarTrail can accurately navigate users throughout the entire supermarket, with a success rate of more than 85% and average accuracy of 0.7m.

7.1 SYSTEM OVERVIEW

The SugarTrail system provides accurate navigation in indoor environments without access to updated maps and prior RF-surveys. The system leverages the structured movement pathways in retail environments to enhance accuracy.

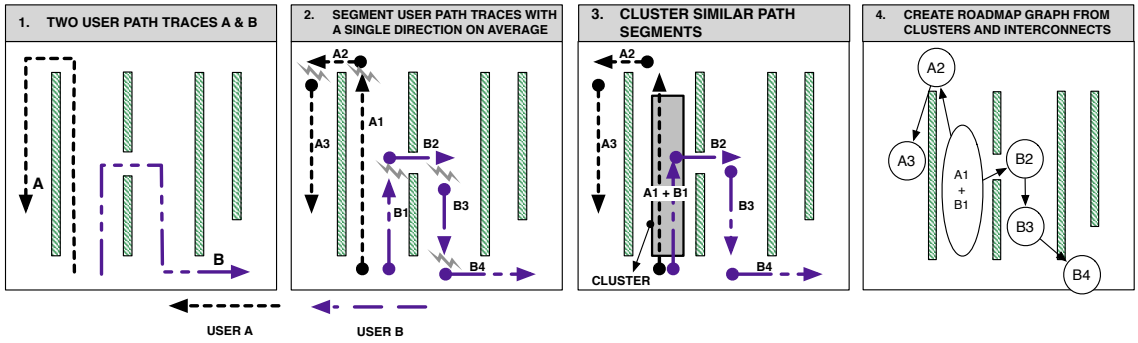


Figure 7.2: An illustration of the SugarTrail algorithm using path traces from nodes A and B to build the VRM graph. 2) The algorithm breaks a path into path-segments, 3) clusters similar path segments, and 4) uses the aggregated information to build the VRM with path-clusters as vertices and their connections as edges. For visual clarity, the nodes' paths are drawn as straight lines.

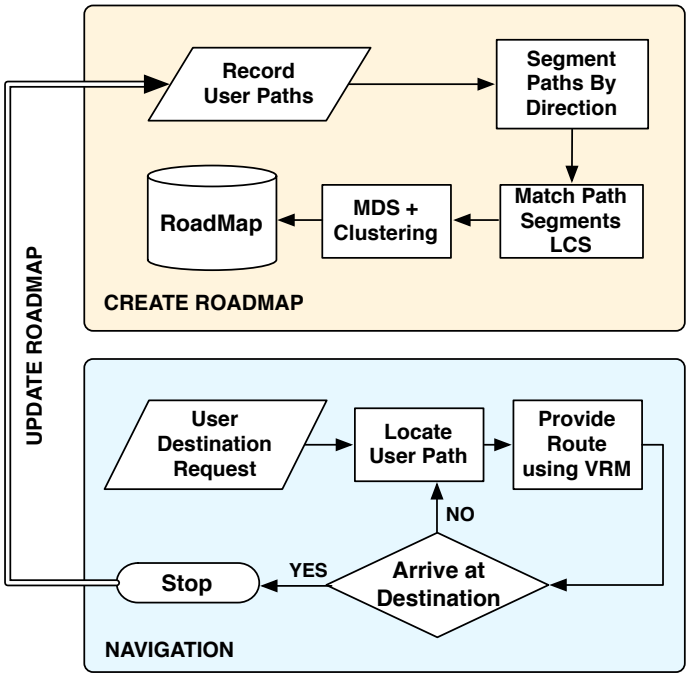


Figure 7.3: The figure shows a flowchart system operation. The system has two main modes of operation. Top) Creating/updating of the "Virtual Roadmap". Bottom) Navigation mode, where the system guides users to their desired destination.

7.1.1 Structure of Indoor Environments

Many indoor environments such as grocery stores and supermarkets have structured movement pathways [89]. Stores typically consist of aisles with products placed in shelves along them. In addition, products may be placed on islands along the broader walkway along the perimeter called the *racetrack*. The product placement limits the trajectory of node movement to approximately unidirectional paths within aisles or along he perimeter. The SugarTrail system leverages this structure to automatically learn the graph of pliable pathways (VRM) in a store from node movement traces (measured using radio and magnetic signatures). Of course, individual nodes may not move along the aisles, may turn around, or linger around. The SugarTrail system uses magnetic signatures to discard such traces that do not contribute to extracting the VRM.

7.1.2 System Operation

The SugarTrail system operation can be described in four steps – 1) deployment of anchors, 2) recording of node path traces, 3) automatic creation of a Virtual Road Map (VRM), and finally 4) navigation of users. This section gives an overview of our approach.

1. Deployment of sensors: The system involves deploying, without location or orientation measurement, a number of *stationary* low-cost wireless nodes equipped with off-the-shelf 802.15.4a radios in the environment. A wireless node with radio and compass moves and explores the environment. Since, no measurements are needed in placing anchor nodes, the deployment requires no prior setup. For example, in a retail store, the anchors can be deployed on merchandise racks in sufficient density for mobile nodes to hear 4 or 5 stationary nodes at most locations.

2. Recording of node path: The mobile nodes record their path as a sequence of mag-

netometer readings and radio round-trip time-of-flight measurements from the stationary nodes. The goal of the radio measurements is to provide approximate signatures for each location along the node's path, while the magnetometer measurements provide a unique signature for the nodes direction at each location on the path. Figure 7.1 shows this process of collecting path traces in a retail store environment.

3. Creating the VRM: Next, the system learns a "Virtual Roadmap" of the environment by comparing and combining the recorded node paths into path-clusters, without any supervised input of the physical location of the paths. This VRM is a representation of the usable pathways in the deployment region and their interconnections with each other.

SugarTrail aggregates paths from multiple nodes into uni-directional path-clusters to quickly build the VRM with maximum coverage of the area. To achieve this, SugarTrail analyzes the recorded paths and divides them into uni-directional segments that correspond to a distinct physical pathway, such as a hallway or a supermarket aisle, based on the modal direction of the nodes. This is based on the observation that nodes in a pathway tend to travel in a few common directions on the whole.

Figure 7.2 illustrates the SugarTrail algorithm for building a VRM from the path traces of two nodes. On the basis of the segmentation, node paths are represented as a sequence of path segments (corresponding to physical pathways) that connect to each other, such as the segments A1 to A3 and B1 to B3 in Figure 7.2. Subsequently, the SugarTrail system finds similar segments from paths of multiple nodes and clusters them together into path-clusters. The connections between path-clusters are determined from the connectivity of their constituent path segments. Finally, path-clusters and their connections are modeled as the vertices and edges of a single directed graph (VRM). For simplicity, figure 7.2 shows node paths as straight lines, however, the system is designed and evaluated for realistic

human paths

4. Navigation: Finally, the system provides users with directions to their desired destination using the VRM. The system first locates users by matching their current path to the paths in the VRM, and then uses the connectivity information in the VRM to suggest a route. The route is conveyed to the as a sequence of direction prompts based on compass directions that relative to the user's local magnetic field (learned from recorded node path traces).

7.2 ALGORITHM DESIGN

In this section, we present the details of the SugarTrail algorithm including practical real-world considerations. A detailed flowchart of system operation is illustrated in Figure 7.3. As shown in the figure, the system operates in two primary modes, namely, 1) the creation or update of the VRM and 2) navigation using the VRM. Each step in the flowchart of system operation is described in detail in the following subsections.

7.2.1 Recording Node Path Traces

To learn the navigable pathways of the environment without a known physical map, the SugarTrail system records node paths as a sequence of node location and direction signatures. The system employs two novel techniques for recording node paths that enable it to overcome noise inherent in indoor environments, 1) sequence of radio-based location signatures based on round-trip time-of-flight measurements, and 2) magnetometer-based local direction signatures. The two types of signatures are described below.

7.2.1.1 Round-Trip Time-of-Flight Signatures

The mobile SensorFly nodes use round-trip time-of-flight (RToF) measurements to estimate location with respect to stationary anchor nodes. The round-trip time-of-flight method measures the elapsed time between the host node sending a data signal to the remote node, and receiving an acknowledgment from it.

The SugarTrail nodes use physical layer timestamps and hardware-generated acknowledgments, supported by their nano-LOC radios, to compute RToF measurements [63]. However, prior experiments analyzing the distance estimates from RToF measurements in various space configurations, show that the distance-to-measurement correlation is not high for multi-path rich indoor environments 2.3.2.

Therefore, in order to compensate for individual signature variation, SugarTrail uses a set of RToF measurements from multiple anchors, distributed around the environment, to create multi-dimensional signature vectors.

The node obtains RToF signatures while moving. Consequently, the readings from all anchors are not obtained for a single point but over a small section of the path. Each round of measurements lasts for a period of 1 second and the location of the signature can be approximated to a point on the path.

7.2.1.2 Magnetometer Direction Signatures

Recording a node's path involves knowing the sequence of location signatures and the spatial relationship (direction) between them. If a navigation system knows the location of the node in Cartesian coordinates, the nodes direction of movement can be determined directly. However, since SugarTrail navigates nodes without existing maps, the spatial relationship between nodes must be measured. SugartTrail uses magnetometer signatures

to determine the spatial relationship.

In indoor environments, magnetometers can be significantly affected by soft-iron effects due to surrounding metallic structures such as cubicles, appliances, support beams, etc., as well as by hard-iron effects due to power lines [90]. To deal with the indoor magnetic interferences, SugarTrail uses the magnetometer as a relative and local direction signature as opposed to a global orientation measurement. Specifically, the system uses raw magnetometer direction signatures to identify two major features of the node path,

- **Points where the node changes direction** corresponding to a transition between pathways. This is used to detect path junctions where paths intersect. These junctions are later used to divide node path traces into segments that have a single direction.
- **Node path segments that have similar directions** for determining co-located path segments and aligning them in the matching phase.

SugarTrail leverages two types of location-dependent signatures, namely RToF signatures and magnetometer readings. Suppose a node moves along a path P, each single location point p_i will be represented by a N-dimensional RToF signature vector

$$\overrightarrow{R_i} = \{r_1, r_2, \ldots, r_N\}, \tag{7.1}$$

where r_i is the RToF measurement from the ith anchor and N is the total number of anchors in the environment, as well as a direction measurement from the magnetometer ϕ_i, together forming a signature set $\overrightarrow{S_i} = \{\overrightarrow{R_i} : \phi_i\}$.

7.2.2 Segmentation of Paths

After recording node path signatures, the system builds the VRM representing usable pathways and their interconnections with each other. To achieve this, SugarTrail analyzes

recorded paths and divides them into segments that correspond to distinct physical pathways, such as a supermarket aisle.

To isolate path segments belonging to the same physical pathway, we observe that human motion along physical pathways (such as aisles) tends to be limited to a few directions on a macro-scale. For example, people tend to move forward or backward in an aisle. While, on a micro-scale people may move in diverse directions resembling a random waypoint walk, by filtering out the micro-scale motion, the system can extract path segments having a single major direction.

Based on the above intuition, SugarTrail first computes the difference in direction between consecutive points in a path, then detects points at which a significant change in direction occurs. These points are marked as candidate turn points. To reduce false detection of segments caused by micro-scale directional variations, SugarTrail computes the modes m_l and m_r of two sliding windows before and after each candidate turn point along a path trace. The mode of the first window m_l is then compared to that of the second window m_r. When the difference exceeds a direction change threshold ${}_{\Delta}m_{THR}$, the point is marked as a valid turn point. Finally the paths are segmented at each of the valid turn points. The pseudo-code is described in Algorithm Box 7.2.1.

7.2.3 Determining Path Segment Similarity

SugarTrail measures the similarity of pair-wise path segments so that it can group physically proximate path segments into path-clusters. This process comprises of two steps. (1) The first step measures the similarity of pair-wise signatures in two path segments. (2) After acquiring the similarity of all the corresponding signatures in the two path segments, the system uses a modified longest common subsequence (LCS) algorithm to compute a

Algorithm 7.2.1 Path Segmentation

1: k=0, do the following for each path P_i ($i \in \{1, I\}$)
2: **for** each point p_j in path P_i **do**
3: Find difference in direction between consecutive points p_j and p_{j+1}: ${}_{\Delta}\phi_{j,j+1} = \phi_{j+1} - \phi_j$
4: **if** ${}_{\Delta}\phi_{j,j+1} >_{\Delta} \phi_{THR}$ **then**
5: Mark the turn index t_k as $j+1$, then $k = k+1$.
6: **end if**
7: **end for**
8: **for** each turn index t_k ($k \in \{1, K\}$) **do**
9: Find the mode over sliding window before t_k:
$m_l = \text{mode}(\{\phi_{t_k - win\_size:t_k}\})$
10: Find the mode over sliding window after t_k:
$m_r = \text{mode}(\{\phi_{t_k:t_k + win\_size}\})$
11: Find difference ${}_{\Delta}m = \text{abs}(m_l - m_r)$
12: **if** ${}_{\Delta}m >_{\Delta} m_{THR}$ **then**
13: Segment path P_i at point p_{t_k}
14: **end if**
15: **end for**

similarity score between paths.

7.2.3.1 Signature-Wise Distance Metric

Based on the nature of obtained signatures, SugarTrail uses a hybrid metric to compute signature-wise distance consisting of a weighted sum of hamming and euclidean distance between signatures.

First, a mobile node may or may not be in communication range from a given anchor node depending on its location. From a signature point of view, an out-of-range or failed measurement leads to *nulls* in the RToF signature vector $\overrightarrow{R_i}$. To quantitatively measure this connectivity difference of RToF measurements between point p_i and p_j, we derive the Hamming distance

$$H_{i,j} = \frac{\overrightarrow{R_i} \oplus \overrightarrow{R_j}}{N}, \tag{7.2}$$

where the "$\oplus$" operation counts the number of non-null common elements in two vectors.

The Hamming distance computes the similarity of visible anchors from the two location signatures.

Second, if two locations are physically close, the magnitude of RToF measurements from the same anchor node should be similar. Based on our analysis of location signatures, this translates to a small Euclidean distance between the N-dimensional RToF signatures.

The normalized Euclidean distance, given by,

$$d_{i,j} = \frac{\|\overrightarrow{R_i} - \overrightarrow{R_j}\|}{\#\text{ of visible common anchors}} \tag{7.3}$$

measures the distance in RToF measurements from anchors visible in both signature vectors. To compensate for RToF measurement variance, in the implementation of the system we penalize faraway location signatures that yield small Euclidean distances by non-linearizing the Euclidean distance $d_{i,j}$ using a Sigmoidal transformation, i.e.

$$d'_{i,j} = \frac{1}{1 + e^{-d_{i,j}}}. \tag{7.4}$$

Finally, the signature-wise signature distance is computed as a weighted sum of both the Hamming and the Euclidean distances, i.e.

$$D_{i,j} = \frac{d'_{i,j} + \omega H_{i,j}}{1 + \omega}, \tag{7.5}$$

where ω is the weight to balance the Euclidean and the Hamming distance. During the evaluation, we empirically set ω to 0.5, such that the signature distance has a high correlation with the physical location distance.

7.2.3.2 Path-wise Similarity Metric

After computing the distance between signatures, the system needs to compute a similarity score for path-pairs that comprise of those signatures. This similarity score is used

to group path segments belonging to the same physical pathways.

Since node path segments belonging to the same physical pathway should be clustered, we use the following criteria to design a suitable similarity metric for path segments,

- **High Point-Wise Similarity:** Nodes moving along the same physical pathway are likely to observe similar location signatures. This means the path segment similarity must consider the signature similarity of individual location points on the paths.

- **Length Independent:** Nodes moving on the same physical pathway may travel over different portions of the pathway. These partial path segments must be clustered together. In other words, the path segment similarity must be independent of the length of the path segment.

- **Robust Against Path Warping:** Node movement varies in speed as well direction on a micro-scale. As the SugarTrail system does not use any inertial motion measurements, the path segment similarity should ignore warping of the path. In other words, skipping up to a specified maximum number of points to determine highest similarity should be possible.

To satisfy the above requirements, SugarTrail represents each node's path as a time-series of sequential signature and direction data and employs a modified version of the Longest Common Sequence (LCS) algorithm [91] to obtain similarity scores. Algorithm Box 7.2.2 describes the modified LCS algorithm.

Our modified LCS algorithm, uses the previously described signature distance metric for element-wise comparisons. Based on these comparisons, the algorithm increments the similarity score if the difference of the signature vectors $D_{i,j}$ and the index difference between the location points of the two paths is less than ϵ and δ, respectively.

ϵ and δ are thresholds that relax the correspondence requirement of signature vectors and point indices. ϵ is chosen through empirical measurements of RToF signature distance for points that are within $1m$ of each other. $delta$ is chosen according to the average walking speed of humans that limits the amount of warping that is permitted in matching two paths.

Algorithm 7.2.2 Modified LCS Subsequence Matching

for each pair of path segments P_m and P_n **do**
  Initialize the LCS matrix as an I-row, J-column all-zero matrix, where I and J are the numbers of location points, i.e. lengths, of P_m and P_n
  for each pair of points p_i^m and p_j^n in P_m and P_n **do**
    Find signature-wise signature distance $D_{i,j}$ between p_i^m and p_j^n using our hybrid distance metric.
    if $D_{i,j} < \epsilon$ and $|j - i| < \delta$ **then**
      Update the element at row i, column j in the LCS matrix as $L_{i,j} = e^{-D_{i,j}^2} + L_{i-1,j-1}$
    else
      $L_{i,j} = \max(L_{i-1,j}, L_{i,j-1})$
    end if
  end for
end for
Compute the LCS score for P_m and P_n as $\frac{L_{I,J}}{\min(I,J)}$

7.2.4 Multidimensional Scaling & Clustering

The system requires the clustering of similar paths as nodes of the VRM graph. Path signatures depend on distorted magnetometer readings that are not symmetrical for opposite directions. Consequently, the direction of traversing a path affects its signature and a dual graph is created. SugarTrail uses the K-means algorithm to cluster paths. Since, the paths are not points in Cartesian space, the system employs an intermediate step that uses the multi-dimensional scaling (MDS) algorithm to represent each path as a point in a n-dimensional Cartesian space [92].

Using the LCS similarity score matrix, the MDS algorithm creates a configuration of relative coordinates for the points, such that the Euclidean distances between them can approximately represent the original distance matrix. The K-means algorithm is then applied to the path coordinates to group them into path-clusters. The maximum number of clusters k is chosen based on the number of unidirectional pathways in the retail store.

7.2.5 Navigation

The SugarTrail system clusters all node collected path segments to create vertices of the VRM graph. It records the connection between individual path segments in a nodes path to create a connection matrix representing the edges. The system uses the VRM to route users from any designated point in the map to the desired destination.

7.2.5.1 Localization Over Paths

To navigate users, SugarTrail needs to determine their starting position, track whether they follow the navigation path, and note when they reach their destination. Algorithm 7.2.3 shows the steps involved in computing the alignment of two paths, where P_n is the user's current path and P_m is a recorded path with the best overall similarity score to P_n. Using the LCS path matching algorithm, the system locates the user by finding the best alignment of the current user path segment with recorded clusters of path segments. Subsequently, the system guides users to the destination by prompting them the next direction to travel in.

Localization of a user on a recorded path determines the accuracy of navigation. The performance of the system and it's dependence on various parameters is evaluated in the following section.

Algorithm 7.2.3 Modified LCS Path Alignment

for each path pair P_m and P_n **do**
  Initialize P_m's index counter to $i = 1$
  Initialize P_n's index counter to $j = 1$
  Initialize aligned path length to $k = 1$
  while $i \leq$ Length(P_m) and $j \leq$ Length(P_n) **do**
    if $D_{i,j} \leq \epsilon$ and $|j - i| \leq \delta$ **then**
      $j = j + 1$
    else if $L_{i+1,j} \geq L_{i,j+1}$ **then**
      $i = i + 1$
    else
      $j = j + 1$
    end if
  end while
end for

7.3 EVALUATION AND RESULTS

In this section, we present our evaluation methodology and experimental results. First, we validate the performance of the SugarTrail system in large-scale deployment. Second, we compare SugarTrail to an ideal system with access to accurate locations. And third, we analyze the effect of environmental conditions and amount of training data on performance.

7.3.1 Small-Scale Campus Experiment

To validate the clustering phase of SugarTrail, we created a small-scale live testbed in a single-intersection (T-shaped campus) hallway. This experiment enables us to determine if node traces can be segmented correctly and grouped into path-clusters corresponding to physical pathways.

The testbed has 10 anchors placed along the walls, distributed over the hallway, with no position measurements. The position of anchors is kept static for the period of the experiment. The mobile node in the test setup is attached to a laptop computer. The laptop

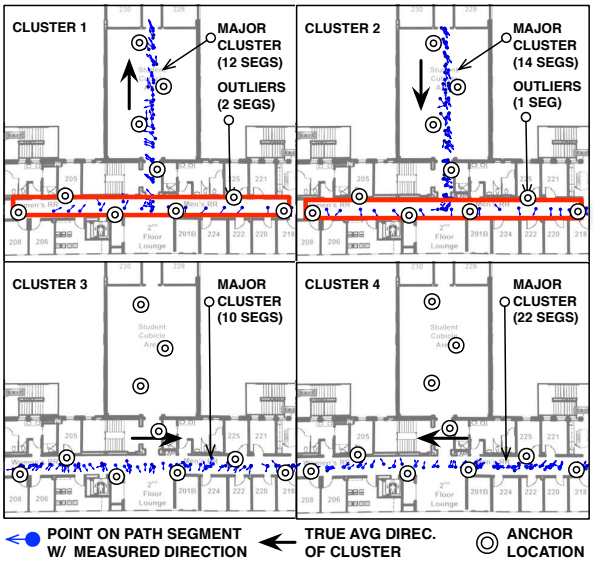


Figure 7.4: Shows the clustering of path segments plotted against ground-truth locations in the T-shaped campus hallway. The segments grouped in the wrong cluster are highlighted in red.

computer aggregates raw compass and radio measurements, and simultaneously collects the ground truth location (recorded by clicking on a pre-loaded map of the hallway).

In our preliminary tests, 3 nodes were moved along the hallways, recording radio and magnetometer measurements at every step. In all, 20 node movement traces were obtained with 395 signatures. The obtained traces were then segmented and clustered by SugarTrail to be compared with the ground-truth paths.

The results of the segmentation and clustering are shown, superimposed on the physical map, in Figure 7.4. The 20 node traces were segmented into 66 path segments. The SugarTrail algorithm was able to correctly cluster the path segments into 4 uni-directional path-clusters corresponding to the 2 pathways (horizontal and vertical arms of the "T") and the 2 major directions of movement in each. 88% of path segments were clustered correctly with path segments in their ground-truth pathway and the correct direction. 3 horizontal direction segments were mis-clustered with vertical segments. This was due to

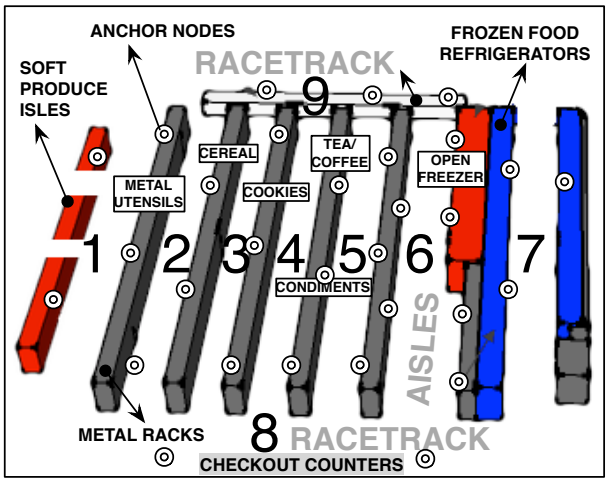


Figure 7.5: Layout of the supermarket used for the experimental deployment. The open pathways are labelled by numbers 1-9.

the high similarity in the compass readings (indoor magnetic distortion) and small length of the path segments. 5 segments were discarded for having no dominant direction.

The small-scale testbed experiment shows that the path-cluster algorithm performs well in realistic conditions. Based on these results, we perform an end-to-end system evaluation in a large-scale supermarket scenario.

7.3.2 Large-Scale Supermarket Experiment

To test the system in a large-scale realistic application scenario, we deployed anchors in a supermarket during operational hours.

The supermarket is a challenging environment for indoor localization due to a number of factors. First, the supermarket has metallic aisles, refrigerators, and merchandise that significantly affects RF as well as magnetic characteristics. Second, the environment is dynamic due to changes in arrangement of merchandise and constant movement of people. Consequently, the experiment provides a comprehensive evaluation of the performance of the SugarTrail system at scale.

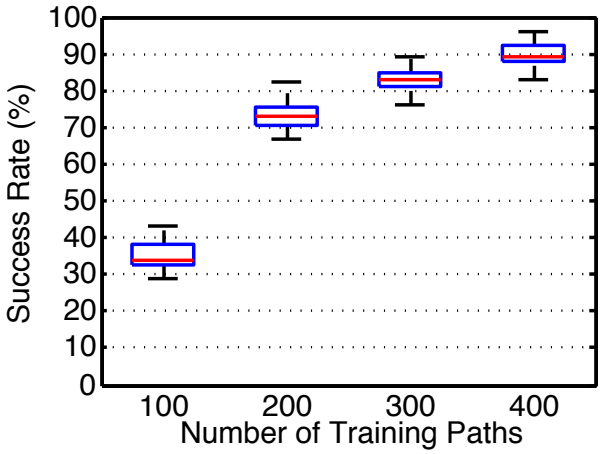


Figure 7.6: The figure shows a box plot (median, 25th and 75th percentiles) of the success rate of navigation obtained by different number of paths for training data.

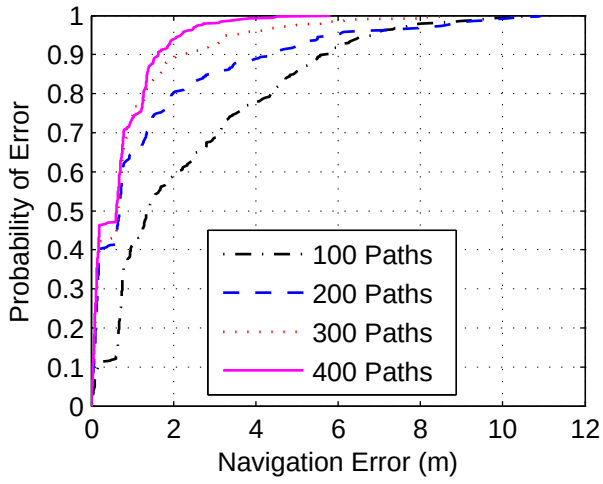


Figure 7.7: Figure shows the CDF of navigation error obtained by the SugarTrail system over the entire supermarket area.

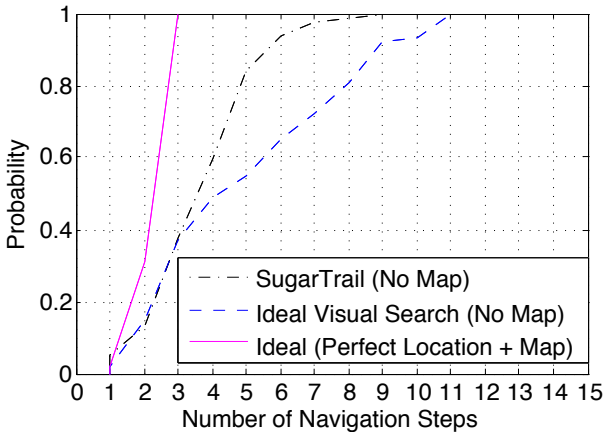


Figure 7.8: The figure shows the CDF of the number of navigation steps required to navigate a node between 1000 randomly chosen start and destination point pairs.

Real Measurements: The modeled nodes obtain measurements from a database of previously collected high granularity sensor readings. The database of ground truth sensor readings is created by collecting real radio and magnetometer measurements over a dense square grid (each point 0.6m apart from others) in all walkable aisles in the 26m$\times$24m supermarket.

Having validated the ability of SugarTrail to segment and cluster node paths in the campus testbed, the supermarket experiment allows us to scale the system to a large number of paths without sacrificing the fidelity of the environment's RF and magnetic characteristics.

7.3.2.1 Experiment Setup

For our experiments, we deployed 30 stationary anchors in the 26m $\times$ 24m New Wing Yuan supermarket in Santa Clara, CA. The anchors were placed at locations on the shelves but distributed over the area of the supermarket. Figure 7.5 shows the layout of the supermarket and arrangement of equipment and produce.

To create a database of RToF signatures and magnetometer readings for pathways in the

entire store (i.e. wherever walking is possible), with the same setup as the campus experiment, the system collected measurements on a grid of 0.6m. 1589 distinct locations were measured from the entire store. At every location, 20 distinct readings were obtained for RToF signatures as well magnetometer measurements, to capture the variance, creating a database of 31780 readings. 75% of these were used as part of the training data for SugarTrail, while 25% were used for the evaluation data set. In our experiments, we observed that signatures generally contain an average of 11 (min: 6, max: 15) anchor readings.

7.3.3 Performance Analysis

To exhaustively evaluate the performance of the SugarTrail system, we overlaid measured signatures in the supermarket with model generated node paths. A subset (75%) of the data is used for initial training and (25%) is used for testing.

400 node paths are used in the supermarket test environment for the learning phase of the SugarTrail system. Additional paths and signature data is used for the evaluation. Each node path traversed between 1-5 pathways in the supermarket and consisted of paths of varying lengths ($75 - 150m$).

Every node collected RToF and magnetometer measurements every 2 seconds. The interval of 2 seconds was derived to approximately capture node at an average pace of $0.5m/s$, corresponding to sensor node typical speed.

The node path training data set is provided as input to the SugarTrail algorithm, which in turn builds the corresponding VRM. A **new set of nodes (different from the training data set)** is used to query the SugarTrail system and follow navigation directions between randomly selected start and destination points. The performance of the system is evaluated based on the success rate of the navigation, the accuracy of navigation, and the number of

steps in the navigation route. These metrics are further defined in the following subsections.

7.3.3.1 Metric – Navigation Success Rate

The SugarTrail system guides nodes by providing a route consisting of a sequence of path-clusters, and a set of direction prompts (based on the expected node's local raw compass reference reading) to navigate along those clusters. We define the navigation success rate as the percentage of times a node/user is guided correctly from a start point to the proximity of node's desired destination point.

Specifically, an instance of node/user navigation is designated as a success if the following conditions are satisfied – First, the node is guided into the correct physical pathways as the desired destination point. Second, the node is not prompted to take an invalid direction (obstructed path) at any point along the route. Finally, the node is not guided into any incorrect pathways along the route. Such unexpected pathways may arise due to inconsistencies in the map or inaccuracy of localization.

To evaluate the navigation success rate of the SugarTrail system we performed 20 trials, each containing 100 node/user navigation requests separate from the training data set. In each trial, we note the number of times the navigation was successful according to the above mentioned criterion. Figure 7.6 shows a box plot of the success rate of navigation obtained with varying sizes of training data. The success rate of navigation improves as the size of training data increases over time, with the system achieving 90% success rate with 400 node path training data set. This corresponds to about half-a-day's worth of data.

The trend in success rates can be explained by the fact that the SugarTrail system relies on the record of peoples' prior movement to build the navigable virtual roadmap. Consequently, the larger the number of node paths recorded the better is the expected success rate of finding a route to the destination.

7.3.3.2 Metric - Navigation Accuracy

We measure the fine-grained ability of the SugarTrail system to guide a node/user to an exact destination point (specified within the VRM). We define this as navigation accuracy, which is the distance from the actual physical coordinates of nodes' desired destination to the location that the SugarTrail system can guide them to.

The navigation accuracy is obtained by computing the location error of the destination for 1000 node queries. Figure 7.7 depicts the CDF of navigation error obtained by the SugarTrail system over the entire supermarket area. As seen from the figure, the SugarTrail system can provide a navigation accuracy better than $1.3m$, 80% of the times, with a median accuracy of under $0.6m$. Figure 7.7 also shows that as the amount of training data increases, the navigation accuracy also improves.

7.3.3.3 Comparing Efficiency of Navigation

The SugarTrail system provides a set of direction prompts to guide the node/user to the destination via intermediate pathways (where node changes direction). In the indoor supermarket scenario, where length of any single pathway is relatively small, the efficiency of the system depends on the number of direction prompts or *navigation steps*. The number of navigation steps can be used to compare the performance of the SugarTrail system versus other possible approaches to navigating indoors. We compare the number of navigation steps required by SugarTrail to the following approaches:

- **Ideal Oracle System:** The ideal system has knowledge of the accurate location of the node as well as access to the map of the environment. This system provides the shortest, and most efficient path possible.

- **Visual Search:** The alternative to using SugarTrail, in indoor environments without

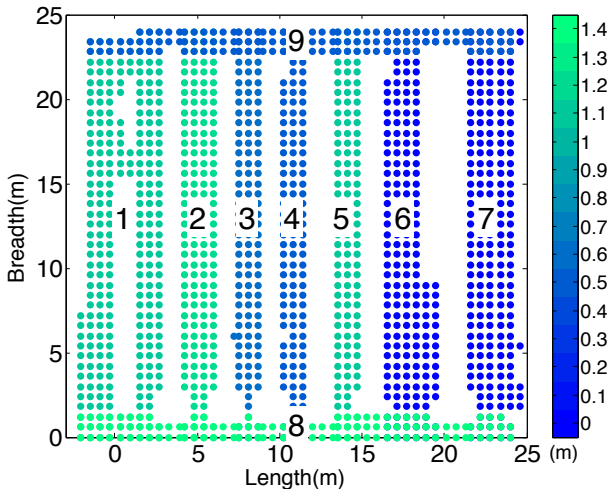


Figure 7.9: The figure shows the 80th percentile navigation accuracy over different regions of the supermarket computed using a 400 path training dataset.

access to location or maps, is to perform a visual sequential search of the environment.

Figure 7.8 shows the CDF of number of navigation steps required to navigate a node/user between 1000 randomly chosen start and destination point pairs using different approaches. The figure shows that difference in navigation steps between SugarTrail and the ideal benchmark system is under 2 steps for 80 percent of the cases as compared to over 5 steps for a visual search of the environment. Thus, the SugarTrail system provides significantly more efficient navigation performance with minimal setup and maintenance effort.

7.3.3.4 Effect of Environment

The layout and material-content of the environment affect the nature of RToF signatures as well as magnetometer measurements. The SugarTrail algorithm is designed to minimize the effect of such variations, however, the performance of the system is naturally affected by environmental factors. We analyze the distribution of navigation accuracy obtained over different geographical regions of the supermarket to gain insight into the environmental

effects.

Figure 7.9 shows the 80th percentile navigation accuracy over different regions of the supermarket. The accuracy is computed over 1000 destinations for each aisle of the supermarket. The figure shows that SugarTrail reduces the error attributed to environmental variations with accuracy better than 1.5 meters throughout the supermarket. The navigation accuracy is highest towards the wider refrigerator aisles (aisle 6 and 7). While, narrow aisles with metal utensils as well as many obstacles (aisles 1,2 and 3) show lower navigation accuracy.

7.4 RELATED WORK

Indoor navigation has been an active area of research in the past, primarily for robotics and increasingly for mobile computing application. Systems using different modalities such as radio frequency (RF), infrared (IR) and vision have been proposed to locate users inside building and consequently provide navigation guidance.

Map-based with Dedicated Infrastructure: Map-based systems require a physical schematic map of the environment and the location of the people to navigate. Several systems that deploy a specialized localization infrastructure have been developed for indoor environments, including infrared [93, 94], ultrasound [95], and more recently RFIDs [96]. Apart from needing maps, these systems require careful positioning and measurement when placing beacons and requires a much larger deployment and maintenance effort when compared to SugarTrail.

Map-based with RF-surveys: Another research thrust has been focused on utilizing RF signals (such as WiFi access points, or Bluetooth) in office environments to provide localization. These systems fingerprint each indoor location using received signal strength

(RSS) measurements from multiple access points. A location database of such fingerprints is created through manual surveys, active tagging, or user corrections [79, 97, 98, 99, 100, 101]. Every collected fingerprint must be tagged with a location and the required active user attention has been the major drawback of such approaches. Furthermore, Wi-Fi localization based navigation is generally courser-grained that required for retail store (e.g. supermarkets) applications. SugarTrail, on the other hand, does not require a pre-existing map and only requires annotation of interesting clusters of signatures for navigation, not every collected signature.

Map-based with Inertial Measurements: Another approach to localization has utilized inertial measurements with sensors attached to people to augment WiFi measurements [70, 102]. However, they depend on the existence of WiFi localization systems for initial estimate and is not fine-grained enough for aisle differentiation in stores. Furthermore, unlike SugarTrail, this approach requires known accurate maps and would be negatively affected by indoor magnetic interference.

Mapless with Advanced Sensing/Processing: Navigation approaches that do not need a preexisting map of the environment to operate have been actively explored for robot path finding applications. A large body of work on simultaneous localization and mapping (SLAM) [103, 104] algorithms allows a robot to determine its location and navigate without pre-existing maps. Recent work on WiFi-SLAM [87] uses a mobile robot to build a map in terms of RSS fingerprints. However, most SLAM implementations [105] are suited for robots with sensing (vision, laser range-finders, stereo-cameras, accurate inertial sensors etc.) capabilities unavailable on mobile devices.

7.5 CONCLUSION

This chapter presents the SugarTrail system that enables accurate indoor navigation leveraging the structured movement pathways in indoor environments such as retail stores. By using the sequence of measurements as combined signatures (paths), the system learns traversable path-cluster to build a navigable virtual roadmap of the environment. This method when applied to a supermarket environment was able to achieve a success rate of $> 85\%$ and an average accuracy of more than $0.7m$ without the need for existing maps and extensive active radio-signature surveys. The system requires no prior surveys, maps or infrastructure.

CHAPTER 8

CONCLUSIONS

This thesis presents hardware design, software architecture, simulation tools, network protocols, and planning algorithms for enabling controlled-mobile aerial sensor networking systems. The research validates proposed approaches through prototype implementations focusing on an illustrative application scenario – urban indoor emergency response. The thesis provides principles and guidelines in the design of controlled-mobile sensor networking system that operate in dynamic and potentially hazardous application environments such as search and rescue, hazardous material monitoring, urban combat surveillance applications.

8.1 CONTRIBUTIONS

In particular, this thesis contributes to the design and development of controlled-mobile sensing platforms by

- Designing a miniature(29g), low-cost($200), aerial sensor networking platform. To the best of our knowledge, it is significantly smaller than any other realized flying

sensor network platform and its cost is comparable to that of low-cost traditional static nodes.

- Evaluate hardware design choices and examining trade-offs that achieve a delicate balance between individual node capability and node resources for miniature aerial controlled-mobile platforms.

- Developing comprehensive cyber-physical simulation tools, focusing on the indoor emergency fire monitoring application, to enable inform design and comparative evaluations of controlled-mobile sensing systems.

These efforts help in creating a prototype and testbed for evaluating sensing and planning approaches for controlled-mobile sensing systems. In addition, this thesis introduces multiple planning approaches that enable these resource-constrained controlled-mobile sensor nodes to achieve sensing and monitoring goals in actual application scenarios. Specifically, this thesis introduces,

- A multi-node infrastructure-less coverage algorithm that uses self-dispersed anchor nodes to obtain radio RF-signatures and provides a common spatial frame of reference for controlled-mobile swarm nodes and uses it to attain efficient sweep coverage of previously unseen environments.

- A fine-grained multi-room autonomous deployment technique that combines a collaborative location estimation and an adaptive planning algorithm to guide nodes to pre-assigned in scenarios with no pre-existing location infrastructure and coarse grained maps.

- An algorithm that enables a controlled-mobile network to provide fine-grained indoor

navigation guidance in previously unseen environments to users (e.g. fire-fighters) by automatically learning a virtual roadmap from node movement traces. The algorithm leverages the structured movement patterns of indoor environments. We use a retail store environment as an illustrative example of a structured indoor environment.

Results show that proposed system architecture and algorithms have the potential to enable controlled-mobile sensor networking systems that autonomously deploy and monitor unknown areas. The work should provide a foundation for further research in the emerging area of controlled-mobile sensing.

8.2 FUTURE DIRECTIONS

Several aspects of this thesis warrant further discussion. This section presents some of the major future research directions and possible extensions as a continuation of this thesis work.

8.2.1 Platform

Interoperability with Static Networks. Pre-deployed static sensor networks have certain advantages such as knowledge of exact locations and ability to provide data from regions that may be occluded due to closed doorways or structural collapse. On the other hand, the point-of-emergency deployment capability and mobility of SensorFly nodes allow larger deployments and higher resolution sensing of spaces where they can be introduced.

A hybrid approach with SensorFly nodes working in collaboration with static sensing infrastructure, where it exists, can combine the advantages of both approaches. Thus, research into making controlled-mobile and static networks inter-operable, as well as, work

on leveraging the static network to augment the mobile-network's localization protocols could be beneficial.

Radio Propagation. Radio propagation may be affected adversely in emergency response environments such as in presence of smoke and fire. This may impact the speed and extent of coverage obtained for a deployment of specific size. Recent research has studied the effect of fire on wireless propagation in wildfire environments [106]. The work suggests that ionization in flames causes particular frequency bands to be attenuated. An empirical study of the characteristics of fire propagation in indoor environments for our chirp spread spectrum radio would be helpful in obtaining a more accurate estimation of system performance in real deployments.

Energy and Flight Time. Miniature aerial platforms remain limited in their flight time due to high power consumption of motors. Improvements in mechanical design and battery technology can extend flight times to the order of 15-20 minutes, which however may not be sufficient for many applications. We seek to explore collaborative techniques to distribute movement tasks and manage energy across the group. For example, in the fire monitoring scenario examined, nodes can collaborate with nodes in their close proximity, and *duty cycle* their flying task. One node can land and act as relay, while another flies and senses the temperature. When the battery level of the flying node becomes low, the roles can be reversed.

8.2.2 Network

The one-way discovery mechanism, although optimizing energy efficiency for neighbor discovery, brings challenges in group membership propagation and maintenance. We designed a synchronized listening and evenly spaced transmitting (SLEST) mechanism to

archive group-level coordination by reusing neighbor discovery beacons. Our experiments show that the protocol scales well with network density. Several extensions can be built on top of WiFlock and possibly improve its performance.

Accommodating Large Groups. WiFlock as presented in this chapter assumes that all the group information can fit into a single frame, which in our case has a maximum size of 128 bytes, limiting the group size to a about 50 nodes (assuming 4 bytes information per node plus frame header). However, WiFlock can accommodate larger group sizes by splitting the group table across multiple frames and multiple beacons. Since correct operation of the synchronized listening and evenly spaced transmissions only depend on the smallest node ID that a node has received so far, it not necessary to repeat the smallest node ID across multiple frames and beacons. However, to prevent premature timeout of node information, a larger initial node TTL value needs to be used with large groups. For large groups spanning multiple beacons, the mapping of nodes rank to the beacon slot should be done with proper attention to the number of slots within a given beacon period.

Dedicated RSSI Measurements. Currently, we use the RSSI value reported by the AGC module of the radio to detect the presence of an on going beacon. This RSSI value is a byproduct of the AGC module, where the primary function is to use the RSSI value to maintain a constant amplified signal inside the radio. Because of this, the carrier detection time is limited by how the AGC module is designed. In contrast, a dedicated carrier sensing module optimized for faster carrier detection might be able to reduce the wakeup duration, hence the duty cycle, below what is reported in this chapter.

8.2.3 Planning

Multi-hop Communication. The explorer nodes obtain RToF measurements from the deployed anchor nodes as a location signature. This requires explorer nodes to be in single-hop communication range with a large enough sub-set of anchor nodes to obtain a unique radio location signatures. In this thesis, we limited our application and evaluation to a single-space coverage scenario where all nodes are within radio range. The explorer nodes stop exploring if they lose radio range with deployed anchor nodes. Thus explorer nodes employ a single-hop communication scheme to relay data back to the base station. We continue to explore large-space or multi-room scenarios in our ongoing work, where the network can extend its reach by progressively deploying exploring nodes as new anchors and requires support for multi-hop communication schemes.

Distributed Operation. The current SugarMap implementation has a central base station that aggregates the coverage estimates of individual nodes to arrive at a global coverage map. The global coverage map is utilized by the explorer nodes to decide their next movement. For a single-space scenario with single-hop communication range this simple mechanism reduces communication and computation on the nodes. For large-space and multi-room scenarios, it is desirable for nodes to broadcast their local coverage maps and for each node to itself aggregate neighboring nodes' maps to determine global coverage. Such distributed operation would remove the single point of failure at the base station and also provide latency advantages as nodes would not have to communicate with the base station over multiple hops. However, a distributed scheme would require consideration for lack of consensus on coverage maps due to lossy wireless links. We seek to explore this in our future work.

REFERENCES

[1] X. Jiang, N. Y. Chen, J. I. Hong, K. Wang, L. Takayama, and J. A. Landay, "Siren: Context-aware Computing for Firefighting," in *Proceedings of the 2nd International Conference on Pervasive Computing*, 2004, pp. 87–105. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.58.3182

[2] M. Klann, T. Riedel, H. Gellersen, C. Fischer, G. Pirkl, K. Kunze, M. Beuster, M. Beigl, O. Visser, and M. Gerling, "LifeNet: an Ad-hoc Sensor Network and Wearable System to Provide Firefighters with Navigation Support," in *Proceedings of the 9th International Conference on Ubiquitous Computing*, 2007. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.103.5633

[3] R. Upadhyay, "Towards a Tailored Sensor Network for Fire Emergency Monitoring in Large buildings," in *Proceedings of the 1st IEEE International Conference on Wireless Emergency and Rural Communications, Rome*, 2007. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Towards+a+Tailored+Sensor+Network+for+Fire+Emergency+Monitoring+in+Large+Buildings#0

[4] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich, "Mobiscopes for Human Spaces," *IEEE*

Pervasive Computing, vol. 6, no. 2, pp. 20–29, Apr. 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4160602

[5] B. Bethke, M. Valenti, and J. P. How, "Uav task assignment," *Robotics & Automation Magazine, IEEE*, vol. 15, no. 1, pp. 39–44, 2008.

[6] G. Hoffmann, D. G. Rajnarayan, S. L. Waslander, D. Dostal, J. S. Jang, and C. J. Tomlin, "The Stanford testbed of autonomous rotorcraft for multi agent control (STARMAC)," in *Digital Avionics Systems Conference, 2004. DASC 04. The 23rd*, vol. 2, 2004, pp. 12.E.4—-121–10 Vol.2. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1390847

[7] "USFA Firefighter Fatality Reports and Statistics," in *http://www.usfa.dhs.gov/fireservice/fatalities/statistics/index.shtm*. [Online]. Available: http://www.usfa.dhs.gov/fireservice/fatalities/statistics/index.shtm

[8] "FIRE project," in *http://fire.me.berkeley.edu/*. [Online]. Available: http://fire.me.berkeley.edu/

[9] H. Liu, J. Li, Z. Xie, S. Lin, K. Whitehouse, J. A. Stankovic, and D. Siu, "Automatic and robust breadcrumb system deployment for indoor firefighter applications," in *International Conference On Mobile Systems, Applications And Services*, 2010, pp. 21–34. [Online]. Available: http://portal.acm.org/citation.cfm?id=1814433.1814438

[10] R. Mangharam, A. Rowe, and R. Rajkumar, "FireFly: a cross-layer platform for real-time embedded wireless networks," *Real-Time Systems*, vol. 37, no. 3, pp. 183–231, Dec. 2007. [Online]. Available: http://dx.doi.org/10.1007/s11241-007-9028-z

[11] L. Shantou Syma Toys Industrial Co. Syma RC Helicopter. http://www.symatoys.com/. [Online]. Available: http://www.symatoys.com/

[12] Parrot SA. {Parrot AR.Drone.} http://ardrone.parrot.com/parrot-ar-drone/usa/. [Online]. Available: http://ardrone.parrot.com/parrot-ar-drone/usa

[13] C. H. Everett, "Survey of collision avoidance and ranging sensors for mobile robots," *Robotics and Autonomous Systems*, vol. 5, no. 1, pp. 5–67, May 1989. [Online]. Available: http://dx.doi.org/10.1016/0921-8890(89)90041-9

[14] Nanotron Tecnologies Gmbh, "nanoLOC AVR Module Technical Description," 2008. [Online]. Available: http://www.nanotron.com/

[15] Digi International, "XBee Multipoint RF Modules Datasheet," 2009. [Online]. Available: www.digi.com

[16] Honeywell, "HMC6346 3-Axis Compass Datasheet," 2008. [Online]. Available: www.honeywell.com

[17] R. Morlok and M. Gini, "Dispersing robots in an unknown environment," *Distributed Autonomous Robotic Systems 6*, 2007. [Online]. Available: http://www.springerlink.com/index/K26W678734587TH7.pdf

[18] Nanotron Tecnologies Gmbh, "Real Time Location Systems White Paper Version 1.02," pp. 1–20, May 2007. [Online]. Available: http://www.nanotron.com

[19] H. Huang, G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering," *Proceedings of the 2009 IEEE international conference on Robotics and*

Automation, 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1703840

[20] P. Muren, "Passively stable rotor system for indoor hovering UAS," *Society*, no. July, 2008.

[21] MaxBotix, "LV-MaxSonar-EZ1 High Performanxe Sonar Range Finder Datasheet," 2007. [Online]. Available: www.maxbotix.com

[22] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. F. Abdelzaher, "Range-free localization and its impact on large scale sensor networks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 4, no. 4, 2005. [Online]. Available: http://portal.acm.org/citation.cfm?id=1113830.1113837

[23] D. Niculescu and B. Nath, "Ad-Hoc positioning systems (APS)," *Proceedings IEEE GlobeCom*, pp. 2926–2931, 2001. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Ad-hoc+positioning+system#5

[24] "Intelli Mini RC Helicopter," in *http://www.hobbytron.com*. [Online]. Available: http://hobbytron.com/

[25] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "An analysis of a large scale habitat monitoring application," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2004, pp. 214–226. [Online]. Available: http://dx.doi.org/10.1145/1031495.1031521

[26] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *OSDI '06: Proceedings of*

the 7th symposium on Operating systems design and implementation. Berkeley, CA, USA: USENIX Association, 2006, pp. 381–396. [Online]. Available: http://portal.acm.org/citation.cfm?id=1298455.1298491

[27] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet," in *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, vol. 37, no. 10. New York, NY, USA: ACM Press, Oct. 2002, pp. 96–107. [Online]. Available: http://dx.doi.org/10.1145/605397.605408

[28] A. Kansal, M. Rahimi, W. J. Kaiser, M. B. Srivastava, G. Pottie, and D. Estrin, "Controlled Mobility for Sustainable Wireless Networks," in *IEEE Sensor and Ad Hoc Communications and Networks (SECON)*. nstitute of Electrical and Electronics Engineers, Inc., 2004.

[29] A. A. Somasundara, A. Kansal, D. D. Jea, D. Estrin, and M. B. Srivastava, "Controllably Mobile Infrastructure for Low Energy Embedded Networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 8, pp. 958–973, Aug. 2006. [Online]. Available: http://dx.doi.org/10.1109/TMC.2006.109

[30] O. Holland, J. Woods, R. De Nardi, and A. Clark, "Beyond swarm intelligence: the UltraSwarm," in *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, 2005, pp. 217–224. [Online]. Available: http://dx.doi.org/10.1109/SIS.2005.1501625

[31] J. Allred, A. Hasan, S. Panichsakul, W. Pisano, P. Gray, J. Huang, R. Han, D. Lawrence, and K. Mohseni, "An Airborne Wireless Sensor Network of Micro-Air Vehicles," in *Proceedings of the 5th international conference on Embedded networked sensor systems*. ACM, 2007, p. 129. [Online]. Available: http://portal.acm.org/citation.cfm?id=1322275

[32] R. J. Wood, "The First Takeoff of a Biologically Inspired At-Scale Robotic Insect," *IEEE transactions on robotics*, vol. 24, no. 2, pp. 341–347, 2008. [Online]. Available: http://cat.inist.fr/?aModele=afficheN&cpsidt=20276799

[33] {RoboBees.} http://robobees.seas.harvard.edu.

[34] R. Vaughan, "Massively multi-robot simulation in stage," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 189–208, Aug. 2008. [Online]. Available: http://www.springerlink.com/content/n07226r442887871/

[35] L. Hugues and N. Bredeche, "Simbad: An Autonomous Robot Simulation Package for Education and Research," in *From Animals to Animats 9*, ser. Lecture Notes in Computer Science, S. Nolfi, G. Baldassarre, R. Calabretta, J. Hallam, D. Marocco, J.-A. Meyer, O. Miglino, and D. Parisi, Eds. Springer Berlin / Heidelberg, 2006, vol. 4095, pp. 831–842. [Online]. Available: http://dx.doi.org/10.1007/11840541_68DO-10.1007/11840541_68

[36] R. D. Peacock, W. W. Jones, P. A. Reneke, and G. P. Forney, "CFAST Consolidated Model of Fire Growth and Smoke Transport (Version 6) Users Guide," *Nist Special Publication*, no. Version 6, 2005.

[37] D. Rutledge, *Investigation of indoor radio channels from 2.4 GHz to 24 GHz*. IEEE. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1219197

[38] P. Dutta and D. Culler, "Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications," *Conference On Embedded Networked Sensor Systems*, pp. 71–84, 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1460420

[39] A. Kandhalu, K. Lakshmanan, and R. Rajkumar, "U-connect: a low-latency energy-efficient asynchronous neighbor discovery protocol," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM, 2010, pp. 350–361. [Online]. Available: http://portal.acm.org/citation.cfm?id=1791212.1791253

[40] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," *Conference On Embedded Networked Sensor Systems*, p. 95, 2004. [Online]. Available: http://portal.acm.org/citation.cfm?id=1031508

[41] Texas Instruments, "CC2500Single Chip Low Cost Low Power RF Transceiver," 2006. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:CC2500#1

[42] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," *Conference On Embedded Networked Sensor Systems*, p. 307, 2006. [Online]. Available: http://portal.acm.org/citation.cfm?id=1182807.1182838

[43] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal, "Firefly-inspired sensor network synchronicity with realistic radio effects," *Conference On Embedded Networked Sensor Systems*, p. 142, 2005. [Online]. Available: http://portal.acm.org/citation.cfm?id=1098918.1098934

[44] A. Rowe, R. Mangharam, and R. Rajkumar, "RT-Link: A Time-Synchronized Link Protocol for Energy- Constrained Multi-hop Wireless Networks," *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, pp. 402–411, 2006. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4068297

[45] Texas Instruments Corporation, "eZ430-RF2500 Development Tool User's Guide," \url{http://focus.ti.com/lit/ug/slau227e/slau227e.pdf}.

[46] Epson Toyocom Corporation, "SG-3030 Crystal Oscillator," \url{http://www.eea.epson.com/portal/pls/portal/docs/1/1219458.PDF}.

[47] Texas Instruments, "Design Note DN505," pp. 1–11, 2010.

[48] R. Cohen and B. Kapchits, "Continuous Neighbor Discovery in Asynchronous Sensor Networks," *IEEE/ACM Transactions on Networking*, no. 99, pp. 1–1, 2010. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5508435

[49] P. Zhang, C. Sadler, S. Lyon, and M. Martonosi, "Hardware design experiences in ZebraNet," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM, 2004, pp. 227–238. [Online]. Available: http://portal.acm.org/citation.cfm?id=1031522&dl=7

[50] C. Perkins, E. Belding-Royer, and S. Das, "RFC3561: Ad hoc On-Demand Distance Vector (AODV) Routing," *Internet RFCs*, 2003. [Online]. Available: http://portal.acm.org/citation.cfm?id=RFC3561

[51] M.-h. A. H. Networks, Y.-c. Tseng, C.-s. Hsu, and T.-y. Hsieh, "Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks," *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 00, no. c, pp. 200–209, 2002. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1019261

[52] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," *INFOCOM 2002. Twenty-First*, pp. 1567–1576, 2002. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1019408

[53] M. McGlynn and S. Borbash, "Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks," *symposium on Mobile ad hoc*, p. 137, 2001. [Online]. Available: http://portal.acm.org/citation.cfm?id=501435

[54] A. Purohit, Z. Sun, F. Mokaya, and P. Zhang, "SensorFly: Controlled-mobile sensing platform for indoor emergency response applications," in *In Proceeding of the 10th International Conference on Information Processing in Sensor Networks (IPSN)*, 2011, pp. 223–234.

[55] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 2. IEEE, 2001,

pp. 1927–1933. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=932890&contentType=Conference+Publications

[56] R. Morlok and M. Gini, "Dispersing robots in an unknown environment," in *7th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.99.7977

[57] L. Ludwig and G. Maria, "Robotic swarm dispersion using wireless intensity signals," in *International Symposium on Distributed Autonomous Robotic Systems*, 2006. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.153.9210

[58] J. McLurkin and J. Smith, "Distributed Algorithms for Dispersion in Indoor Environments using a Swarm of Autonomous Mobile Robots," *7th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2004.

[59] J. Liu, *Monte Carlo strategies in scientific computing*. Springer Publishing Company, Jan. 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1571802http://books.google.com/books?hl=en&lr=&id=R8E-yHaKCGUC&oi=fnd&pg=PR7&dq=Monte+Carlo+Strategies+in+Scientific+Computing&ots=WcsTyQy97M&sig=6Nuq_i2o0qowYa8l39Eh7elEbF4

[60] D. W. Gage, "Randomized Search Strategies With Imperfect Sensors," in *SPIE Mobile Robors VIII*, Boston, 1993, pp. 270–279. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.3301

[61] N. Hazon, F. Mieli, and G. Kaminka, "Towards robust on-line multi-robot coverage," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, pp. 1710–1715. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=1641953&contentType=Conference+Publications

[62] A. Purohit and P. Zhang, "Controlled-mobile sensing simulator for indoor fire monitoring," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, 2011, pp. 1124–1129.

[63] Nanotron Tecnologies Gmbh, "Real Time Location Systems (RTLS)," 2007.

[64] H. Choset, "Coverage for robotics - A survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1-4, pp. 113–126, May 2001. [Online]. Available: http://dl.acm.org/citation.cfm?id=590424.590433

[65] I. Wagner, M. Lindenbaum, and A. Bruckstein, "Distributed covering by ant-robots using evaporating traces," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 918–933, 1999. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=795795&contentType=Journals+&+Magazines

[66] S. Thrun, "An Online Mapping Algorithm for Teams of Mobile Robots," *International Journal of Robotics Research*, 2001. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.8778

[67] M. J. M. Andrew Howard, "Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem," *Proceedings of the 6th International Symposium on Distributed*

Autonomous Robotics Systems (DARS02), 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.8990

[68] G. S. S. Maxim A. Batalin, "Spreading Out: A Local Approach to Multi-robot Coverage," *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics System*, 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.1.2838

[69] W. B. Dieter Fox , "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots," in *Sixteenth National Conference on Artificial Intelligence (AAAI'99)*, 1999. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.342

[70] L. Klingbeil and T. Wark, "A wireless sensor network for real-time indoor localisation and motion monitoring," in *Proceedings of the 7th international conference on Information processing in sensor networks*. IEEE Computer Society Washington, DC, USA, 2008, pp. 39–50. [Online]. Available: http://portal.acm.org/citation.cfm?id=1372720

[71] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, May 2001. [Online]. Available: http://dl.acm.org/citation.cfm?id=379734.379740

[72] A. Purohit, Z. Sun, and P. Zhang, "SugarMap: Location-less Coverage for Micro-aerial Sensing Swarms," in *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, ser. IPSN '13.

New York, NY, USA: ACM, 2013, pp. 253–264. [Online]. Available: http://doi.acm.org/10.1145/2461381.2461412

[73] I. Constandache, R. R. Choudhury, and I. Rhee, "Towards Mobile Phone Localization without War-Driving," in *INFOCOM, 2010 Proceedings IEEE*, 2010, pp. 1–9.

[74] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2006.

[75] A. Doucet, J. F. G. de Freitas, and N. J. Gordon, Eds., *Sequential Monte Carlo methods in practice*. Springer-Verlag, 2001.

[76] Z. Sun, A. Purohit, S. Pan, F. Mokaya, R. Bose, and P. Zhang, "Polaris: Getting Accurate Indoor Orientations for Mobile Devices Using Ubiquitous Visual Patterns on Ceilings," in *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, ser. HotMobile '12. New York, NY, USA: ACM, 2012, pp. 14:1—-14:6. [Online]. Available: http://doi.acm.org/10.1145/2162081.2162101

[77] A. Kolling and S. Carpin, "Extracting surveillance graphs from robot maps," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 2323–2328.

[78] J. Borenstein, L. Feng, and H. R. Everett, *Navigating Mobile Robots: Systems and Techniques*. Natick, MA, USA: A. K. Peters, Ltd., 1996.

[79] P. Bahl and V. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, vol. 2, no. c.

IEEE, 2000, pp. 775–784. [Online]. Available: http://www.mendeley.com/research/radar-an-inbuilding-rfbased-user-location-and-tracking-system-1/

[80] A. Purohit, Z. Sun, S. Pan, and P. Zhang, "SugarTrail : Indoor Navigation in Retail Environments without Surveys and Maps," in *Tenth IEEE Conference on Sensing, Communication, and Networking (SECON)*, 2013.

[81] A. Howard and M. Mataric, "Cover me! a self-deployment algorithm for mobile sensor networks," in *2002 International Conference on Robotics and Automations, Washington DC*. Citeseer, 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.16.1394&rep=rep1&type=pdf

[82] G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba., "A solution to the simultaneous localisation and map building ({SLAM}) problem," *IEEE Transactions of Robotics and Automation*, vol. 17, no. 3, pp. 229–241, 2001.

[83] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping iwht {Rao-Blackwellized} particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 36–46, 2007.

[84] A. Ladd, K. Bekris, A. Rudys, D. Wallach, and L. Kavraki, "On the Feasibility of Using Wireless Ethernet for Indoor Localization," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 555–559, 2004.

[85] J. Twigg, J. Fink, P. Yu, and B. Sadler, "Efficient base station connectivity area discovery," *International Journal of Robotics Research*, 2013.

[86] G. Shen, Z. Chen, P. Zhang, T. Moscibroda, and Y. Zhang, "Walkie-Markie: Indoor Pathway Mapping Made Easy," in *Proceedings of the 10th USENIX*

Conference on Networked Systems Design and Implementation, ser. nsdi'13. Berkeley, CA, USA: USENIX Association, 2013, pp. 85–98. [Online]. Available: http://dl.acm.org/citation.cfm?id=2482626.2482636

[87] B. Ferris, D. Fox, and N. Lawrence, "WiFi-SLAM using Gaussian process latent variable models," pp. 2480–2485, Jan. 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1625275.1625675

[88] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No Need to War-drive: Unsupervised Indoor Localization," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '12. New York, NY, USA: ACM, 2012, pp. 197–210. [Online]. Available: http://doi.acm.org/10.1145/2307636.2307655

[89] J. S. Larson, E. T. Bradlow, and P. S. Fader, "An exploratory look at supermarket shopping paths," *International Journal of Research in Marketing*, vol. 22, no. 4, pp. 395–414, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167811605000479

[90] J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavai, and M. Wiseman, "Indoor location sensing using geo-magnetism," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*, ser. MobiSys '11. New York, NY, USA: ACM, 2011, pp. 141–154. [Online]. Available: http://doi.acm.org/10.1145/1999995.2000010

[91] L. Bergroth, H. Hakonen, and T. Raita, "A survey of longest common subsequence algorithms," *Proceedings of SPIRE 2000.*, pp. 39–48, 2000. [Online]. Available:

http://dx.doi.org/10.1109/SPIRE.2000.878178

[92] S. Schiffman, M. L. Reynolds, and F. W. Young, *Introduction to Multidimensional Scaling: Theory, Methods, and Applications*. Emerald Group Publishing Limited, 1981. [Online]. Available: http://www.amazon.com/Introduction-Multidimensional-Scaling-Methods-Applications/dp/0126243506

[93] R. Want, A. Hopper, V. Falcão, and J. Gibbons, "The active badge location system," *ACM Transactions on Information Systems*, vol. 10, no. 1, pp. 91–102, Jan. 1992. [Online]. Available: http://portal.acm.org/citation.cfm?id=128756.128759

[94] A. Ward, A. Jones, and A. Hopper, "A New Location Technique for the Active Office," *IEEE Personal Communications*, vol. 4, pp. 42–47, 1997. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.49.1301

[95] N. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, vol. 2000, no. August. ACM New York, NY, USA, 2000, pp. 32–43. [Online]. Available: http://portal.acm.org/citation.cfm?id=345917&coll=GUIDE&dl...N=20241836&ret=1

[96] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, "LANDMARC: Indoor Location Sensing Using Active RFID," *Wireless Networks*, vol. 10, no. 6, pp. 701–710, Nov. 2004. [Online]. Available: http://portal.acm.org/citation.cfm?id=1035678.1035686

[97] P. Bolliger, "Redpin - adaptive, zero-configuration indoor localization through user collaboration," in *Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments - MELT '08.*

New York, New York, USA: ACM Press, Sep. 2008, p. 55. [Online]. Available: http://portal.acm.org/citation.cfm?id=1410012.1410025

[98] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking - MobiCom '10*. New York, New York, USA: ACM Press, Sep. 2010, p. 173. [Online]. Available: http://portal.acm.org/citation.cfm?id=1859995.1860016

[99] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki, "Practical robust localization over large-scale 802.11 wireless networks," in *Proceedings of the 10th annual international conference on Mobile computing and networking - MobiCom '04*. New York, New York, USA: ACM Press, Sep. 2004, p. 70. [Online]. Available: http://portal.acm.org/citation.cfm?id=1023720.1023728

[100] J.-g. Park, B. Charrow, D. Curtis, J. Battat, E. Minkov, J. Hicks, S. Teller, and J. Ledlie, "Growing an organic indoor location system," *International Conference On Mobile Systems, Applications And Services*, pp. 271–284, 2010. [Online]. Available: http://portal.acm.org/citation.cfm?id=1814433.1814461

[101] M. Youssef and A. Agrawala, "The Horus WLAN location determination system," in *Proceedings of the 3rd international conference on Mobile systems, applications, and services - MobiSys '05*. New York, New York, USA: ACM Press, Jun. 2005, p. 205. [Online]. Available: http://portal.acm.org/citation.cfm?id=1067170.1067193

[102] O. Woodman and R. Harle, "Pedestrian localisation for indoor environments," in *Proceedings of the 10th international conference on Ubiquitous computing -*

UbiComp '08. New York, New York, USA: ACM Press, Sep. 2008, p. 114. [Online]. Available: http://portal.acm.org/citation.cfm?id=1409635.1409651

[103] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, Jun. 2006. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?reload=true&&arnumber=1638022

[104] J. Leonard and H. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*. IEEE, pp. 1442–1447. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=174711

[105] A. J. Davision and D. W.Murray, "Simultaneous Localisation and Map-Building Using Active Vision," *Transections on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 865–880, 2002.

[106] J. Boan Jonathan Alexander, "Radio Experiments With Fire," *IEEE Antennas and Wireless Propagation Letters*, vol. 6, pp. 411–414, 2007.