Wireless Sensor Network for Forestry Applications:
Overcoming the Uncertainty

by

HE, Yuan

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
in Computer Science and Engineering

May 2010, Hong Kong
Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

________________________

He Yuan

HE, Yuan
Wireless Sensor Network for Forestry Applications: Overcoming the Uncertainty

by

HE, Yuan

This is to certify that I have examined the above PhD thesis and have found that it is complete and satisfactory in all respects, and that any and all revisions required by the thesis examination committee have been made.

Dr. Yunhao Liu, Supervisor

Prof. Mounir Hamdi, Head of Department

Department of Computer Science and Engineering
31 May 2010
Wireless Sensor Network for Forestry Applications:
Overcoming the Uncertainty

by Yuan He

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Abstract

A wireless sensor network (WSN) is a self-organized wireless network that consists of a number of sensor nodes. The deployment of a number of sensor nodes enables people to monitor and interact with the physical world. Forest is one of those environments where WSNs are applied. Under the circumstances of global climate changes and environmental pollution, WSNs for forestry applications attract increasing attention in recent years.

This dissertation research focuses on developing techniques that make the sensor network systems more applicable and efficient for practical forestry applications. I have been working on GreenOrbs, a long-term large-scale WSN system deployed in the forest. In order to overcome the uncertainty emerging from the forestry deployments, I address several key issues, including error-resilient localization for networked sensor nodes against irregular wireless signals, forwarding-quality-aware data collection against unstable link and node behavior, and quantitative canopy closure estimates against error-prone sensor readings.

I develop theoretical principles, design practical protocols, and implement my ideas with GreenOrbs. I evaluate those approaches through real-world large-scale experiments and trace-driven simulations. The results validate their effectiveness and efficiency in overcoming the uncertainty. The proposed approaches can be further applied to other fields.
ACKNOWLEDGMENT

First and foremost, I would like to deeply thank my advisor Dr. Yunhao Liu for his continual guidance and helping me throughout my PhD study. What I have learned from him is far more than knowledge and research experience.

I also want to thank Dr. Lionel M. Ni, Dr. Vicent Yun Shen, Dr. Xiang-Yang Li, Dr. Weijia Jia, Dr. Ke Yi, Dr. Lin Gu, Dr. Lei Chen, Dr. Xue Liu, and Dr. Jizhong Zhao for their precious suggestions and comments during this work.

I am thankful to Lufeng Mo, Dr. Hao Ren, Dr. Feng Hong, Dr. Baijian Yang, Ying Guo, and Shaojie Tang for their kind help and extensive discussion on research.

I am also grateful to my former and current colleagues in the department of computer science and engineering of HKUST, especially Dr. Mo Li, Dr. Jinsong Han, Dr. Deke Guo, Jiliang Wang, Wei Xi, Wei Dong, Zheng Yang, Kebin Liu, Tao Chen, Shan Chang, Jingyao Dai, Dr. Yiyang Zhao, Chen Qian, Dr. Yunhuai Liu, and Dr. Yanmin Zhu for their valuable inputs and exchanging ideas with me.

I would like to thank my fiancée Haizhu Yu for her unconditional love and encouragement. Her patience and support accompany me throughout the most difficult time of my PhD study.

I dedicate this thesis to my family. They are always on the bottom of my heart.
TABLE OF CONTENTS

1 INTRODUCTION ......................................................................................................................... 1

1.1 Research Background ........................................................................................................ 2

1.2 Motivation .......................................................................................................................... 3

1.3 Contributions ....................................................................................................................... 6

1.4 Dissertation Organization .................................................................................................... 7

2 RELATED WORK .................................................................................................................. 9

2.1 Existing Sensor Network Systems ....................................................................................... 9
  2.1.1 Application-Driven Systems ........................................................................................ 9
  2.1.2 Sensor Network Test-beds .......................................................................................... 12

2.2 Localization ......................................................................................................................... 13
  2.2.1 Range-Based Approaches ........................................................................................... 13
  2.2.2 Range-Free Approaches ............................................................................................ 14
  2.2.3 Error Control and Management ................................................................................... 14

2.3 Link Estimation and Data Collection .................................................................................. 15
  2.3.1 Link Estimation Methods ............................................................................................. 15
  2.3.1 Data Collection Protocols .......................................................................................... 17

3 GREENORBS: A LONG-TERM LARGE-SCALE WSN IN THE FOREST ...... 19

3.1 Project Motivation .............................................................................................................. 19
  3.1.1 Research Background in the WSN field ................................................................... 19
  3.1.2 Forestry Applications ................................................................................................. 20

3.2 System Design Overview ................................................................................................... 21
  3.2.1 Building Blocks .......................................................................................................... 21
  3.2.2 System Settings .......................................................................................................... 23

3.3 Real-World Deployments ................................................................................................. 25

4 ERROR-RESILIENT LOCALIZATION .......................................................... 26

4.1 Preliminary Attempts .......................................................................................................... 27
  4.1.1 Monotonicity of RSSI ................................................................................................. 28
  4.1.2 Perpendicular Localization ........................................................................................ 29
  4.1.3 The Mobile Trajectory ............................................................................................... 31
  4.1.4 Analysis and Evaluation ............................................................................................. 35

4.2 GreenOrbs-based Observations on RSSI and Range Measurements ......................... 38
  4.2.1 Anisotropic problem .................................................................................................. 38
  4.2.2 Irregularity of RSSI ................................................................................................... 39
  4.2.3 Asymmetry and Dynamics of RSSI ......................................................................... 40
4.2.4 Errors of RSSI-based range measurements

4.3 CDL: A Combined and Differentiated Localization Approach ........................................ 41
4.3.1 Virtual-Hop Localization ............................................................................................. 42
4.3.2 Local Filtration ............................................................................................................. 45
4.3.3 Ranging-Quality Aware Calibration ............................................................................ 50

4.4 Evaluation ..................................................................................................................... 52
4.4.1 Experiments ................................................................................................................. 53
4.4.2 Simulation .................................................................................................................... 58

5 FORWARDING-QUALITY-AWARE DATA COLLECTION .............................................. 63
5.1 Observations on Packet Losses in WSNs ..................................................................... 65
5.1.1 Basic Observations ..................................................................................................... 65
5.1.2 Anatomy of Packet Loss .............................................................................................. 66

5.2 QoF: Comprehensive Path Quality Measurement ......................................................... 69
5.2.1 Generic Link Model ..................................................................................................... 69
5.2.2 QoF of a Generic Link ................................................................................................. 70
5.2.3 QoF of a Forwarding Path ............................................................................................ 71

5.3 Implementation ............................................................................................................. 73
5.3.1 Measuring PDR within a Node .................................................................................... 74
5.3.2 Measuring PDR over a Link ........................................................................................ 75
5.3.3 QoF Calculation ........................................................................................................... 75
5.3.4 Collaborative Reporting............................................................................................... 76

5.4 Evaluation ..................................................................................................................... 76
5.4.1 Methodology ................................................................................................................ 77
5.4.2 Performance Comparison ............................................................................................ 77
5.4.3 Observations on QoF ................................................................................................... 82
5.4.4 Summary of Results ..................................................................................................... 83

6 CANOPY CLOSURE ESTIMATES WITH GREENORBS ........................................... 84
6.1 Application Background ............................................................................................. 85
6.2 Theoretical Foundation ............................................................................................... 86

6.3 Design with GreenOrbs ............................................................................................... 88
6.3.1 Pre-deployment Training ......................................................................................... 89
6.3.2 Online Data Processing ............................................................................................. 93
6.3.3 Sink-side Data Translation ......................................................................................... 96

6.4 Evaluation ..................................................................................................................... 99
6.4.1 Effectiveness of Design ............................................................................................ 99
6.4.2 Estimates of Canopy Closure ............................................................................... 101

7 CONCLUSIONS ............................................................................................................ 107
7.1 Summary .................................................................................................................... 107
7.2 Future Work............................................................................................................... 108
8 REFERENCES ........................................................................................................... 110
1 Introduction

A wireless sensor network (WSN) is a self-organized wireless network that consists of a number of sensor nodes [1]. With the recent advances in wireless communication and microelectronic technologies, various low-power and cost-effective sensor motes have been developed and produced. Those progresses have enabled WSNs to better interact with the physical world and serve people by automatically monitoring the environments. WSN have been spread across a variety of application fields, such as environment surveillance [2, 3], object tracking [4], scientific observation [5], and medical care, etc.

Figure 1-1: A TelosB mote produced by our research group

The operation of a WSN mainly includes three aspects: sensing, computation, and communication. The sensor nodes, equipped with application-specific sensors, provide low-cost sensing in the physical environments. The sensor data are cached or stored on the local storage, and processed by the microprocessor embedded on the sensor node. The wireless communication module, which is usually made up of a low-power radio and an internal or external antenna, is in charge of encapsulating, transmitting, and receiving data packets.

Figure 1-1 shows a TelosB mote [6] produced by our research group. It includes two types of sensors, namely the temperature and humidity sensor and the illuminance sensor. The embedded microprocessor is MSP430 F1611, which is on the bottom side of the mote and thus not shown in the picture. The communication module includes CC2420 radio and an on-board antenna.

One or more sinks (also called base stations) play the roles of data assembling points and administrative interface for external users. Since the targeted application scenarios usually require deploying a large-scale network of numerous sensor nodes, it is a general practice for
WSNs to implement multi-hop transmission. The intermediate nodes not only transmit their own data, but also forward the data from the other nodes to the base station.

1.1 Research Background

To enable sustainable and reliable operation of WSN systems in practical applications, there are many issues that need to be taken into account. I summarize them into several aspects as follows.

(1) Energy efficiency

As opposed to the traditional wired networks and wireless local area networks, sensor nodes are usually equipped with limited energy supplies [6, 7]. Hence, a fundamental issue to be addressed in a practical WSN system is save energy consumption while in operation. Energy efficiency becomes especially important for WSNs deployed in remote and complicate environments, where human beings seldom go in general and replacing/recharging batteries are therefore difficult. Duty cycling [8] is a general practice to address the energy constraint, while there are a large number of existing designs at all layers of WSNs to address the energy efficiency problem [9-16].

(2) Infrastructural elements

Due to the need of low manufacture costs, operational efficiency, and deployment flexibility, sensor nodes are designed and developed to be resource constrained in both hardware and software functionalities, which result in limited infrastructural support. Specifically, in many application scenarios, infrastructural issues such as deployment methods, localization of sensor nodes, time synchronizations, and etc [17-20] are left to study. Driven by the application requirements, it is of essential importance to design protocols, techniques, and mechanisms to obtain the infrastructural information in an efficient manner.

(3) Wireless communication

Due to the impact of environmental factors, wireless communication is much less reliable than wired communication, resulting in relatively unstable quality of communication. Moreover, WSNs typical adopt energy-saving communication standards, such as IEEE 802.15.4 which has a maximum bandwidth of 250Kbps. If the potential interference caused by simultaneous transmissions is taken into account [9, 21], the achievable communication bandwidth will be even lower than originally rated. In short, WSNs demand efficient, reliable, and resilient communication protocols to maintain the network capacity at a reasonable level for meeting the application goals. The issues left to explore mainly include link estimation
(4) Data processing

By nature interacting with the dynamic physical world, the data generated from WSNs appear to be dynamic, evolutinal, and often spatially or temporally correlated. Given the constraints in energy and wireless communication quality, data processing in WSNs generally take measures to compress, suppress, or aggregate data during the data collection process. By exploiting the temporal or spatial correlations among the data of different sensor nodes, one can further reduce the traffic overhead and energy cost by executing coordinated sensing and data fusion. Meanwhile, it is important yet challenging in the above processes to preserve the accuracy and integrity of the processed data [31-33]. It is also worth mentioning that efficient data processing methods of WSNs are often application-specific.

(5) System reliability, performance consistency, and network manageability

Due to the cost-saving manufacture, limited hardware and software functionalities, and various environmental influences, sensor nodes are intrinsically prone to errors, faults, and failures. The sensor readings are often inaccurate with undesired errors. The computations sometimes yield faulty results. The sensor nodes and the wireless communication among them often face unexpected failures. All those problems exhibit considerable challenges to the system reliability, performance consistency, and network manageability of WSNs. How to maintain such performance factors at a satisfactory level against the unreliable dynamic deployment context remains a challenging issue to study in the practical WSN applications. Existing work in this aspect include designs at different layers, such as error-management in localization and synchronization [34, 35], outlier detection of sensor data [36, 37], faulty node detection and recovery [38, 39], program debugging [40, 41], and network diagnosis [42], etc.

1.2 Motivation

Environmental surveillance is a representative class of applications of WSNs. Under the circumstances of global climate changes and environmental pollution, forest ecosystem management is attracting increasing attention from human beings.

In order to make full use of the capabilities of forest in protecting the environment, maintaining the ecological balance, mitigating climate changes, and avoiding natural disasters, forest research and ecosystem management undertake a series of significant missions. First, we need to continuously monitor the forest, including the creatures and the microclimate, at large scale for long term, so that we can obtain comprehensive understanding of the evolution
of forest ecosystems and sufficient knowledge for forestry research. Second, we need to accurately and precisely measure the forest with respect to various forestry and biological factors (e.g. canopy closure and soil respiration), so that we can get quantitative scientific information to support efficient ecosystem planning and sustainable development. Third, we need to surveil the forest with respect to geological and environmental factors, so as to predict and avoid potential natural disasters (e.g. mud-rock flows and flood).

Forestry is a classical science with a longer history than that of computer science. Traditional forestry approaches to accomplish the above missions generally involve too much human intervention, adopt costly devices, but often support only small-scale measurements at selected areas. Those approaches often fail to satisfy the goals of forestry application, which usually require long-term large-scale measurements in a huge area with diverse creatures. As a consequence, traditional approaches often refer to indirect empirical models to provide estimated results at large scale. Such results are inaccurate, not scalable, and easily affected by the surveyors’ subjectivity.

WSNs offer inexpensive sensing of the physical world with a number of sensor nodes. The deployments can be large-scale, covering sufficiently large areas of interest. The sensor nodes typically adopt battery power supply and can be designed to sustain for several months and even years. More importantly, the sensing and data collection from a WSN are fully self-organized and automatic, which do not require much human intervention. Surveillance and measurements of forest using WSNs are thus convenient, cost-efficient, and powerful.

Though WSNs have the abovementioned advantages, they still face many critical challenges in supporting forestry applications. The research and engineering efforts to address those challenges motive this work.

- **Accurate localization with noisy range measurements.**

The complex forest terrain and dynamic environments make it more difficult to realize infrastructural support for WSNs. A typical challenge is to determine the geographical locations of the sensor nodes. GPS devices [43] are widely employed to obtain the global coordinates for civil applications. In the forest, GPS devices do not always work. Many areas in the forest are covered by dense tree canopy, where a GPS device cannot stably receive the satellite signals and thus fails to locate itself. Another reason that GPS does not work is GPS-devices are relatively expensive and large in size, increasing the cost to equip and enclose a sensor node.

Due to the above mentioned reasons, WSNs mostly adopt in-network localization. A certain portion of nodes who know their coordinates by nature are regarded as the anchors. By
using certain range measurement techniques and geometric methods, the other sensor nodes can then be located. It is worth noticing that such localization approaches highly rely on accurate range measurements among the nodes to work. Many high-quality ranging techniques cannot be applied in the forest WSNs, due to either the prohibitive cost or the inapplicability in an environment with large amount of obstacles. RSSI (received signal strength indication) [44] among the nodes is an affordable measure of node distances in the forest. But the wireless signals also appear to be quite noisy (i.e. with remarkable errors), irregular, and dynamic, producing additional challenges to realize accurate WSN localization.

- **Efficient data collection via unstable wireless communication among faulty nodes.**

Due to the constraints in energy budget and communication bandwidths, data collection in WSNs must be efficient with respect to energy consumption and traffic overhead. On the other hand, the quality of wireless communication is relatively unstable in the complex forest environment. The external interference further degrades the performance stability of sensor nodes. As a result, packet losses are observed to be a familiar phenomenon in such a context.

We have witnessed the current practice in WSNs, namely making routing decisions based on proper link estimation and exerting retransmissions to improve delivery ratio of data packets. Such countermeasures are effective, however, at the cost of extra energy consumption and traffic overhead. Thus a crucial issue left to study is how to make the data collection as efficient as possible while keep the associated costs at a reasonable level.

- **Accurate measurements against error-prone sensor readings in dynamic environments.**

When WSNs are applied to provide quantitative measurements of forestry and biological factors, they have intrinsic advantages, such as large-scale, continuous, synchronous, inexpensive, and fine grained, etc, compared to the traditional forestry approaches. Nevertheless, they also have potential limitations. Restricted by the hardware and software capabilities, the sensors usually yield noisy readings, which might exhibit higher dynamics and less consistency in the forest environments. How to ensure the accuracy and precision of WSN-based measurements is a significant issue to be studied.

Moreover, during the process of long-term measurements, the forest vegetation, the weather, and the surrounding environments are all likely to change along with time. In order to obtain more accurate measurement results, the autonomous sensor nodes should intelligently complete the sensing tasks in a manner that is adapted to the evolution of forest and changes in surroundings. While this is a non-trivial task that more or less requires human supervision, the appropriate solutions should make full use of domain knowledge, e.g. the
computation models in forestry. Many issues are left to study in the interdisciplinary field of forestry and computer science.

1.3 Contributions

In order to address the aforementioned challenges in WSNs for forestry applications, we have launched the GreenOrbs project [45], which aims at deploying a long-term large-scale WSN in the forest to support a series of forestry applications. As of the time this dissertation is completed, we have implemented GreenOrbs with two outdoor deployments and one indoor test-bed. The system scale is up to 330 nodes. The most lasting deployment has been in continuous operation for one year and is still in operation. Based on GreenOrbs, my research work mainly focuses on the following three aspects.

First, I address the issue of WSN localization using RSSI among the sensor nodes. As preliminary attempts, I design a mobile-assisted localization scheme called PI [46], which has better accuracy and low overhead. I design the optimal trajectory of the mobile beacon in PI and theoretically prove its correctness. Moreover, we have implemented a prototype of PI with 100 sensors, and evaluate its performance in real environments, including indoor and outdoor spots. In order to further improve the applicability and accuracy of localization in the forest, I propose CDL, a Combined and Differentiated Localization approach. CDL inherits the advantages of both range-free and range-based methods, and keeps pursuing better ranging quality throughout the localization process. In CDL, I propose a range-free scheme called virtual-hop localization, which carefully examines the local connectivity to mitigate the anisotropic problem. Using virtual-hop, the initial estimated node locations are more accurate than those output by other range-free schemes. For better ranging quality, I devise two local filtration techniques, namely neighborhood hop-count matching and neighborhood sequence matching. The filtered good nodes maintain a high standard of location accuracy. Using the good nodes to calibrate the bad ones, I employ the technique of weighted robust estimation to respect the contributions of the best range measurements, eliminate the interfering outliers, and suppress the impact of the middle class. CDL is implemented with GreenOrbs and demonstrated to be more accurate and efficient than the existing localization approaches.

Second, based on the real-world experience from GreenOrbs, I reveal the limitations of existing link-based estimation methods like ETX [47] and ETF [23] in measuring the forwarding quality of intact paths. In a practical system, routing decisions based on ETX often result in severely degraded network performance. Thus I present QoF, a new metric which
estimates the chances for a packet to be successfully forwarded, through either a link or a node. The link-QoF not only considers the transmission cost at the sender but also considers the data delivery ratio at the receiver. The node-QoF estimates the quality of forwarding within a node, and it plays an important role in identifying the nodes that drop packets. I have implemented QoF based on TinyOS 2.1 and incorporate it in the current CTP [48] implementation. I have evaluated the performance of QoF-based routing in the GreenOrbs test-bed with 50 TelosB nodes. The results demonstrate that using QoF as the metric of routing path quality enhances the network throughput and reduces the expected cost of a successful packet delivery.

Third, based on the GreenOrbs, I conduct study on introducing WSN technology into the field of forestry measurements. I propose the design and implement the system for canopy closure estimates [49]. The proposed design enables accurate canopy closure estimates using inexpensive sensors randomly deployed in the forest. In addressing the error-prone sensor readings, I propose techniques to effectively calibrate sensor readings and to discriminate the node states between light versus shade. Moreover, I design light-weight mechanisms for node state monitoring, which reduces communication overhead. The proposed designs are evaluated with GreenOrbs and compared with conventional forestry methods. The results demonstrate the advantages of WSN techniques and their great potential benefits by introducing WSNs into the traditional forestry field.

1.4 Dissertation Organization

The rest of this dissertation is organized as follows.

In Chapter 2, I first present an extensive review of the existing sensor network systems, including both application-driven systems and test-beds. Then I briefly summarize the related work in localization, link estimation, and data collection of WSNs.

Chapter 3 introduces GreenOrbs, the system foundation of my research work. I present the project motivation, system design, and the deployments we have carried out.

Chapter 4 presents the design of error-resilient localization. In that chapter I first introduce my preliminary attempts, namely the mobile-assisted perpendicular localization, and then present a combined and differentiated localization approach based on the real-world experience from GreenOrbs.

Chapter 5 elaborates the design of a forwarding-quality-aware data collection protocol called QoF. Chapter 6 presents the design with GreenOrbs to support a real forestry
application: canopy closure estimates. Chapter 7 concludes this dissertation and discusses the future work.
2 Related Work

2.1 Existing Sensor Network Systems

Since the concept of WSN is propose a decade ago, there have been a number of sensor network systems and deployments. According to their functions, I basically classify them into two categories: application-driven systems and test-beds.

2.1.1 Application-Driven Systems

The first well-known WSN deployment is carried out by Alan Mainwaring et al in 2002 [3]. They deploy a network of 32 MICA motes on the Great Duck Island for habitat monitoring. The Mica motes run on a pair of AA batteries and are equipped with Mica Weather Boards, which are able to provide various sensor data, such as temperature, light, barometric pressure, and humidity, etc. The sensor nodes are deployed near the nests of petrels and keep monitoring the storm petrel’s breeding season. They have been in operation for four weeks as of being reported. As one of the earliest real WSN deployments, it opens out a series of significant research issues in WSNs, such as duty cycling mechanism, data sampling and collection, routing and communication protocols, network retasking, localization, time synchronization, and self configuration, etc.

Berkeley’s Smart Dust project [50], led by Professors Pister and Kahn in 2001, is the early effort on sensor systems, which explores the limits on size and power consumption of autonomous sensor nodes. The science/engineering goal of the Smart Dust project is to demonstrate that a complete sensor/communication system can be integrated into a cubic millimeter package. This involves both evolutionary and revolutionary advances in miniaturization, integration, and energy management. The project is funded by DARPA, so they demonstrate Smart Dust with one or more applications of military relevance. In addition, they are pursuing several different applications with commercial importance, and they have got a long list of applications to work on. This project promises a bright future for wireless sensor networks, but until now the researchers still struggling in the sensor systems.

Pei Zhang et al report a ZebraNet system [51], which use GPS to record fine-grained position data in order to track long term animal migration in central Kenya, 2004. The ZebraNet hardware is composed of a 16-bit TI microcontroller, a 4 Mbits of off-chip flash memory, a 900 MHz radio, and a low-power GPS chip. The ZebraNet system is sustainable because of its renewable energy supply (solar battery) and environmentally-hardened
enclosure. However, it uses an unscalable TDMA MAC with time-synchronized and statically assigned timeslots. ZebraNet is also limited in its flexibility since occasionally-connected, mobile nodes cannot be easily reprogrammed.

Gilman Tolle et al report a wireless sensor network for monitoring the microclimate of a redwood tree in 2004 [52]. The experiment contains a vertical Mica2Dot deployment along the trunk every 2 meters from 15m above ground level to 70m above ground level in a 44 days’ time. The system collects three types of data, which is air temperature, relative humidity, and photosynthetically active solar radiation. They measure the system on several aspects of network functions and obtain some interesting findings according to their analysis, e.g., strongly bimodal distributions can be found in the log files, etc. It is of significance to the monitoring deployment experiences. Although these findings are important, the measurements at this scale, usually tens of sensors, can hardly reveal some network behaviors, such as routing dynamics and topology evolution, which exist only in large-scale sensor networks. Another similar sensor network system for environmental research, called Luster [53], uses a hierarchical architecture that includes distributed reliable storage, delay-tolerant networking, and deployment time validation techniques.

Geoff Werner-Allen et al [54] present a science-centric evaluation of a 19-day sensor network deployment at an active volcano in Ecuador. 16 TMote Sky sensors with D cell batteries, continuously sample seismic and acoustic data and initiate reliable data transmission to the base station at 100Hz. During the deployment time, totally 229 earthquakes, eruptions, and other seismacoustic events have been detected. The volcano project is carried out from 2004; experience 3 major deployments, all the data of the system collected is used for geographic monitoring and academic research. The authors use the data fidelity and yield as the main metric to evaluate the sensor network and present lessons learned from the volcano deployment, e.g. ground truth and self-validation mechanism is critical, the infrastructure and protocol may failure unexpected, which could give some help on others’ system deployment.

The ability of deploy unmanned surveillance missions, by using wireless sensor networks, is of great practical importance for the military. VigilNet [55] is one of the major efforts in the sensor network community to build an integrated sensor network system for surveillance missions. The focus of this effort is to acquire and verify information about enemy capabilities and positions of hostile targets. The group deployed 70 MICA2 motes [7], along a 280 feet long perimeter in a grassy field that would typically represent a critical choke point or passageway to be monitored. Each of the motes runs on a pair of AA batteries and is equipped with a sensor board that has magnetic, acoustic, motion, and photo sensors on it.
Time synchronization and location are important for a surveillance application because the collaborative detection and tracking process relies on the spatio-temporal correlation detection and tracking reports sent by multiple motes. In their prototype system, they design and implement the walking GPS solution, which assigns motes their location at the time they are deployed. Once the technique is mature enough, static configuration can be replaced with dynamic location schemes.

SensorScope [56] is a generation of measurement systems with a built-in capacity to produce high-density spatio-temporal measure. The solution is based on multiple wireless stations that auto-organize into a smart spatial entity, monitoring its environment. This innovative system is composed of multiple solar-powered sensing stations. Each sensing stations includes several external sensors, which make the station capable of measuring nine different inputs: ambient temperature and humidity, infrared surface temperature, solar radiation, wind speed and direction, precipitation, soil moisture, and soil pressure. The tests and deployments are carried out in different places. One of their recent deployments is carried out for Patrouille des Glaciers, an exceptional race. Weather monitoring before and during the race is of extreme importance for the safety of the participants. They deployed 10 stations in the most emblematic along the itinerary, and these stations were installed one week before the race. Organizers and participants were able to follow in real-time the weather conditions before and during the race through a dedicated web interface.

FloodSafe Honduras is an application of river flood prediction, carried out by several MIT researchers in 2004 [57]. The system uses only a small number of nodes to cover basins of 1000-10000km² using a unique heterogeneous communication structure to provide real-time sensed data, incorporation self-monitoring for failure, and adapting measurement schedules to capture events of interest. They build a sensor network for flood prediction that consists of 9 nodes and deploy the system on the Aguan River basin in northern Honduras. This installation was used to test the sensing, networking, deployment, and maintenance insures in rural Honduras. They choose three sensor types, rainfall, air temperature, and water pressure, and get them connected. The charging circuit on the sensor board allows photovoltaic charging of lithium-polymer batteries. Predictive environmental sensor network provide complex engineering and system challenges. The system allows model-driven control, thereby optimizing the prediction capability of the system.

The largest wireless sensor network which installed for structural health monitoring purposes is carried by Sukun Kim et al [58]. They deployed 64 nodes (Mica2 and MicaZ) on the Golden Gate Bridge for structural health monitoring. The motes run on 4 lantern batteries
which power can vary between 6V and 12V. The sensor nodes were deployed in the main span and the tower and collecting ambient vibrations synchronously. The sampled data is collected reliably over a 46-hop network. Their goal was to determine the response of the structure to both ambient and extreme conditions and compare actual behavior to design predictions. As the largest WSN deployment for structural health monitoring, its contributions have great significance in WSNs, such as data transmission, routing and communication protocols, link quality, etc.

2.1.2 Sensor Network Test-beds

Trio, a large-scale WSN testbed was deployed by UC Berkeley during four months of 2005 [59]. The motivation behind this testbed was to evaluate robust multi-target tracking algorithms at scale. Trio is an outdoor sensor network deployment and consists of 557 solar-powered motes. The testbed comes an area of approximately 50,000 square meters. The Trio node is designed for long-lived operation with minimal physical maintenance. Trio itself is based on Telos and includes passive infrared motion sensors, magnetometer, and microphone.

The Trio nodes reside in the lowest tier (Tier-1) of the architecture. Trio provides support for application level experimentation through a sensor suite optimized for detection and classification of humans and vehicles. The 557 Trio nodes in the testbed are organized into multiple routing trees, with each tree rooted at a gateway. Gateways (Tier-2) forward traffic between the 802.15.4 Trio network and an 802.11 wireless backbone network. A single root server sits in Tier-3 and connects to all of the gateways.

MoteLab [60] is an experimental wireless sensor network deployed in Maxwell Dworkin laboratory, the Electrical Engineering and Computer Science building at Harvard University. Motelab deploys 190 TMote Sky sensor motes for research in sensor network programming environments, communication protocols, system design, and applications. Each note includes sensors for light, temperature and humidity. The mote is powered from wall power (rather than batteries) and is connected to the departmental Ethernet. MoteLab is a free testbed and show their topology in the web. You can see the situation of any node. MoteLab provides a public, permanent testbed for development and testing of sensor network applications via an intuitive web-based interface. You can use the MoteLab web interface to upload your program to the building-wide network.
The Kansei project [61] is headed by Anish Arora at the Ohio State University and has been developing since spring 2004. Kansei is a Heterogeneous testbed of 265 Nodes. It’s hardware infrastructure comprises three arrays of sensor nodes: stationary, portable, and mobile. Its sensors include a photocell, a passive infrared, a temperature, and a magnetometer sensor, as well as a microphone. Kansei sensing testbed takes into account both sensing and environmental characteristics, as well as networking and platform properties. Kansei built an indoor testbed for larger experiments and deployed smaller test-beds in outdoor environments similar to those of the intended field.

In December 2004, the OSU DARPA-NEST team headed by Anish Arora completed the first demonstration and experiments of ExScal [62]. Meanwhile, ExScal’s demonstration is also the largest ad hoc 802.11 network thus far created. They deployed a 1000+ XSMs (eXtreme Scale Mote) [63] wireless sensor network and a 200+ XSSs (eXtreme Scale Stargate) [64] peer-to-peer ad hoc network of 802.11 devices in a 1.3km by 300m remote area in Florida, USA. The XSM’s lifetime approaches 1,000 hours of continuous operation on two AA alkaline batteries and XSS run on a lead-acid battery. ExScal uses a planned topology for node placement so as to efficiently cover the protected region. Its concept of operation is to deploy a dense wireless sensor network “tripwire” that detects, tracks, and classifies multiple intruders of different types in a long perimeter region. As one of the greatest project in WSNs, ExScal has provided us with a rich set of experiences and has given us a visceral understanding of the "real" problems that are posed by networks of extreme scale.

2.2 Localization

2.2.1 Range-Based Approaches

Range-based approaches assume that sensor nodes are able to measure the distance and/or the relative directions of neighbor nodes. Various techniques are employed to measure the physical distance. For examples, Time of Arrival (TOA) obtains range information via signal propagation times [65], and Time Difference of Arrival (TDOA) estimates the node locations by utilizing the time differences among signals received from multiple senders [66, 67]. As an extension of TOA and TDOA, Angle of Arrival (AOA) allows nodes to estimate the relative directions between neighbors by setting an antenna array for each node [68]. All those approaches require expensive hardware.
Received signal strength indicator (RSSI) is utilized to estimate the distance between two nodes with ordinary hardware [69, 70]. Various theoretical or empirical models of radio signal propagation have been constructed to map absolute RSSI values into estimated distances [71]. The accuracy and precision of such models, however, are far from perfect. Factors like multi-path fading and background interference often result in inaccurate range estimations [70, 72].

Recently, mobile-assisted localization approaches are proposed to improve the efficiency of range-based approaches [73-75]. The location of a sensor node can be calculated with the range measurements from the mobile beacon to itself, so no interaction is required between nodes, avoiding cumulative errors of coordinate calculations and unnecessary communication overhead. The localization accuracy can also be improved via multiple measurements obtained when the mobile beacons are at different positions.

### 2.2.2 Range-Free Approaches

Knowing the hardware limitations and energy constraints required by range-based approaches, researchers propose range-free solutions as cost-effective alternatives.

Having no distances among nodes, range-free approaches depend on the connectivity measurements from sensor nodes to a high density of seeds. For example, in Centroid [76], seeds beacon their positions to their neighbor nodes that record all received beacons. Each node estimates its location by calculating the center of all seeds it hears. In APIT [77], each node estimates whether it resides inside or outside several triangular regions bounded by the seeds it hears, and refines the computed location by overlapping the regions the sensors likely reside in. SeRLoc [78] employs a similar approach while emphasizes a secure mechanism against malicious attacks. As an alternate solution, DV-HOP only makes use of constant number of seeds [79]. Instead of single hop broadcasts, seeds flood their locations throughout the network, maintaining a running hop-count at each node along the path. Nodes calculate their positions based on the received seed locations, the hop-counts from the corresponding anchors, and the average-distance per hop through trilateration.

### 2.2.3 Error Control and Management

More recent proposals mainly focus on the issue of error control and management [80-84]. J. Liu et al [34] propose iterative localization with error management. Only a portion of nodes are selected into localization, based on their relative contribution to the localization accuracy,
so as to avoid error accumulation during the iterations. Similarly, H.T. Kung et al [85] propose to assign different weights to range measurements with different nodes and adopt robust statistical technique to tolerate outliers of range measurements. Nevertheless, SISR does not clearly examine the location accuracy of every node being located. Using ranging error as the only metric for weight assignments, it indeed prefers the situations where the majority of range measurements are accurate. A range-free approach beyond connectivity is proposed [86], which identifies location outliers by contrasting Euclidean proximity among the nodes. The noisy and outlier range measurement can also be sifted by utilizing the topological properties of a network [87].

2.3 Link Estimation and Data Collection

2.3.1 Link Estimation Methods

The quality of packet forwarding is a fundamental factor in sensor networks, which has been studied in a number of works. Such a factor can be estimated on different dimensions over the communication in between nodes, e.g., radio signal strength (RSS), transmission delay, packet reception ratio (PRR), and etc. Those parameters are measured at different layers across the communication stacks.

At the physical layer, RSSI and LQI are two most widely used parameters that describe the communicational quality between nodes. RSSI is measured on top of the received signal strength while LQI measures the chip error rate for successfully received packets. Both two parameters reflect the physical quality of the wireless channel in between the nodes, though as suggested in [44], the two parameters are not adequate to represent the quality of packet forwarding over the link.

At the link layer, many other metrics have been proposed. ETX measures the expected number of transmissions for successfully delivering a packet over the link. Specifically, if we denote $df$ as the probability that a packet is successfully received by the receiver and $dr$ the reverse probability that the link ACK can be successfully received. The ETX value over a link is then calculated as $1/(df*dr)$. Intuitively, ETX implies an expected number of transmissions it takes for an acknowledged packet transmission between the sender and the receiver. Another metric ETF is designed for links of high asymmetry. Using ETX as the routing metric will exclude links which have good one-way link quality. ETF, however, suggests that though the reverse link quality is low, the ACK still has a high probability being received by
the sender (as the synchronous ACK usually takes a short time period). It is shown in [23] that, routing based on ETF can improve the performance in wireless networks with very asymmetric links. There are some other link layer metrics, such as ETT (Expected Transmission Time), competence [88], L-NT [24], ENT [89], end-to-end success rate (SR) [90], required number of packets [91], EDR [92], and etc. Among aforementioned link estimation metrics, ETX is the most widely used one. ETX has worked as the de facto link quality indicator and has been used for a variety of wireless protocols [93, 94]. Currently, ETX estimation has been integrated into MintRoute [26] and CTP [48], for reliable and efficient data collection in wireless sensor networks.

Besides those estimators at separate layers, 4-bit link estimator uses 4 bits to combine information at PHY layer, link layer, and network layer. It considers both link layer packet reception quality and network layer congestion information and uses PHY signal strength to do the first step link filtering.

Due to the constrained communication radius and wide deployment area, data in WSNs are often delivered in multi-hop manner to the destination, i.e., a packet needs to select a path to the destination. In order to remove ambiguity, I use the terms sender and receiver to denote the nodes at the two ends of a single link and the terms source and destination to denote the nodes at the two ends of a multi-hop path. There are different metrics proposed to characterize the forwarding quality of a path. Minimal hop count can be used to select a path. Obviously, this metric equally treats different links and does not take into consideration the instability of the low power links in WSNs. Using ETX metric maximizes the throughput for a single link. Therefore, summing up all link-ETX values along the path gives the path-ETX. As we found in our test-bed, however, the path-ETX is not always adequate in practical system because it only gives an incomplete description of the path quality. Other link-based metrics [24, 89, 95] overlook the node forwarding quality as well. Simply aggregating the estimated link qualities does not give comprehensive description of the intact path quality.

There are many works focusing on node’s capability and stability. For example, Schemid et al. [35] investigate the relation between the timer stability and the power and other environment factors such as temperature, humidity, the cut of the oscillator, and etc. Embedded systems depend a lot on the timer system. For an event driven embedded OS, such as TinyOS, a very common source of events are driven by timers. The timer instability largely renders the OS in-stable.

Queue length is another important factor that affects the forwarding quality of individual nodes. Backpressure routing uses local queue length information to select a node with the
largest positive differential backlog. It is designed to be throughput optimal. BCP [96] is a recent realization of backpressure routing protocol for sensor networks. While the queue length is an indicator for congestion, there are also a lot of other works for congestion control such as [97, 98]. My work is different from those since I not only consider packet loss due to congestion but also consider packet loss due to problematic nodes.

Previous path metrics mainly overlook the gap while nodes are processing data, i.e., during the period from the packet is passed from MAC layer to network layer to the packet is passed back from network layer to MAC layer. Little attention was paid to fill such a gap in existing work, resulting in incomplete path quality measurement.

2.3.1 Data Collection Protocols

Since most WSNs have only one or a few base stations, data collection generally takes the form of many-to-one routing. As a result, data collection is different with the other routing scenarios, such as one-to-one routing [99] and one-to-many diffusion [100]. Meanwhile, the design of a data collection protocol for WSNs is usually associated with the link estimation method, and involves considerations on building the routing structure [101], rate control [98, 102], congestion control [103], load balance [104], reliability [26, 105], and fairness [106], etc.

To the best of my knowledge, the most widely adopted data collection protocol is CTP (collection tree protocol) [48] in WSNs. CTP is a tree-based collection protocol. The sinks advertise themselves as the tree roots. Nodes form a set of routing trees to those roots. CTP is address-free in that a node does not send a packet to a particular root. Instead, it implicitly chooses a root by choosing a next hop. Nodes generate routes to roots using a routing gradient, based on the link estimation results provided by ETX [47]. Thus CTP needs an adaptive beaconsing mechanism to exchange necessary packets via the links so that the link quality can be estimated according to the history of packet delivery. Moreover, CTP respects the impact of data traffic in link estimation. In order to restrict the protocol overhead while well characterizing the link quality, the adaptive beaconsing extends Trickle [107] to time its routing beacons. Using Trickle, nodes can quickly discover of new neighbors and recover from failures, while at the same time prolong the beacon intervals when the network is relatively stable. This results in a nearly beaconless routing in stable and static networks without sacrificing the ability to be adaptive to the environmental changes or new node discovery.
The other earlier collection protocols, such as MultihopLQI [108] and MintRoute [26] adopt simpler link estimation methods and address the tradeoff between cost and responsiveness of routing. Other recent proposals, such as geographic routing [109] and GHT-based mechanisms [110], aim at scalable routing. While geographic routing requires nodes to know their locations, it is often a non-trivial task to obtain location information of sensor nodes. The applicability of geographic routing is thus limited.

A large body of existing WSN protocols focus on how to mitigate the congestion that occurs when simultaneous traffic collides with each other during multi-hop transmissions. For example, CODA [111] and Fusion [97] propose different solutions to mitigate congestion by using channel occupancy sampling or a combined scheme of backpressure and rate-limiting. IFRC [102] proposes to allocate a fair rate to every sender in the network, so that the in-network traffic distribution tends to be even among different areas. Y. Gu et al. in [92] study the issue of data forwarding in extremely low duty-cycle sensor networks with unreliable links. Dynamic Switch based Forwarding is proposed to optimize the network performance with regards to the expected data delivery ratio, the expected communication delay, or the expected energy consumption.
3 GreenOrbs: A Long-Term Large-Scale WSN in the Forest

3.1 Project Motivation

3.1.1 Research Background in the WSN field

There have been a number of well-known WSN systems and deployments. Figure 3-1 plots several typical examples and compare their system scales and deployment durations. The previous systems and deployments are colored blue. We can see none of them sustains long-term and large-scale deployment indeed. By far the largest deployment called ExScal [62] included more than 1000+ nodes, but it was kept in operation for only a few days. A SensorScope [56] deployment is the longest one among all the previous deployments, lasting for around 6 months, but the system scale is restricted to no more than 97 nodes. Trio [59] realizes a continuous deployment of 557 nodes for 4 months. It employs solar-powered sensor motes, however, which avoids the energy constraints on the system design.

![Figure 3-1: Comparison of GreenOrbs and existing systems](image)

Does WSNs scale and sustain for long?

What are the fundamental challenges in deploying long-term large-scale WSNs?

Bearing the above questions in mind, we have launched GreenOrbs, a collaborative research project to study long-term large-scale WSNs in the forest.
3.1.2 Forestry Applications

“The world has just ten years to bring greenhouse gas emissions under control before the damage they cause becomes irreversible.” This is a famous prediction raised by climate scientists and environmentalists recently. It reflects the increasing attention in the past decade from human beings on the earth’s environment and climate change. Specifically, forest, which is regarded as the earth’s lung, is a major force in the battle against global warming. Forest is a critical component in global carbon cycle. It is able to absorb 10%~30% of the carbon dioxide from industrial emissions. Moreover, it has large capacity of water conservation, preventing water and soil loss, and hence reducing the chance of nature disasters like mud-rock flows and floods.

Regarding the above points, forest management and surveillance become important missions nowadays. Forestry applications usually require long-term, continuous, synchronized monitoring of huge measurement areas with diverse creatures and complex terrains. As a result, they usually demand large quantities of human resource and equipments. The state-of-arts of forestry techniques, however, only offer small scale, asynchronous, discontinuous monitoring or coarse-grained measurements. For long-term large-scale applications, forestry application developers have to refer to indirect and imprecise inferential models.

WSNs have great potential in resolving the challenges in forestry. Under this circumstance, we have launched the GreenOrbs project, a long-term large-scale wireless sensor network system in the forest. GreenOrbs realizes all-year-round ecological surveillance in the forest, collecting various sensory data, such as temperature, humidity, illumination, and content of carbon dioxide. The collected information can be utilized to support various forestry applications, detailed as follows.

**Canopy closure estimates.** Canopy closure is defined as the percentage of ground area vertically shaded by over-head foliage. It is a widely-used significant forestry indicator but the traditional measurement techniques have either poor accuracy or prohibitive cost. Based on the readings of illuminance sensors and Monte Carlo Theory, GreenOrbs realizes accurate and efficient canopy closure estimates of vast forest. Using the similar method, another forestry indicator called Leaf Area Index can also be measured by GreenOrbs with sensors deployed in the three-dimensional space.

**Fire risk prediction.** Traditional techniques provide only inaccurate forecast according to the macroscopic weather conditions, such as temperature, humidity, and wind force. The real
fire risks, however, are indeed closely related to the microscopic ground condition and human activities. Using the data of sensors in the forest, namely temperature and humidity, GreenOrbs is able to monitor the local environmental factors. Those data act as important input elements of fine-grained real-time fire risk prediction.

Study on biodiversity. The sensor readings of temperature, humidity, and illuminance precisely characterize the microclimate in the forest. On the other hand, research on biodiversity, e.g. multispecies competition, highly relies on the observations on microclimate. Compared to the traditional forestry approaches, GreenOrbs have apparent advantages. The surveillance can be long-term and continuous, while the measurement is generally inexpensive and fine-grained.

Carbon sequestration. It is well-known nowadays forest is significant in the mitigation of global warming. To maximize the utility of forest carbon sequestration, the capacity of carbon sequestration of different tree species need to be accurately measured. This can be realized with carbon dioxide sensors in the three-dimensional forest space. By comparing the sensor readings at different heights, the amount of carbon dioxide a tree canopy absorbs can be continuously monitored.

3.2 System Design Overview

3.2.1 Building Blocks

Hardware. GreenOrbs employs the TelosB mote with a MSP430 processor and CC2420 transceiver. On a sensor node, the program flash memory is 48K bytes. The measurement serial flash is 1024K bytes. The RAM is 10K bytes.

Since GreenOrbs deployments are carried out in the wild forest, enclosure of sensor nodes is extremely important to protect the nodes from being corrupted by possible inclement weather or destroyed by physical destruction (e.g. knocks from wild animals). During its development process, GreenOrbs has experienced five versions of the enclosure design. The current design is shown in Figure 3-2.

Figure 3-2. The current enclosure of a GreenOrbs node
A typical GreenOrbs node is equipped with four sensors, providing five types of sensor readings. Table 3-1 lists the functions and supporting software components of the sensors. The manufactory cost of a GreenOrbs node, including the deployment and maintenance cost, is 50 dollars.

Table 3-1: Sensors on a GreenOrbs node

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Function</th>
<th>*Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensirion Sht11</td>
<td>Temperature Humidity</td>
<td>SensirionSht11C</td>
</tr>
<tr>
<td>Hamamatsu S1087</td>
<td>Illuminance</td>
<td>HamamatsuS1087ParC</td>
</tr>
<tr>
<td>Internal Voltage Sensor</td>
<td>MCU-Internal Voltage</td>
<td>VoltageC</td>
</tr>
<tr>
<td>GE Telaire 6004</td>
<td>Content of CO2</td>
<td>Self-developed</td>
</tr>
</tbody>
</table>

*We develop the supporting software component of Telaire 6004 by ourselves. The supporting software components of the other three sensors are based on the software stack in TinyOS 2.1.*

Software and Protocols. In order to enable sustainable sensing in the forest, in the early GreenOrbs deployments, we used to adopt the LowPowerListening interface to enable low power listening on the duty-cycled nodes. Later we notice the forestry applications usually require synchronous sensor readings. Thus we modified the duty cycling mechanism. In a globally synchronized manner, in every hour all the nodes wake up at the same time, then keep radio on for a certain period, and switch to sleep at the same time.

The software on the GreenOrbs nodes is developed on the basis of TinyOS 2.1. Figure 3-3 is the design diagram of the software modules.

Figure 3-3. The diagram of software modules
The GreenOrbs system mainly carries bi-directional data streams. The mainstream is multi-hop data collection from the ordinary nodes to the sink. The Data Collector component based on CTP [48] is employed for this purpose. The rest transmissions are the configuration packets sent from the sink to the ordinary nodes. Hence Configurator component based on DRIP [112] is devised to achieve efficient data disseminations. Meanwhile, the FTSP protocol [113] plays the functions of network-wide synchronization, so as to enable the globally synchronized duty cycles. The Logger component is in charge of data access (read and write) to the measurement serial flash. The Status Viewer component merges and encapsulates the data from the sensors, network, and flash, according to the preconfigured message formats. Such encapsulated messages are transmitted via the serial communication port.

### 3.2.2 System Settings

In the measurement period of GreenOrbs, we decrease the beaconing frequencies of CTP and FTSP to restrict the control overhead. The CTP protocol has been further tailored to the GreenOrbs system. The original CTP protocol triggers an update of routing table when encountering any retransmission. Such updates may cascade to the whole network, resulting in heavy traffic load. We have modified CTP and made the routing mechanism less sensitive to link failures. An update of routing table is triggered only when the number of retransmissions exceeds a predefined threshold.

As described in previous subsection, the DRIP-based Configurator enables us to regulate the operational parameters of the nodes without collecting them back. The DRIP protocol uses TrickleTimer [107]. When receiving a new version of data, Drip disseminates it at the minimum interval. Otherwise, it doubles the broadcasting interval until the maximum value. The maximum interval of TrickleTimer is 1024 seconds by default. Our observation shows such a long interval is often inefficient to ensure network-wide consistency. Thus in our latest implementations, the maximum interval of TrickleTimer is set at 20 seconds.

Table 3-2 lists all the configurable parameters in our implementation. Specifically, TinyOS 2.1 only provides an interface to set the transmission power of an individual packet. There is not an interface to configure the transmission power of all the outgoing packets (namely broadcast and unicast packets). Thus we hack the CC2420Transmit component and add a component named CC2420SetPower to enable online configuration of transmission power of CC2420.
Table 3-2: Configurable parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target ID</td>
<td>The ID of the node whose parameters need to be configured. 0xFFFF denotes a configuration applicable for all the nodes</td>
</tr>
<tr>
<td>Data rate</td>
<td>The frequency of a node to send data towards the sink.</td>
</tr>
<tr>
<td>Duty cycle</td>
<td>The ratio of the time on duty to that of sleep.</td>
</tr>
<tr>
<td>Cycle length</td>
<td>The length of a period that includes one time of duty and sleep.</td>
</tr>
<tr>
<td>Tx_Power</td>
<td>Transmission power of CC2420.</td>
</tr>
<tr>
<td>Application beaconing interval</td>
<td>Interval between application beacons</td>
</tr>
<tr>
<td>FTSP beaconing interval</td>
<td>Interval between FTSP beacons</td>
</tr>
<tr>
<td>Duty_Cycle_On</td>
<td>A Flag indicating whether the nodes are duty cycled.</td>
</tr>
<tr>
<td>Parent_Switch_Threshold</td>
<td>The threshold that triggers a parent switch using CTP</td>
</tr>
</tbody>
</table>

Table 3-3: GreenOrbs deployments

<table>
<thead>
<tr>
<th>Place</th>
<th>Area</th>
<th>Duration</th>
<th>Battery</th>
<th>Size</th>
<th>Network Diameter</th>
<th>Duty Cycle</th>
<th>Data Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>University woodland #1</td>
<td>20,000 m²</td>
<td>1 month (2008)</td>
<td>AA size, 800 mAh 1.5V</td>
<td>50 nodes</td>
<td>6 hops</td>
<td>No</td>
<td>15 Mbytes</td>
</tr>
<tr>
<td>University woodland #2</td>
<td>20,000 m²</td>
<td>10 months (2009)</td>
<td>AA size, 2200 mAh 1.2V</td>
<td>120 nodes</td>
<td>10 hops</td>
<td>5%</td>
<td>272 Mbytes</td>
</tr>
<tr>
<td>University woodland #2 and #3</td>
<td>40,000 m²</td>
<td>Ongoing (2009.12~)</td>
<td>D size, ~8000mAh, 1.5V</td>
<td>330 nodes</td>
<td>12 hops</td>
<td>8% or No</td>
<td>140 Mbytes</td>
</tr>
<tr>
<td>Tianmu Mountain</td>
<td>200,000 m²</td>
<td>1.5 months (2009)</td>
<td>D size, ~8000mAh, 1.5V</td>
<td>50 nodes</td>
<td>10 hops</td>
<td>5%</td>
<td>3 Mbytes</td>
</tr>
<tr>
<td>Tianmu Mountain</td>
<td>200,000 m²</td>
<td>Ongoing (2009.10~)</td>
<td>D size, ~8000mAh, 1.5V</td>
<td>200 nodes</td>
<td>~ 20 hops</td>
<td>5%</td>
<td>10 Mbytes</td>
</tr>
</tbody>
</table>
3.3 Real-World Deployments

The first GreenOrbs deployment was carried out in July 2008. Ever since then, GreenOrbs has experienced a number of deployments at different places, with different scales, and for different durations. The GreenOrbs website [45] includes brief information on the team members who have contributed to the design, implementation, and deployments of GreenOrbs.

Table 3-3 lists the information of all the GreenOrbs during the past two years. “No” in the column of “Duty Cycle” means the CC2420 radio is always powered on. Figure 3-4 shows the bird’s-eye picture of a GreenOrbs deployment in the campus woodland.
4 Error-Resilient Localization

The work in this chapter is motivated by the need of accurate location information in GreenOrbs [45, 49], a large-scale sensor network system in the forest. The location information of sensor nodes is an indispensable element in various GreenOrbs applications, such as fire risk evaluation, canopy closure estimates, microclimate observation, and search and rescue in the wild. The real-world experience from GreenOrbs reveals that localization in the wild remains very challenging, in spite of the substantive efforts existing in the literature. The non-uniform deployment of sensor nodes inevitably causes anisotropic problem with range-free localization. Nevertheless, for range-based localization, the received signal strength indications (RSSI) used for ranging are highly irregular, dynamic, and asymmetric between pairs of nodes. To make it even worse, the complex terrain and obstacles in the forest easily affect RSSI-based range measurements, thus incurring undesired but ubiquitous errors.

Ranging quality is the key that determines the overall localization accuracy. Bearing this point in mind, recent proposals focus more on error control and management. Some of those proposals enhance the localization accuracy by deliberately reducing the contribution of error-prone nodes to the localization process [34]. The rest schemes improve localization by identifying large ranging errors and outliers, relying on the topological or geometric properties of a network [86] [87].

Ranging quality indeed includes two aspects. One of them refers to the location accuracy of the reference nodes. The other concerns the accuracy of range measurements. The two aspects are equally important, with respect to the impact on the localization results. Most of the proposals, however, unilaterally address only one aspect, thus failing to achieve the desired localization accuracy.

To address the above challenges and limitations, I propose **CDL**, a Combined and Differentiated Localization approach. CDL inherits the advantages of both range-free and range-based methods, and keeps pursuing better ranging quality throughout the localization process. The contributions of the work in this chapter are summarized as follows.

1. I propose a range-free scheme called virtual-hop localization, which carefully examines the local connectivity to mitigate the anisotropic problem. Using virtual-hop, the initial estimated node locations are more accurate than those output by other range-free schemes.
2. For better ranging quality, I devise two local filtration techniques, namely neighborhood hop-count matching and neighborhood sequence matching. The filtered good nodes maintain a high standard of location accuracy.

3. Using the good nodes to calibrate the bad ones, I employ the technique of weighted robust estimation to respect the contributions of the best range measurements, eliminate the interfering outliers, and suppress the impact of the middle class.

4. I implement CDL with GreenOrbs and evaluate it with extensive experiments and simulations. The results demonstrate that CDL outperforms the existing approaches with high accuracy, efficiency, and consistency of performance.

4.1 Preliminary Attempts

Figure 4-1: Experimental observations of RSSI: (a) Outdoor deployment sketch; (b) Outdoor RSSI values; (c) Indoor deployment sketch; (d) Indoor RSSI values.
4.1.1 Monotonicity of RSSI

RSSI is initially used for power control in wireless networks. The existing signal propagation models of RSSI, however, are far from perfect, because of the uncertain influences such as background interference, non-uniform spreading, signal fading and reflections etc. To better understand RSSI patterns, I conduct some initial experiments with several TelosB motes, as illustrated in Figure 4-1(a).

In this set of experiment, node $A$ broadcasts signals, and the rest nodes receive RSSI values from their CC2420 transceivers. Node $A$ is moving from 10 meters from $O$ to 20, 30, 40, and 50 meters. All the measured RSSI values are shown in Figure 4-1 (b).

With the RSSI values from node $A$ to a node, in ideal case the distance between other nodes and node $A$ should be calculated according to the log-normal shadowing model in Equation (4-1), which is widely used in range-based localization approaches [66, 69].

$$RSSI = P_T - PL(d_0) - 10 \times \eta \times \log \left( \frac{d}{d_0} \right) + X_\sigma$$ (4-1)

where, $P_T$ is the transmission power and $PL(d_0)$ is the path loss for a reference distance of $d_0$, and $\eta$ is the path loss exponent. The random variation in RSSI is expressed as a Gaussian random variable of $X_\sigma = N(0, \sigma^2)$. All powers are in dBm and all distances are in meters. $\eta$ is set between 2 and 5. $\sigma$ is set between 4 and 10, depending on the specific environment [71].

<table>
<thead>
<tr>
<th>Table 4-1: Observation Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
</tr>
<tr>
<td>Estimated distance (m)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4-2: Indoor Observation Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
</tr>
<tr>
<td>Estimated distance (m)</td>
</tr>
</tbody>
</table>

The real and estimated distances between $A$ and $O$ are compared in Table 4-1. The average relative error is 9.06%.

Similar results are observed in an indoor environment. I conduct the second group of experiments in our laboratory. As illustrated in Fig. 4(a), node $A$ moves from 4 meters away from $O$ to 8, 12 and 16 meters. All the measured RSSI values are shown in Fig. 4(b). Table
4-1 lists the real distances between $A$ and $O$ and those estimated ones using Equation (4-1). The average relative error of estimation is 10.09%.

I conduct another group of experiments in an indoor environment and get the similar results as shown in Figures 4-1 (c), (d), and Table 4-2.

Interestingly I find that, the closer a node is to the signal sender, the larger RSSI value it perceives.

### 4.1.2 Perpendicular Localization

In the experiment, we see that when the mobile beacon moves along a straight line, the largest RSSI value received by a sensor node corresponds to a point on the line that is closest to the node. In theory, this point should be the projection of the node on the line. Given two different projections of the sensor node on the trajectory, this node can be located as the intersection point of two perpendiculars that cross the mobile beacon’s trajectory over the two projections, respectively.

In order to illustrate how PI works, I show an example in Figure 4-2, where, a mobile beacon traverses the region while broadcasting beacon packets periodically. A beacon packet contains the coordinates of the position of the mobile beacon. The solid black lines in Figure 4-2 form the trajectory of the mobile beacon, with the arrows denoting its moving directions. The mobile beacon (in red) starts at point $P_1$, turns its direction at point $P_2$, and stops at point $P_3$. By combining the trajectory with the virtual line $P_1P_3$, we obtain a virtual triangle $\triangle P_1P_2P_3$ (we call it VT from now on).

Let $R$ be the transmission range of mobile beacon. To ensure that all the nodes in a VT can receive the signals from the mobile beacon, the edges $P_1P_2$ and $P_2P_3$ must be shorter than $R$. Meanwhile, the angle $\theta$ between the two lines should satisfy $0 < \theta \leq \pi/3$.

Suppose the five nodes (in blue) in Figure 4-2 are located using PI. We use node $N(x, y)$ as an example. The mobile beacon starts at point $P_1$ and it broadcasts the start signal with its current location. Node $N$ records the start position when it hears the start signal. Along its trajectory from $P_1$ to $P_2$, the mobile beacon broadcasts beacon packets periodically with its current location. Node $N$ receives all the beacon packets, and records the one with the largest RSSI value. When the mobile beacon arrives at $P_2$, it broadcasts a stop signal with its current location. When node $N$ receives the stop packet, it knows that the mobile beacon has just finished traversing the line from $P_1$ to $P_2$. The recorded position is the position where the
beacon packet with largest RSSI value is broadcast. We label the recorded position as \(A(x', y')\).

According to the observational result in the previous subsection, line segment NA is the shortest one among all the line segments connecting node N and any point on line \(P_1P_2\). In other words, Node A is the projection of node N on line \(P_1P_2\). Hence line NA is perpendicular to line \(P_1P_2\), and we have:

\[
\frac{y_2 - y_1}{x_2 - x_1} \times \frac{y' - y'}{x - x'} = -1 \tag{4-2}
\]

![Figure 4-2. An example of PI scheme](image)

Similarly, when the mobile beacon moves from \(P_2\) to \(P_3\), another position \(B(x'', y'')\) is recorded which is the projection of node N on line \(P_2P_3\). Thus we have

\[
\frac{y_3 - y_2}{x_3 - x_2} \times \frac{y' - y''}{x - x''} = -1 \tag{4-3}
\]

By solving Formulas (2) and (3), we can compute the coordinates \((x, y)\) of node N,

\[
\begin{bmatrix}
  x \\
  y
\end{bmatrix} = \begin{bmatrix}
  x_2 - x_1 & y_2 - y_1 \\
  x_3 - x_2 & y_3 - y_2
\end{bmatrix}^{-1} M
\]

where

\[
M = \begin{bmatrix}
  x_2 - x_1 & y_2 - y_1 & 0 & 0 \\
  0 & 0 & x_3 - x_2 & y_3 - y_2
\end{bmatrix}
\]

In the above process, we do not use the absolute RSSI values, so as to avoid the errors brought by the translation of physical distance and RSSI.
4.1.3 The Mobile Trajectory

Clearly, a sensor node can be easily located when it is in the scope of a VT. When the whole deployment area of a sensor network cannot be covered by one VT, however, the trajectory of the mobile beacon to locate all the sensor nodes should be taken further considerations of. We require an optimal trajectory with the following characteristics:

It is able to locate all the sensor nodes;

It is the shortest trajectory so that the mobile beacon traverses the whole area in the shortest time. Moreover, the optimal trajectory consumes the minimum energy cost for the mobile beacon, when its velocity and broadcast frequency is fixed.

The elapsed time, from a node receiving the first beacon packet to determining its location, is minimized.

For ease of illumination, we define the optimal VT to be a VT that covers the largest area, given its perimeter.

Theorem 4-1: The optimal VT in PI is an equilateral triangle with the lengths of its sides all equal to R, where R is the transmission radius of the mobile beacon.

Proof: A VT $\triangle ABC$ with its inscribed circle O is shown in Figure 4-3. The radius of circle $O$ is $r$. Let $S$ be the acreage and $L$ be the perimeter of $\triangle ABC$.

We define the parameter $\lambda = S/L$, which represents the ratio of the acreage to the perimeter of a VT. Consequently, the optimal VT has the largest value of $\lambda$.

$$\lambda = \frac{S}{L} = \frac{1}{2} \times \frac{L \times r}{L} = \frac{r}{2} \quad (4-5)$$

Obviously, $\lambda$ reaches its maximum when the radius $r$ reaches its maximum. Note that $AF = AD$, $BD = BE$, $CE = CF$, because $F$, $D$ and $E$ are the intersection points of inscribed circle $O$ with the sides of $\triangle ABC$. We have:

$$L = 2 \times r \times \left( \cot \frac{A}{2} + \cot \frac{B}{2} + \cot \frac{C}{2} \right)$$
The equality in the above formula is valid iff.
\[ ctg \frac{A}{2} = ctg \frac{B}{2} = ctg \frac{C}{2} \]

which means angles $A$, $B$ and $C$ equal to each other. Thus $\triangle ABC$ is an equilateral triangle.

Furthermore, let $a$ be the side length of equilateral triangle $\triangle ABC$, then

\[
\lambda = \frac{r}{2} = \frac{\sqrt{a}}{12} \quad (4-6)
\]

Then $\lambda$ is proportional to $a$. From the previous subsection, we know $a \leq R$, which ensures all nodes in the VT can be located. Hence, $\lambda$ of the virtual triangle in PI reaches its maximum, when $a=R$. □

According to Theorem 4-1, we can conclude that the trajectory of mobile beacon is optimal, when it consists of multiple joint optimal VTs, as depicted in Figure 4-4.

If a node only receives two pairs of start and stop signals broadcast by the mobile beacon, it knows the three vertices of the VT and then locates itself. Nodes at special positions, however, might receive more than two pairs of start and stop signals and PI needs to deal with this situation.
Figure 4-4. A sensor network and its optimal trajectory of the mobile beacon

As illustrated in Figure 4-4, nodes $N_1$, $N_2$, $N_3$, $N_4$ represent four special cases, where $N_1$ can receive three pairs of start and stop signals when the mobile beacon traverses sides $P_1P_2$, $P_2P_3$ and $P_3P_4$. $N_2$ receives four pairs of signals when beacon traverses $P_3P_6$ and $P_6P_7$ of one VT and $P_8P_9$ and $P_9P_{10}$ of another VT. $N_3$ receives three pairs of signals when the mobile beacon traverses sides $P_8P_9$, $P_9P_{10}$, and $P_{10}P_{11}$. Node $N_4$ receives six pairs of signals when the mobile beacon traverses the six sides of four VTs $\triangle P_5P_6P_7$, $\triangle P_8P_9P_{10}$, $\triangle P_9P_{10}P_{11}$, and $\triangle P_{10}P_{11}P_{12}$.

Lemma 6-1 A node can receive at most 6 pairs of start and stop signals, during the whole localization process of PI.

It is straightforward to prove Lemma 6-1 by enumerating all the possible locations of a sensor node. Due to the page limit, I skip the proof.

If a node receives start and stop signals from all the three vertices of a VT, we call this VT a locating VT for the node. PI let each node compute the sum of RSSI values from the three vertices of a locating VT, and the locating VT whose vertices have the largest sum of RSSI values is used to calculate the node location.

We define $side(i)$ $(1 \leq i \leq 6)$ as the side from which the node receives the $i$th pair of start and stop signals. Let $cp(i)$ denote the point on $side(i)$ that is closest to the node. Variable $side$ presents the current side traversed by the mobile beacon. Variable $rssi$ and $position$ saves the current largest RSSI and its corresponding position of the beacon signals. Variable $loc$ saves
the location calculated from current locating VT. The final result of the node coordinates is kept with the variable of location. SumRSSI(l) is the function to calculate the sum of the three vertices of the node l’s location VT. The pseudo code of the main function on message processing in PI is shown in Figure 4-5.

PI can address all the special cases. For example, N₁ receives three pairs of start and stop signal, respectively from the locating VT of ΔP₁P₂P₃ and ΔP₂P₃P₄, so the corresponding location results by using these two locating VTs are the coordinates of points N₁’ and N₁. For a node at point N₁, ΔP₂P₃P₄ has larger sum of RSSI values than ΔP₁P₂P₃. Thus, the coordinates of point N₁ can be correctly selected in place of the coordinates of N₁’. Similarly, nodes N₂, N₃ and N₄ can determine their coordinates from multiple calculated results.

```
OnMessageReceived(Message m)
{
    if (m.flag=start)
    {
        //start of the side(i)
        side.clear(); //clear the variable of side(i-1)
        Record(side.start,m);
        //save RSSI and position of the first beacon signal on side(i)
        r=m.RSSI; p=m.position;
    }
    else if ((m.flag=beacon)and(m.RSSI¸rssi))
    {
        //larger RSSI,update current r and p
        r=m.RSSI; p=m.position;
    }
    else if (m.flag=stop)
    {
        //end of the side(i)
        Record(side.stop,m);
        side(i)=side; cp(i)=position;
        if (side(i-1).stop=side(i).start)
        {
            loc=calculate(side(i-1),side(i));
            // keep the one with larger SumRSSI() if two different locations are obtained
            if((loc!=location)and(SumRSSI(loc)
                        >SumRSSI(location)))
                location=loc;
        }
    }
}
```

Figure 4-5. PI Algorithm
4.1.4 Analysis and Evaluation

This section theoretically analyzes the performance on the estimation error, the trajectory length, the localization time, and miscellaneous overhead.

(1) Theoretical Estimation Error

Although the mobile beacon can keep moving in a continuous manner, the beacon packets have to be broadcast periodically, with a specified interval between every two consecutive packets. As a result, the beacon trace is chopped into a series of discrete beacon points. Hence there exists an error between the estimated location by PI and the real location.

Figure 4-6 shows an example in which $P_1$, $C_1$, $C_2$, $C_3$, $C_4$ and $P_3$ are beacon points. Given the velocity of the mobile beacon as $V$ and the broadcast frequency as $F$, the distance between two beacon points is $V/F$. With high probability, the closest points to node $N$ on sides $P_0P_1$ and $P_1P_2$ (i.e. the theoretical projections of node $N$ on the sides) lie between two beacon points. Accordingly, node $N$ stores the coordinates of $C_2$ instead of $A$ in its location computation, as the beacon packet broadcast at $C_2$ has the largest RSSI value. Similarly, $C_4$ is stored instead of $B$.

Suppose $|AC_2| = d_1$ and $|BC_4| = d_2$, the distance between the real location of node $N$ and the estimated location $N'$ is:

$$|NN'| = \sqrt{\frac{4}{3} \left( d_1^2 + \frac{d_2^2}{2} + d_1 d_2 \right)}$$

![Figure 4-6. VT with all beacon positions](image)

If $C_3$ is the closest to position $B$,

$$|NN'| = \sqrt{\frac{4}{3} \left( d_1^2 + \frac{d_2^2}{2} - d_1 d_2 \right)}$$

Thus the upper bound of theoretical estimation error ($TERR$) is given by:
\[
TERR_{\text{max}} = \frac{V}{F} \quad \text{when} \quad d_1 = d_2 = \frac{V}{2F}
\] (4-7)

Meanwhile, the theoretical mean estimation error is:
\[
VERR = \frac{V}{F} \times \frac{4}{\sqrt{3}} \times \left( \int_{-\sqrt{3}/2}^{\sqrt{3}/2} \int_{-1/4}^{1/4} \sqrt{x^2 + y^2 + xy} \, dx \, dy \right) + \int_{0}^{\sqrt{3}/2} \int_{-1/4}^{1/4} \sqrt{x^2 + y^2 - xy} \, dx \, dy = 0.8128 \frac{V}{F}
\] (4-8)

(2) Trajectory Length and Localization Latency

A rectangle deployment area with length \(L\) and width \(H\) is shown in Figure 4-4. The trajectory length of the mobile beacon labeled as \(LM\) is:
\[
LM = \left( \frac{L}{R} \right) \times 2R + R \times \left[ \frac{H}{\sqrt{3}/2} R \right] + \left[ \frac{H}{\sqrt{3}R} \right] \times \frac{\sqrt{3}R}{2}
\]

Furthermore, sensor locations are calculated when the mobile beacon moves. This process ends when the mobile beacon arrives at the terminal of the trajectory. The latency for locating all the sensors can be calculated by:
\[
T = \frac{LM}{V}
\] (4-9)

(3) Overhead

Communication Cost. The communication cost of a sensor node depends on the total number of beacon packets it receives. The number of beacon packets (\(NBP\)) received by a sensor node when the mobile beacon traverses one VT side is
\[
NBP = \frac{FR}{V}
\] (4-10)

Meanwhile, a sensor node receives the beacon packets from 6 sides at most according to Lemma 6-1. Therefore the upper bound of communication cost of a sensor node is \(6FR/V\).

Computation overhead. From Figure 4-4 we can see that the most frequent operation in PI is to compare the recorded RSSI value with the latest RSSI value on one side of VT. The number of beacon packets received by a sensor node on one VT side is \(FR/V\), and the sensor node can receive the beacon packets from 6 VT sides at most. Thus the computation overhead on a sensor node is \(O(FR/V)\).
Storage overhead. PI only stores two vertices and one point of maximum RSSI value for each VT side. The localization result is achieved based on the information from two joint sides. When a node has two possible location results, it compares the aggregate RSSI values of both locating VTs and only saves one after comparison, as illustrated in Figure 4-4. Therefore PI only needs to store 14 vertices with their corresponding RSSI values, which costs 70 bytes. In short, both the computation and storage overhead are applicable for the ordinary sensor nodes.

(4) Outdoor Evaluation

Now I move the experiments to the outdoor environments: racket court and parking lots. The moving velocity of the mobile beacon is set at 0.1m/s and the broadcast frequency is set at 1 time per second. The localization errors of 4 typical sensor nodes in the two environments are shown in Figures 4-7 (a) and (b), respectively.

We can see that the three approaches achieve the smallest estimation errors in the hall experiments, similar estimation errors in two outdoor experiments, and the largest estimation errors in the laboratory experiments. This is consistent with the fact that all RSSI-based localization approaches are more or less affected by the interference in wireless signal propagation, and the dynamic of the environments.

We can also see even the worst result of PI (in the laboratory) is better than the best results of BI and TRL. This result again confirms the advantage of PI, which compares the measured RSSI values of beacon packets to calculate the coordinates of the nodes, effectively eliminating the side-effect of the irregularity of measured RSSI values.
4.2 GreenOrbs-based Observations on RSSI and Range Measurements

4.2.1 Anisotropic problem

Driven by the forestry applications, GreenOrbs deploys more sensor nodes in the regions with diverse or uneven vegetations, so as to provide comprehensive and fine-grained information of the monitored area. Such a rule leads to non-uniform deployment of sensor nodes. As we observe in a 330-node GreenOrbs deployment, some nodes have more than 20 neighbors (a neighbor is defined as a node that is within the transmission range of the local node) while some nodes have less than 5 neighbors.

Figure 4-8: Cumulative distribution of node distances

The distances between pairs of neighboring nodes differ a lot as well, as shown in Figure 4-8. The shortest distance is 5m and the longest one is around 108m.
The above facts imply the anisotropic problem with localization. When using a rang-free approach (e.g. DV-Hop) without considering this problem, nodes are located at the same position if their hop-counts to the landmarks are equal. Nevertheless, they might be at a distance from each other in reality. The anisotropic problem also brings severe challenges to range-based localization. I present the details in the next group of observations.

4.2.2 Irregularity of RSSI

Besides the anisotropic problem, the complex terrain and obstacles (e.g. shrubs and trunks) also affect signal propagation in the forest. The resulting RSSI among the nodes is very irregular. Figure 4-9 plots the RSSI between node pairs in GreenOrbs at a certain time. It further includes a curve, which shows the mapping between RSSI and distance based on the log normal shadowing model.

\[
PL(d) = PL(d_0) + 10 \times \eta \times \log \left( \frac{d}{d_0} \right) + X_\sigma
\]

(4-11)

where \(PL(d)\) denotes the reduction in received signal strength after propagating through a distance \(d\), \(PL(d_0)\) stands for the path loss at a short reference distance \(d_0\), \(\eta\) is the path loss factor (also named signal propagation constant), and \(X_\sigma\) is a random environment noise following \(X \sim N(0, \sigma^2)\). For all the experiments in this chapter, I set the parameter values as \(\eta = 3.3, \sigma = 6\), according to the empirical knowledge reported in [71].

We can see that the real distances between node pairs differ greatly from the model-based estimations. The mapping from RSSI to distance is actually very uncertain. For example, \(RSSI = -85\text{dBm}\) in the figure may correspond to a distance from 31m to about 51m. Yet the contrast of RSSI still offers useful information. In most cases, a stronger RSSI corresponds to a shorter distance, as is also observed by some related work [46, 86]
4.2.3 Asymmetry and Dynamics of RSSI

Figure 4-10 shows the pairwise RSSI between two nodes in GreenOrbs, along with the temperature and humidity over time. The distance between A and B is 41.27 meters. We can see that RSSI between two nodes is asymmetric. Two pairwise links probably have unequal RSSI. Moreover, RSSI is very susceptible to the environmental factors, such as humidity and temperature. At different times of a day, the RSSI over a link is highly dynamic. Figure 11 plots the mean RSSI versus varying humidity. Basically, the RSSI appears higher when the humidity goes up.

![Figure 4-10: Asymmetry and dynamics of RSSI](image)

(a) RSSI between nodes A and B  
(b) Corresponding environmental dynamics

![Figure 4-11: Mean RSSI vs. varying humidity](image)
4.2.4 Errors of RSSI-based range measurements

I randomly select a time to collect the RSSI readings from all the 100 nodes. The RSSI are then used for range measurements based on Equation (4-11). By comparing the converted distances with the ground-truth ones, Figure 4-12 plots the mean error of range measurements between every node and its neighbors. Generally, if a node has line-of-sight connections with all its neighbors, the mean error of its range measurements is small. If a node lies in a pit, has a faulty antenna, or is relatively far from all its neighbors, the mean error is likely to be large.

4.3 CDL: A Combined and Differentiated Localization Approach

I consider locating a network of wireless nodes on the two dimensional plane by using the connectivity information and RSSI readings among them. A few nodes are used as landmarks, which know their own coordinates once they are deployed.

The design of CDL mainly consists of virtual-hop localization, local filtration, and ranging-quality aware calibration. Figure 4-13 illustrates the work flow of CDL.
Virtual-hop localization initially estimates node locations using a range-free method. In order to approximate the distances from the nodes to the landmarks, it counts virtual-hops instead of DV-hops, particularly compensating the errors caused by the anisotropic problem.

After that, CDL executes an iterative process of filtration and calibration. In each filtration, CDL use two filtering methods to identify good nodes whose location accuracy is already satisfactory. Neighborhood hop-count matching filters the bad nodes by verifying a node’s hop-counts to its neighbors. Further, neighborhood sequence matching distinguishes good nodes from bad ones by contrasting two sequences on each node. Each sequence sorts a node’s neighbors using a particular metric, such as RSSI and estimated distance.

Those identified good nodes are then regarded as references and used to calibrate the locations of the other nodes. Links with different ranging errors are given different weights in the calibration process. Outliers in range measurements are tolerated using robust estimation.

![Figure 4-14: Nodes with same hop counts have different distances to the landmark](image)

4.3.1 Virtual-Hop Localization

As the first phase of CDL, Virtual-hop localization initially computes the node locations. It is an enhanced scheme of hop-count based localization. Compared to the traditional hop-count schemes, virtual-hop particularly mitigates the anisotropic problem in per-hop distance.

This section starts with the analysis of the weakness of DV-hop, a representative of hop-count based localization scheme. Then I elaborate how I upgrade DV-hop to virtual-hop, followed by a quick comparison of their localization accuracy.

1. Weakness of DV-hop
DV-hop utilizes the connectivity information to estimate node locations. Every node counts its hop counts (i.e. the number of hops on the shortest path between two nodes) to at least three landmarks. The distance between a node and a landmark is calculated as the product of the hop count between them and the per-hop distance, which is a pre-determined constant for all the nodes. The location of a node is then calculated by using Least Square Estimation [79].

Due to the usual non-uniform deployment of sensor nodes, the shortest path from a node to a landmark might be straight or tortuous. Figure 4-14 shows an example to illustrate the weakness of DV-hop. A landmark floods its location information in the direction of the arrows. A circle denotes a node. We can see many nodes have equal hop counts but clearly different distances to the landmark. The distance between the two ends of a straight path is clearly longer than that between the two ends of a tortuous path. A constant value of per-hop distance for every node often causes errors of distance calculation from the node to the landmarks, as is called the anisotropic problem of per-hop distance. As a result, the localization accuracy of DV-hop is far from being satisfactory.

(2) Virtual-hop

Figure 4-14 implies another interesting fact as follows. For two nodes that have equal hop counts to the landmark, the node nearer to the landmark is likely to have more previous-hop neighbors and less next-hop neighbors. For ease of understanding, let’s compare the red node P and the blue node Q in the figure. They are both two hops from the landmark. P has two previous-hop neighbors and zero next-hop neighbor, while Q has one previous-hop neighbor and three next-hop neighbors. We can observe that P is apparently nearer to the landmark than Q is.

Based on the above observation, I propose the design of virtual-hop localization. Each node maintains two types of hop counts. One of them is the real hop counts, based on which a node determines its previous-hop and next-hop neighbors. The other is virtual hop count, which is calculated in hop-by-hop recurrence as follows. I use the following notations.

- \( n_j \) is the number of previous-hop neighbors of node \( P_j \).
- \( m_j \) is the number of next-hop neighbors of node \( P_j \).
- \( m_j' = \min \{ n_i \mid \text{node } P_i \text{ is a next-hop neighbor of node } P_j \} \).
- \( n_j' = \min \{ m_i \mid \text{node } P_i \text{ is a previous-hop neighbor of node } P_j \} \).

\( VH_{jk} \) denotes the virtual hop count from landmark \( R_k \) to node \( P_j \), which is defined as follows.
\begin{equation}
VH_{jk} = \begin{cases} \\
\frac{1}{n_j} \sum_{i \in n_j} VH_{ik} + \frac{m_j}{m_j + m_j'}, & \text{if } m_j > 0 \\
\frac{1}{n_j} \sum_{i \in n_j} VH_{ik} + \frac{n_j'}{n_j + n_j'}, & \text{if } m_j = 0
\end{cases}
\end{equation}

The per-virtual-hop distance regarding landmark \( j \) is calculated by
\[ \tilde{d}_k = \frac{\sum_{R_t \neq k} \rho_{tk}}{\sum_{i=1} VH_{ik}}, \]
where \( R_t \) is a landmark, \( t \neq k \).

\[ \rho_{tk} = \sqrt{(X_t - X_k)^2 + (Y_t - Y_k)^2} \]
is the Euclidean distance between landmarks \( R_t \) and \( R_k \). Each node computes its distance to the landmarks by
\[ \rho_{jk} = \tilde{d}_k \cdot VH_{jk}, \]
where \( \rho_{jk} \) is the estimated distance from node \( P_j \) to landmark \( R_k \). After calculating the distances to the landmarks, each node computes its coordinates using Least Square Estimation, which is similar with DV-hop.

(3) Localization Accuracy of Virtual-hop

Now I conduct a simulation to compare the localization accuracy of virtual-hop localization with DV-hop. The trace I use in the simulation includes 100 nodes and 4 landmarks. The simulation results are shown in Figure 4-15. We can see virtual-hop remarkably outperforms DV-hop. The performance gain of using virtual-hop varies a lot among different nodes. Compared with DV-hop, the localization errors are reduced by 10%~99%.

By fully exploiting the connectivity information of local neighborhood, virtual hop counts finely characterize the anisotropic property and address it by more precise hop counting. Nevertheless, it is worth noticing that there are still sizable errors (>5m) with many nodes. Those nodes with sizable location errors can be indentified and accurately calibrated. I leave the solutions in the next two sections.

Without causing confusions, hereafter I use “initial coordinates” to denote the node coordinates output by virtual-hop localization.

Given the initial coordinates, the iterative process of filtration and calibration further enhances the localization accuracy. It involves the following two design criteria: First, filtration must identify as many as possible good nodes with high localization accuracy to facilitate calibration. Second, a good node is likely to have both good and bad links. Only the
good links (with small ranging errors) should dominate calibration, while the impact of the bad links must be restrained. Filtration resolves the first criterion, while calibration resolves the second.

![Comparison of DV-hop and Virtual-hop](image)

**Figure 4-15: Comparison of DV-hop and Virtual-hop**

### 4.3.2 Local Filtration

Filtration consists of two steps: neighborhood hop-count matching and neighborhood sequence matching.

1. **Neighborhood Hop-Count Matching**

   Every node takes neighborhood hop-count matching as the first step to identify whether it is a bad node. It mainly utilizes the local connectivity information. I use node $P_i$ as an example to illustrate the matching procedure.

   First, every node exchanges its initial coordinates with other nodes. To restrict the communication overhead, I set the scope of exchanges as within 2-hop neighborhood.

   Second, when node $P_i$ receives the initial coordinates of another node $P_k$, $P_i$ estimates the distance between them, denoted by $d_{ik}$.

   Third, for each node $P_k$ within its 2-hop neighborhood, $P_i$ estimates the hop-count to $P_k$ as $h_{ik} = d_{ik} / D$, where $D$ is the per-hop distance obtained during virtual-hop localization.

   Fourth, $P_i$ compares every pair of $(h_{ik}, h_{ik}')$, where $h_{ik}$ is the hop-count based on the real network topology. The ratio of unmatched hop-counts (i.e. $h_{ik} \neq h_{ik}'$) on $P_i$ (denoted by $r_i$) is then exchanged within its 2-hop neighborhood. Let $\beta_i$ denote the mean value of unmatched ratios. If $r_i > \beta_i$, $P_i$ regards itself as a bad node, which has an apparent error in its initial coordinates. Otherwise, the role of $P_i$ is left undetermined for further filtration.

   Hop-counts actually offer relatively limited information to filtration. As a result, neighborhood hop-count matching only identifies a small portion of bad nodes with apparently wrong coordinates. In order to ensure that all the sifted good nodes do have
satisfactory location accuracy, I need to further filter the bad nodes. In the next two subsections, I will first discuss the infeasibility of model-based filtration, and then illustrate my scheme of neighborhood sequence matching.

(2) Infeasibility of Model-based Filtration

There are two ways to estimate the distances between two nodes, e.g. node $P_i$ and its neighbor $P_j$. One way is to calculate with their initial coordinates, denoted by $l_{ij}$. The other converts the RSSI from $P_j$ to $P_i$ into a distance based on the log-normal shadowing model, denoted by $d_{ij}$. Ideally, $d_{ij} = l_{ij}$. Due to the errors of initial coordinates, however, there is often some difference between them. By summing up the difference $|d_{ij} - l_{ij}|$ corresponding to every neighbor $P_j$, we can measure the Aggregated Degree of Mismatches (ADM) of $P_j$.

ADM actually reflects the error of a node’s estimated location. For example in Figure 4-16 (a), node $A$ is a good node with six neighbors. Among them only node $F$ is a bad node while $F'$ denotes its estimated location. Clearly the ADM of node $A$ is mainly caused by $F'$. In Figure 4-16 (b), node $A$ is a bad node with six good neighbors. The link to every neighbor contributes to the ADM of node $A$. By comparing these two figures, we can see the ADM of a bad node is higher than that of a good one. Thus we may distinguish good nodes from bad ones by contrasting their ADM.

(a) A good node with one bad neighbor

(b) A bad node with six good neighbors

Figure 4-16: ADM reflects the localization error of a node

It is worth noticing that the above observation depends on the following assumption: in the neighbor set of a node, good nodes are more than bad ones. This is almost always true, given the fairish accuracy of virtual-hop localization. Even if bad nodes are more than good ones in some rare cases, the consequence is merely a false negative judgment, i.e. a good node
is identified as a bad one. Such a judgment is conservative but does not introduce any fault into the subsequent calibration process.

The above method to quantify ADM, however, relies on the log-normal shadowing model to convert RSSI into distance. As I have observed, such conversion is error-prone, which leads to totally incorrect filtration.

Here I conduct a group of experiments to validate the above opinion, using the same trace as that in Subsection 4.3.1. I call the straightforward model-based calibration as rude calibration. Using such calibration, every node’s location is adjusted based on the distances to its neighbors, while the distances are calculated based on the log-normal shadowing model.

![Figure 4-17: The incorrectness of rude calibration](image)

Figure 4-17 compares the localization errors of nodes before and after rude calibration. Surprisingly, I find the output of rude calibration is even worse than that before rude calibration. To rely on the RSSI-to-distance conversion clearly hurts instead of benefits the localization accuracy. In short, model-based filtration is infeasible, considering the irregularity of RSSI.

(3) Neighborhood Sequence Matching

Though model-based filtration is infeasible, the contrast of RSSI still offers useful information. Generally the RSSI between two nodes decreases monotonically as the node distance increases, as can be observed from the RSSI readings in Figure 3. Based on such an observation I propose the filtration scheme called neighborhood sequence matching.

The filtration on a node $A$ runs as follows.

First, according to the node IDs, $A$ sorts its neighbors in the ascending order, generating an ID sequence denoted by $N_A$. 
Second, node $A$ sorts its neighbors in the descending order with regard to the RSSI from the neighbors, generating a sequence number for each neighbor. By mapping the sequence numbers into $N_A$, I get the second sequence called RSSI sequence. Let $S_A$ denote it.

Third, according to the initial coordinates, node $A$ sorts its neighbors in the ascending order with regard to the estimated distance to the neighbors, generating the third sequence called distance sequence. Let $S'_A$ denote it.

In theory, $S_A$ and $S'_A$ should be identical. If there is significant mismatch between them, it indicates a large error in the node’s initial coordinates. I use the same examples in Figure 4-16 to illustrate the above idea. As shown Figure 4-18 (a), node $A$ is a good node while node $G$ is the only bad neighbor whose location is mistaken as $G'$. There is not much mismatch between $S_A$ and $S'_A$ in this case.

Comparatively in Figure 4-18 (b), when node $A$ is a bad node with six good neighbors, there appears to be significant mismatch between $S_A$ and $S'_A$.

Now that the difference between $S_A$ and $S'_A$ is a measure of the location error of node $A$. The next step of filtration is to quantify their distance. Thus I define the matching degree between the RSSI sequence and distance sequence as follows.

$$
\tau = \delta \cdot \frac{a_1a'_1 + a_2a'_2 + \cdots + a_na'_n}{\sqrt{a_1^2 + a_2^2 + \cdots + a_n^2} \sqrt{a_1'^2 + a_2'^2 + \cdots + a_n'^2}}
$$

(4-15)

where $n$ is the number of $A$’s neighbors, $\delta$ is the proportion of the length of the longest matched neighborhood sequence and the length of neighborhood sequence of node $A$, $a_1, a_2, ..., a_n$ are the sequence numbers in $S_A$ while $a'_1, a'_2, ..., a'_n$ are the sequence numbers in $S'_A$.

Since $\{a_1, a_2, ..., a_n\} = \{a'_1, a'_2, ..., a'_n\} = \{1, 2, ..., n\}$, we have:

$$
\tau = \frac{\delta (a_1a'_1 + a_2a'_2 + \cdots + a_na'_n)}{1^2 + 2^2 + \cdots + n^2}
$$

$$
= \frac{6 \times \delta (a_1a'_1 + a_2a'_2 + \cdots + a_na'_n)}{n(n+1)(2n+1)}
$$

(4-16)
I use the same trace as that in Figure 4-15 to calculate the matching degree of all the nodes after initial localization. The results are plotted in Figure 4-19. Nodes with matching degree over 0.6 have location errors of less than 4 meters. I regard them as good nodes. Nodes with matching degree less than 0.4 have location errors over 5 meters. I regard them as bad nodes. Besides, it is hard to determine the type of the rest nodes right now. Their matching degree is between 0.4 and 0.6, but their location errors vary from 0.1 to 12 meters. Thus I tentatively set them as undetermined nodes.
\( \tau = 0.4 \) and \( \tau = 0.6 \) are two empirical parameters, called lower matching threshold and upper matching threshold. I use them for filtration. One can increase both thresholds to execute stricter filtration. One can also decrease both thresholds to allow more nodes to contribute as good nodes in the calibration process. The tradeoff in the threshold settings could be an interesting issue to be studied. I leave it in my future work.

4.3.3 Ranging-Quality Aware Calibration

After the phase of filtration, I use the good nodes as references to calibrate the coordinates of bad nodes. As the set of undetermined nodes include both good and bad ones, I do not include any undetermined node in the calibration process. My scheme is range-quality aware calibration (RQAC). From the viewpoint of node \( P_i \), the ranging quality of its neighbor \( P_j \) is simultaneously determined by two factors: the location accuracy of \( P_j \) and the ranging error over the link from \( P_j \) to \( P_i \).

Given the range measurements between a node \( P_i \) and its neighbors, estimation of \( P_i \)’s location, denoted by \( f^* \), usually works by minimizing an objective function over node pairs \((i, j)\), which is denoted by

\[
 f^* = \sum_j g(i, j), \quad (4-17)
\]

where \( g(i, j) \) takes different forms with different approaches.

When least square estimator (LSE) is used,

\[
g(i, j) = (l'_i - d_{ij})^2, \quad (4-18)
\]

where \( l'_i \) denotes the distance estimated by LSE and \( d_{ij} \) denotes the RSSI-based range measurement between \( P_i \) and its \( j \)-th good neighbor. The problem with LSE is that it equally treats the nodes and links without any differentiation. It suffers great errors when outliers are present in the node locations or range measurements.

SISR [85] outperforms LSE by assigning different weights to the range measurements with different neighbors.

\[
g(i, j) = \begin{cases} 
  \alpha(l'_i - d_{ij})^2 & \text{if } |l'_i - d_{ij}| < \lambda \\
  \ln(|l'_i - d_{ij}| - u) - v & \text{otherwise}
\end{cases}, \quad (4-19)
\]

where \( \alpha, \lambda, u, \) and \( v \) are constant parameters. SISR is effective in restraining the errors introduced by noisy range measurements. Nevertheless, being unaware of the location accuracy of the reference nodes, SISR actually prefers the situations where the majority of
range measurements are accurate. It becomes inefficient in the rest cases, as will also be proved by the experiments in Section 4.4.

To address the limitations of LSE and SISR, RQAC adopts the technique of weighted robust estimation. With RQAC, I first estimate the location accuracy of a good node as follows.

$$\omega_j = \sum_{k=1}^{m} \omega'_{jk}$$

(4-20)

for each good neighbor Pk of node Pj.

$$\omega'_{jk} = \begin{cases} 1 & \text{if } |l'_j - d_{jk}| < \theta \\ 0 & \text{otherwise} \end{cases}$$

(4-21)

where \( \theta \) is a parameter to be configured. It is easy to see that a larger value of \( \omega_j \) implies a more accurate estimate of node \( P_j \)'s location as well as a higher overall ranging quality of \( P_i \).

Suppose \( P_j \) is a good neighbor of a bad node \( X \), the weight of \( P_j \) in calibrating \( X \) is defined as a normalized value of \( \omega_j \).

$$\omega_j = \frac{\omega_j}{\sum_{k=1}^{n} \omega_k}$$

for all good neighbors of \( X \). (4-22)

The objection function of RQAC is defined as follows.

$$g(i,j) = \begin{cases} \omega_j (l'_i - d_i)^2 & \text{if } |l'_i - d_i| < \varepsilon \\ \ln(|l'_i - d_i| - \varepsilon + 1) + \omega_i \varepsilon^2 & \text{if } |l'_i - d_i| \geq \varepsilon \text{ and } \omega_i > \frac{1}{\varepsilon} \\ 0 & \text{otherwise} \end{cases}$$

(4-23)

where \( \varepsilon \) is a parameter to be configured. Note that \( |l'_i - d_i| \) is a measure of the ranging error. \( \omega_j \) and \( |l'_i - d_i| \) thus jointly denote the ranging quality from \( P_j \) to \( P_i \).

As we can see from Equation (4-23), range measurements to \( P_i \) are divided into three classes, according to their ranging quality. The range measurements with errors less than \( \varepsilon \) contribute more to the calibration process by taking the quadratic form of \( |l'_i - d_i| \).

For a range measurement with an error not less than \( \varepsilon \), if the corresponding reference node has relatively good location accuracy (i.e. \( \omega_j > 1/\varepsilon \)), its contribution is suppressed by taking the logarithmic form of \( |l'_i - d_i| \).

The rest range measurements are ignored in calibration. Moreover, range measurements in the same class are also differentiated from each other, by taking the weights of reference nodes (\( \omega_j \)) into account. In this way, RQAC respects the contributions of the best range
measurements, eliminates the interference of outliers, and suppress the contributions from the middle class.

As for the parameter setting in RQAC, small values of $\theta$ and $\epsilon$ express a conservative calibration strategy. Only a small fraction of the best range measurements receives enough respect, which leads to highly accurate calibration but likely more rounds of iterations. Large values of $\theta$ and $\epsilon$ express an optimistic calibration strategy. Many good range measurements can make considerable contributions, as increases the efficiency of iterations but likely introduces new errors.

4.4 Evaluation

I have implemented CDL with GreenOrbs. The performance of CDL is evaluated through large-scale real experiments as well as simulations. For comparison, I have also implemented three existing localization approaches, namely DV-hop [79], MDS-MAP(C,R) [114], and SISR [85].

Specifically in MDS-MAP(C,R), the 1-dimensional MDS is applied first. Then four corner nodes are used as landmarks for scaling and rotation of the map in 2-dimensional MDS. MDS-MAP(C,R) avoids the problem that some different nodes are estimated to the same location. Its localization performance is usually better than DV-hop. However, the bad links cause big errors in the range measurements. This problem remains unresolved in MDS-MAP(C,R). The estimated coordinates of some nodes still often deviate a lot from the real ones.

Figure 20 plots the 100 GreenOrbs in a rectangular region. Four nodes positioned near the border of the deployment area are selected as landmarks, because they are not covered by the tree canopy and can be directly located using a precise GPS device. I have collected the localization results of all the four approaches.
4.4.1 Experiments

(1) Comparison among Approaches

Figure 21 plots the cumulative distribution of the localization errors using the four approaches. It is easy to see that SISR performs better than DV-hop and MDS-MAP(C,R). Thus I only compare the results of SISR and CDL in Figure 20.
Figure 20 shows that for almost all the nodes, CDL achieves higher localization accuracy than SISR. Here are the detailed explanation of the results in Figures 20 and 21.

Using CDL, 100% of the nodes have errors of less than 7 meters, while 65% of them have errors of less than 3 meters. Using SISR, at most 70% of nodes have errors of less than 7 meters and at most 35% of nodes have errors of less than 3 meters. It is also interesting to see that CDL achieves the most consistent performance among the four approaches.

From Figure 21, we can see the performance of DV-hop is the worst one. Actually, I observe in the experimental results that many different nodes are estimated to the same locations by DV-hop, because they have same hop-counts to the landmarks, but their real locations are far from each other.

Another interesting finding is SISR and MDS-MAP have similar performance. In other words, a node with a large error in MDS-MAP usually has a large error in SISR, too. Moreover, due to the “snap-in” behavior of SISR, it is able to suppress the negative impact of noisy range measurements. SISR therefore achieves slightly better accuracy than MDS-MAP.

(2) Efficiency of Iteration

Note that CDL and SISR both propose iterative localization processes. Other than comparing the overall performance, I conduct experiments to evaluate the efficiency of iteration.

As shown in Figure 22, the mean localization errors of CDL and SISR both decrease as the iterations go on. Their performance both converges after 6~8 rounds of iterations. The localization accuracy of CDL is always better than that of SISR.

I want to emphasize that, setting an objective of localization accuracy, the required number of iterations actually determines the communication and computational cost of a localization approach. By examining the results in Figure 22, we are pleased to see CDL achieves very satisfactory localization results even with only two rounds of iterations. The performance gain of a few more rounds of iterations is also more apparent with CDL than with SISR. The accuracy of SISR starts to go up apparently until the 4th round of iteration.

CDL outperforms SISR mainly because of the following reasons.

First, the initial round of iteration of CDL starts with the output of virtual-hop localization, while SISR starts totally undetermined locations of the nodes. With only straightforward location adjustment, SISR cannot converge sufficiently fast. As a result, the localization error of SISR in the first few rounds of iterations is especially large.

Second, filtration and calibration in CDL explicitly filter the bad nodes and identify the good nodes and good ranging quality. Hence CDL may maximize the contribution of those
good nodes in calibration and clearly eliminates the negative impact of bad nodes. Such a capacity is especially preferred at the first few rounds of iterations, when the number of good nodes is relatively limited.

On the other hand, SISR conducts calibration without explicit differentiation on the quality of ranging. As I observe with GreenOrbs, there are many mild ranging errors as shown in Figure 4-12. SISR is effective only when there is a relatively small portion of sizable ranging errors. When sizable ranging errors widely exist with most of the nodes, SISR cannot accurately distinguish between bad nodes and good ones. The consequence is inefficiency in the starting stage of iterations.

We may notice that after some rounds of iterations, the localization errors of CDL reach relatively a stable value and does not goes down any more. The errors remaining in the localization results are mainly incurred by a small portion of bad nodes which cannot be better located. Those bad nodes do not have sufficient number of good neighbors or they are likely to have potential ranging errors to almost all their good neighbors. When usable information from the good neighbors is used up, the localization errors converge. This is a common fact with CDL and SISR. Increasing the transmission power might be a simple but effective method to improve such a situation. Since this is not the key issue to be studied in this chapter, I leave it in my future work.

![Comparison of SISR and CDL with iterations](image.png)

**Figure 4-22: Comparison of SISR and CDL with iterations**
(3) Evolution of Node Types

In this subsection, I particularly observe the evolution of node types, namely good, bad, or undetermined, along with the iterative filtration and calibration of CDL.

Figure 4-23 (a) shows the trends of the changing numbers of good, bad, and undetermined nodes along with 10 iterations. First of all, CDL makes deterministic judgments on node types (good or bad) for almost all the nodes after 10 rounds of iterations. The effect of calibration is also satisfactory. We can see the number of good nodes quickly increases as iterations go on.

In order to avoid false positive and false negative filtration, I conservatively leave some nodes with medium values of matching degree. Hence there remains a considerable amount of undetermined nodes in each of the first iterations. As the filtration and calibration continue,
more and more bad nodes are successfully calibrated by the good ones. The uncertainties of node location are weakened, so the undetermined nodes gradually obtain the opportunity to be identified and calibrated.

Figure 4-23 (b) shows the detailed evolitional process of all the 100 nodes in 10 iterations. Specifically, 42% nodes are correctly identified to be good from the very beginning. 28% nodes are initially identified as bad, and then calibrated to be good. 26% of them are initially undetermined, then identified as bad nodes, and finally calibrated to be good. The rest nodes, which count only 4% of all, are left undetermined at the end of 10 rounds of iterations.

![Bar chart showing the impact of humidity on localization error](image)

**Figure 4-24: Impact of humidity**

(4) The Impact of Environmental Factors

Our observation in Section 4.2 lends us an impression that the dynamics of environmental factors have significant impact on RSSI among the nodes. RSSI concerns the range-based calibration. It also affects the transmission range of nodes. Therefore dynamics of RSSI also concerns the accuracy of virtual-hop localization. Thus it would be interesting to investigate how the localization approaches perform under different environmental conditions.

Figure 4-24 plots the localization results of the four approaches under two representative conditions in GreenOrbs. Since GreenOrbs collects the humidity data, I’m able to monitor the humidity along time while carrying out the localization experiments.

Actually RSSI is affected by both temperature and humidity. Note that in Figure 11(b), temperature and humidity presents completely inverse trends of variation. I believe the humidity data alone is sufficient for us to evaluate the impact of both humidity and temperature.

As we can see from Figure 4-24, all the four approaches achieve better performance with the higher humidity. Recall the experience reported in [49], RSSI increases with humidity increases. I conjecture that the performance improvement with higher humidity is due to the
following reasons. Larger RSSI results in longer transmission range and in turn less anisotropic problem with range-free location. Large RSSI also enables the nodes to have more neighbors, increasing the chance of calibration.

This group of experiments implies that environmental factors do matter in localization. One may expect to achieve higher localization accuracy in the weather when the wireless signals are relatively stable and strong. When we make judgments on the quality of a localization approach, the environmental conditions of the localization being executed should definitely be considered.

4.4.2 Simulation

Besides the above experiments based on the implementation of CDL in GreenOrbs, I have carried out extensive simulations to evaluate the performance of CDL. I examine the location accuracy of CDL by tuning a series of parameters such as node density, the number of landmarks, and the relative ranging errors. The simulation results of DV-hop, MDS-MAP(C,R), and SISR are also presented for more comprehensive comparison.

In the simulations, nodes are randomly deployed in a 500m*500m square region. I set the transmission range of nodes to be 30m.

(1) Node Density

The node density ($N_d$) is the average number of one hop neighbors of a node. It is often an affecting factor to localization. In this group of simulations, I keep the number of landmarks constant in the network and tune the node density by change the number of nodes. I set the number of landmark equal to 6 and the percentage of bad links is 30%. To simulate the range

![Figure 4-25: Impact of node density](image-url)
measurements on the bad links, I let the relative ranging errors on the links conform to a Gaussian distribution, denoted by $\mathcal{N}(0.5, 0.1)$.

Figure 25 compares the converged value of mean localization errors using the four approaches. For DV-hop and MDS-MAP(C,R), since they are not iterative methods, I just present their localization results from one time of execution. The curves in Figure 25 show that a higher node density results in higher location accuracy for all the four approaches.

Specifically, DV-hop performs very poor when the node density is low. This is because the anisotropic problem appears to be more serious with sparse deployments. When the node density increases, the anisotropic problem is mitigated and the localization accuracy is thus improved.

For MDS-MAP(C,R), a high node density makes the total length of a shortest path between two nodes correspond well to their Euclidean distance.

With a higher node density, it is also easier for SISR to obtain enough good range measurements for localization.

As for CDL, it is least susceptible to low node density among the four approaches, because virtual-hop localization already addresses the anisotropic problem to a certain extent. Another advantage of CDL is that even with limited good nodes, it still can make efficient filtration and calibration.

Nevertheless, node density does matters for CDL as well. As the node density goes up, the mean localization error quickly drops and then gradually converges. A higher node density usually results in better accuracy of virtual-hop localization. Moreover, as the node density increases, the chance also gets better for CDL to make more efficient filtration and calibration.

(2) Impact of the Number of Landmarks

Due to the complexity of deployments and location measurements of WSNs in the wild, there are usually a limited number of landmarks. Thus it is worth examining how CDL works with different amount of landmarks. In my simulation, I tune the number of landmarks from the minimum required number (i.e. 3) to a relatively large number, namely 16. The total number of nodes is 1000. The node density $N_{d}=12$ and the percentage of bad links is 30%. All the four approaches perform well with such settings, according the previous experimental results.

Figure 26 plots the mean localization errors of the four approaches as the number of landmarks is increased. We can see that MDS-MAP(C,R) and SISR are less sensitive to the number of landmarks that DV-hop and CDL. The main reason is that landmarks are not
involved throughout the localization process with MDS-MAP(C,R) and SISR. For example, SISR first generates a relative location map of the network and then transforms it to absolute positions when sufficient landmarks are available.

**Figure 4-26: Impact of the number of landmarks**

On the other hand, DV-hop simply relies on counting hops to the landmarks to calculate the node locations. CDL is also a bit affected by the number of landmarks, because it includes virtual-hop localization as a component. Yet I notice that CDL performs slightly worse than SISR only when there are 3 landmarks. The localization accuracy of CDL is improved remarkably by adding one or two landmarks to the network. Overall, 6 landmarks are more than enough for CDL to achieve satisfactory performance.

(3) Ranging Error

Considering the ubiquitous ranging errors and poor overall ranging quality in the wild, the robustness of a localization approach against such interfering factors is the last but not least metric I want to evaluate. For this purpose, I conduct a group of simulations with 1000 nodes. Node density $N_d=12$ and the number of landmark is set at 8.

I use another two parameters to control the degree of ranging errors. The first one is the percentage of bad links which is respectively set at 0%, 10%, 20%, 30%, 40%, and 50%.

Another parameter is the relative ranging error. We assume in the simulations that the links on a node are either all good or all bad. The relative ranging error of a link conforms to a Gaussian distribution $N(\mu_{bad}, 0.2\mu_{bad})$, where $\mu_{bad}$ denotes the average of relative ranging error and set at 0%, 10%, 20%, 30%, 40%, and 50%, respectively. Meanwhile, we assume the links are asymmetric.

Figure 27 plots the mean localization errors of MDS-MAP (C,R), SISR, and CDL under different settings. I find MDS-MAP (C,R) is relatively insensitive to the changes of ranging
errors, because it mainly relies on the connectivity information instead of inter-node ranging. All MDS-MAP(C,R) results have localization error of more than 2m, even when all links are good.

Figure 4-27: Comparison of localization errors:
(a) MDS-MAP(C,R); (b) SISR; (c) CDL;
SISR generally performs better than MDS-MAP(C,R). Specifically, it performs well when the percentage of bad links is less than 30%. The mean localization errors are less than 2m due to the “snap-in” behavior of SISR. Its performance seriously degrades when the percentage of bad links gets above 30%, as is in accordance with my analysis in Subsection 4.3.3.

Compared to SISR, CDL has even better performance. When all the links are good, its localization errors are almost near zero. Even when there are 50% bad links, CDL still perform robustly enough. The mean localization error is around 5m. This group of simulation shows the remarkable advantages of CDL in extremely complex environments.
5 Forwarding-Quality-Aware Data Collection

There have been many estimation metrics proposed to measure the forwarding quality of a multi-hop path, such as ETX, ETF, PRR, and etc. Existing metrics mainly focus on estimating the packet delivery quality on links in between the nodes. The quality of forwarding capacity along a path is estimated by the aggregate of the forwarding qualities of all the links on the path. Those link-based metrics while reflect the link performance of the path, however, overlook the forwarding capabilities inside the sensor nodes, thus resulting in an incomplete measurement of the path quality. Using the incomplete path indicators will lead to suboptimal routing decisions and degraded routing performance.

Such an effect has been revealed in our experience in manipulating GreenOrbs, a large-scale sensor network with 330 nodes. In current routing implementation in GreenOrbs, we use a modified CTP that relies on path ETX estimation for routing selection. During the field test of the system, we observe a portion of packets drops on some nodes. They are due to a variety of causes, such as forwarding queue overflow under high traffic pressure, software bugs in the CTP implementation, and etc. Those nodes, however, still respond with ACKs at the radio hardware. The bad fact is that with current path indicators the inability of packet forwarding within the individual nodes cannot be shared among the network, yet there is not a metric to quantify the packet forwarding quality at each node. Therefore, the CTP routing chooses data delivery paths blind of those problematic nodes. As a result, the path estimation is not intact with the link-based indicator and the data delivery performance is severely degraded.

The packet drops on the problematic nodes introduce intrinsic unreliability in data delivery. As a matter of fact, even a single link itself can hardly achieve full reliability in practical systems. ETX over a link measures the expected number of transmissions for successfully delivering a packet, but transmitting the packet at the expected number of times does not guarantee it will be successfully received at the receiver end. In practical systems, we usually set a maximum number of retransmissions on a link to prevent keeping sending a packet on a “bad” link infinitely. That exhausts the finite communication resources. The packet will be eventually dropped by the sender after a maximum number of retransmission tries. The network is thus rendered unreliable due to both node unreliability and link unreliability. ETX of a path is estimated as the summation of ETX values over all links constituting the path. Using path-ETX for path selection minimizes the transmission cost and
achieves a high throughput. However, ETX only works well when end-to-end delivery is presumed reliable which, however, is not always the truth as we see from the above.

\[
P_1: \begin{array}{c}
3 & q_1 = \frac{1}{10} \\
\text{etx} = 10
\end{array} \quad \begin{array}{c}
2 & q_2 = \frac{1}{10} \\
\text{etx} = 10
\end{array} \quad \begin{array}{c}
1 & \text{etx} = 20
\end{array}
\]

\[
P_2: \begin{array}{c}
3 & q_1 = 1 \\
\text{etx} = 1
\end{array} \quad \begin{array}{c}
2 & q_1 = \frac{1}{19} \\
\text{etx} = 19
\end{array} \quad \begin{array}{c}
1 & \text{etx} = 20
\end{array}
\]

**Figure 5-1: Two paths with same ETX but different PRR (packet reception ratio)**

For data delivery within the network of inherent unreliability, a metric that better measures the data productivity is the amount of successful data delivery to the destination, i.e., data yield [115]. Data yield over the actual number of data transmissions, measures both transmission cost as well as achieved throughput. Existing path-ETX cannot capture such a parameter. Consider a simplified example depicted in Figure 5-1. There are two paths, both of which have path-ETX of 20. Suppose there is no link layer retransmission. In path 1, the destination receives 19 packets in expectation provided the source sends 1900 packets while in path 2, the destination receives 100 packets in expectation. This implies that ETX fails to capture the path reliability. The situation is similar if we further consider limited retransmissions and node unreliability. Therefore, routing based on path-ETX does not give us the optimal delivery path in terms of the data yield per transmission.

In this chapter, I comprehensively investigate the unreliability in both links and nodes. I propose QoF, a new metric which estimates the chances for a packet to pass both a link and a node. The link-QoF not only considers the transmission cost at the sender but also considers the data delivery ratio at the receiver. The node-QoF estimates the quality of forwarding within a node, and it plays an important role in differentiating the lossy nodes. Based on link-QoF/node-QoF, we aggregate the QoF measure over a path (path-QoF). The path-QoF metric estimates the intact path forwarding quality and it considers both transmission cost and end-to-end data delivery ratio. The QoF metric measures the expected transmissions for a successful transmission over a path. Hence using such a metric can greatly improve the data yield while having a low transmission overhead.

The contributions of this chapter are summarized as follows.

First, I reveal the limitations of existing link-based indicators like ETX and ETF in estimating the intact path forwarding quality. In a practical system, routing selection based on ETX often leads to severely degraded data yield.
Second, I propose a new metric QoF to measure the path quality. QoF can be used to estimate the forwarding quality over a link or within a node. Using QoF, we are able to characterize both the transmission cost and the data delivery ratio along a forwarding path.

Third, I implement QoF estimation with TinyOS 2.1 [116, 117] and incorporate it in current CTP implementation. I evaluate the QoF based routing performance in a test-bed consisting of 50 TelosB nodes [118]. The results show that using QoF metric improves the data yield while it reduces the per-successful delivery cost.

5.1 Observations on Packet Losses in WSNs

5.1.1 Basic Observations

![Network yield at different scales. (b) Packet loss on different nodes.](image)

Figure 5-2: Observation from the 330-node GreenOrbs deployment:

(a) Network yield at different scales. (b) Packet loss on different nodes.

In this section, I present the outdoor test-bed results over 330 nodes that motivate my work. As used in [115], I use network *yield* to measure the quality of data collection of the network. The network yield measures the quantity of data received at the sink with respect to the total data generated by all nodes in the network. The network yield can be calculated by

$$\text{yield} = \frac{\# \text{ of data pkts received at the sink during } w}{\# \text{ of data pkts sent by all nodes during } w}$$

The network yield gives us the goodput of the network, reflecting both forwarding reliability and the throughput.

During the measurement, I vary the network size. The network yield for different network sizes is shown in Figure 5-2 (a). We see from the statistics that there are about 22~40% data lost in the multi-hop data collection when the network scales from 100 to 330. I further look
into the lost packets. The packet loss on different nodes is shown in Figure 5-2 (b). I find that packet loss is common on different nodes while there are a small portion of nodes which greatly suffer from the packet loss. The packet loss here is mainly due to two reasons. The first reason is transmission timeout on the links (exceeding the retransmission threshold); and the second reason is local packet drops within the node which are mainly due to receive/transmit queue overflow, memory corrupt, routing loops, duplicated packet and program bugs or race condition and etc. I will take a close look at the causes of packet loss at individual sensor nodes.

5.1.2 Anatomy of Packet Loss

![Figure 5-3: The work flow of packet forwarding on a sensor node](image)

(1) Packet loss within a node

To deeply understand the packet loss I observe within the sensor node, I take a close look into the work flow on a node that forwards a packet. As depicted in Figure 5-3, the flow starts with perceiving the physical wave that carries the packet from the sender. It ends when the forwarding node gets ACK from the next-hop receiver or the number of retransmissions reaches the limit.

We can see from Figure 5-3 that there exists a *gray zone* in the work flow, spanning across the MAC, network, and application layers. Existing link/path estimators only measure the forwarding quality of packets outside the *gray zone*. The packet is presumed passing through the gray zone 100% successful. According to aforementioned observations, however, there is indeed a high probability for a packet to get lost when it is processed in the *gray zone*.

When a packet is successfully received and verified, it is buffered at the chips and a packet arrival event is signaled to upper layers. At this stage, there is a risk that the forwarding queue in the network layer overflows, i.e., no buffer space is available to restore
the incoming packet. Besides, the signal event itself may fail. Both cases result in packet drops. After the packet is accepted to the network layer, it will first be handled by the arrival event handler. Duplicate suppression is applied such that if a packet has been received twice because of loops or multiple transmissions it will be discarded. Then CTP will allocate buffer for the incoming packet. If the buffer is used by other packets, this packet will be dropped. After that, the packet is delivered to application layer. The message might get corrupted or made oversize by the program, which result in packet drops. After the application layer processing, the packet is delivered back to the network layer and a send task will be posted. If the task is already in the task queue, the subsequent post of the task will fail, and the packet gets lost. Before the packet is send to the next hop node, CTP checks whether the parent node is still available. When sending the packet, the MAC layer sender will be requested and the send operation will fail if the packet is oversize or the sender is too busy to accept new packets (common for high data rate as observed in the GreenOrbs). After the packet is sent and ACK is received, the buffer storing the packet in the network layer needs be freed. The failure to free the buffer space will result in memory unreleased.

Indeed, a number of TinyOS/CTP software bugs have been reported in practical systems [119-121]. As a typical example, a CTP+FTSP send crash problem was reported in TinyOS-devel mailing list. The problem occurs when FTSP sends a message via the radio driver specific TimeSyncMessageC component, which appended the embedded timestamp and called ActiveMessageC directly. While the ActiveMessageC was busy transmitting the message and the CTP tries to transmit its own message via the AMSenderC, then the AMQueueImplP returned the unexpected FAIL. This failure causes a CTP node hang forever, acting as a problematic node which can potentially absorb a large volume of traffic without forwarding it. T-check[120] recently reported a bug that CTP fails to update the route due to misconfiguration of a parameter.

There are some other possible error instances that lead to packet loss. For example, there are 12 buffers allocated in the forwarding queue in CTP. With high traffic pressure, the packets received at a node may exceed its processing capacity, resulting in a queue overflow. Once the 12 buffers have been occupied, successive packets get dropped at the node.

As reported in [35, 122], the sensor hardware becomes more unstable at a higher temperature. The atrocious weather and environment may also increase the chance of packet losses on a relaying node.
Figure 5-4: Causes of packet loss on sensor nodes

I conduct experiment on a 50-node test-bed. In Figure 5-4, I summarize the occurrence of several causes that lead to packet drops. The receiver queue overflow is due to the resource constraint on sensor nodes. The packets duplicate suppression is due to the fact that routing layer information is not timely updated. Task fail is caused by the OS mechanism that does not allow the same task to be posted twice if the former one is not finished. The sendDone failure is due to the mismatch of number of successfully send operations and number of sendDone events.

The above anatomy reveals us that overlooking the existence of the gray zone inside the sensor node is likely to cause mismatch between the link/path estimation and the real forwarding capacity of the path. Thus we shall carefully estimate such an unreliable factor in the gray zone and use it as an important indicator to evaluate “good” paths for data delivery.

(2) Packet Loss over a Link

Another observation is that the transmission timeout is common and accounts for a large portion of packet drops. For a 100-hour data set collected from the 330-node network, the transmission timeout accounts for 61.08% of all packet drops. Figure 5-5 shows the CDF of the nodes that suffer from packet drops due to transmission timeout. This is the result collected from a network with a relatively high retransmission threshold of 30 in CTP. For lower values of the retransmission threshold, the problem will be more severe. Such a fact implies that existing ETX metric based on unlimited retransmissions and absolute link reliability is inaccurate in estimating the path quality.

ETX, which works well when end-to-end delivery is presumed fully reliable, may not be necessarily accurate in estimating the path quality for low retransmission threshold.
5.2 QoF: Comprehensive Path Quality Measurement

Current metrics for route selection have either of the following two limitations. (1) They only consider the transmission cost without necessary consideration of the data delivery ratio along a forwarding path. (2) They only consider packet losses over the links while overlooking packet drops on a forwarding node. To address these limitations, I propose QoF, Quality of Forwarding, which is a metric for comprehensive characterization of the forwarding paths. Specifically, it captures both the transmission cost and the data delivery ratio on the paths.

5.2.1 Generic Link Model

I use a generic link model as the basis of QoF design. A generic link from A to B in the model represents either a physical link or a traversing path of the packet on a node (so called a virtual link). When it is a physical link, A and B are the corresponding sending and receiving nodes. When it is a virtual link, A and B respectively denote the starting (packet received) and ending (packet successfully sent) points of packet forwarding on a relaying node.

Associated with the link AB, there are two factors.

1) The link quality $q$, which denotes the probability of a packet to successfully go through the link.

2) The limit of retransmissions $r$. The sender is allowed to retransmit the packet for at most $r$ times before giving up.

Then the packet delivery ratio on a generic link is

$$\text{PDR} = 1 - (1 - q)^{r+1}, \quad (5-1)$$
Equation (5-1) indicates the probability of a packet to successfully go through the link with quality $q$.

Note that the physical and virtual links comprise the entire path of packet delivery from source to destination. I go on to examine the PDR on either type of links.

**PDR over a physical link.** For a physical link, PDR is calculated by Equation (5-1) where $q$ is equal to the link’s PRR and $r$ is equal to the limit of MAC layer retransmissions.

**PDR over a virtual link.** Similarly, PDR on a virtual link is given by Equation (5-1), where $q$ denotes the forwarding quality of a relaying node. No retransmission is executed on a node, i.e., $r = 0$. The PDR on a virtual link, i.e. a relaying node, actually means the probability of a received packet to be successfully forwarded by the node. For a typical sensor node, PDR here the procedure starting at the point where the packet is passed upwards from the MAC layer and ending at the point where the packet is passed downwards to the MAC layer, whereafter the packet is ready to be transmitted to the next-hop receiver.

PDR on a virtual link indeed characterizes the forwarding quality of a node. For example, if a problematic node receives a large number of packets but forwards none of them, its PDR is equal to 0. By checking PDR of virtual links, one can avoid choosing such problematic nodes for packet forwarding.

**Impact of retransmissions.** Clearly, when infinite retransmissions are employed, packet delivery on a link is expected to be 100% reliable. We have $PDR = 1$ in Equation (5-1). On the other hand, considering some real-time applications yielding data streams, zero retransmission is tried. A packet will be permanently dropped once it fails to reach the receiver for the first time. Let $r = 0$ in Equation (5-1). PDR degrades to $q$, which is the same as the link quality.

### 5.2.2 QoF of a Generic Link

In order to consider both the transmission cost and the packet delivery ratio, I define QoF of a generic link as ratio of the data delivery ratio to the transmission cost.

$$QoF = \frac{PDR}{ETC}$$

where ETC denotes the expected transmission count for a distinct packet over the link, and is calculated by

$$ETC = \sum_{i=1}^{r} kq(1-q)^{k-i} + r(1-q)^r = \frac{1-(1-q)^{r+1}}{q}$$
Therefore,

\[ \text{QoF} = \frac{\text{PDR}}{\text{ETC}} = q \]

Note that \( \text{QoF}^{-1} \) is equal to the expected transmission count for a successful transmission on the link. For a single link, it is equal to the value of ETX. This is reasonable because QoF and ETX both choose the link with the highest quality.

\[ \begin{align*}
\text{PDR}_{n-1} & \quad \text{PDR}_{n-2} \\
\text{PDR}_{n,n-1} & \quad \text{PDR}_{n->1}
\end{align*} \]

Figure 5-6: QoF computation

5.2.3 QoF of a Forwarding Path

The advantages of QoF, compared to the existing metrics, become apparent when we consider a multi-hop forwarding path.

As shown in Figure 5-6, I use the following notations.

- \( \text{QoF}_n \) is the QoF for the n-hop path.
- \( \text{PDR}_{n,n-1} \) is the PDR over the link from node \( n \) to node \( n-1 \).
- Without causing confusion, \( \text{PDR}_n \) denotes the PDR on node \( n \).
- \( \text{ETC}_{n,n-1} \) is the ETC over the link from node \( n \) to node \( n-1 \).
- I do not count the ETC within a node because it does not incur actual communication cost so that it does not affect the transmission cost for the whole path.

\( \text{QoF}_n \) is calculated in a hop-by-hop recurrence as follows.

\[ \text{QoF}_n = \frac{\text{PDR}_{n->1}}{\text{ETC}_{n,n-1} + \frac{\text{PDR}_{n->1}}{\text{QoF}_{n-1}}} \]

where

\[ \text{PDR}_{n->1} = \text{PDR}_{n,n-1} \times \text{PDR}_{n-1,n-1} \times \text{PDR}_{1} \]

In the above equations, \( \text{PDR}_{n->1} \) represents the data delivery ratio along the forwarding path, considering packet losses over all the links as well as within all the forwarding nodes. The denominator represents the number of transmissions that is actually counted in
communication cost. It is the sum of two parts. The first part $ETC_{n,n-1}$ is the expected transmission count over the link from node $n$ to node $n-1$. The second part $PDR_{n-1}/QoF_{n-1}$ represents the transmission cost incurred during the first $n-1$ hops. Again, I do not include the ETC on the nodes, since there is no transmission cost within a node.

Now I look back to the example shown in Figure 5-1, the following are three scenarios to examine the QoF for forwarding paths:

**Scenario 1.** Assume $r = 0$ and all nodes have PDR = 1 (nodes will never drop packets).

For path $P_1$, we have $PDR_{3,2} = 1/10$, $PDR_{2,1} = 1/10$, and $PDR_{3,1} = PDR_{3,2} \times PDR_{2} \times PDR_{2,1} = 1/100$. The expected transmission count for link between node 2 and 3 can be computed as $ETC_{3,2} = 1$, and $ETC_{2,1} = 1$. The path QoF for $P_1$ is $QoF_2 = PDR_{2,1}/ETC_{3,2} = 1/10$ and $QoF_3 = PDR_{3,1}/(ETC_{3,2} + PDR_{3,1}/QoF_2) = 1/110$, which means a successful end to end delivery incurs 110 transmissions for $P_1$.

For path $P_2$, similarly we have $PDR_{3,2} = 1$, $PDR_{2,1} = 1/19$, and $PDR_{3,1} = 1/19$. $ETC_{3,2} = 1$ and $ETC_{2,1} = 1$. $QoF_3 = 1/38$, which means a successful end to end delivery incurs 38 transmissions for $P_2$.

This scenario shows that although the paths have equal ETX values, the cost for a successful end to end delivery can be quite different. The cost for $P_1$ is about 2 times larger than the cost for $P_2$. Such difference cannot be measured by path quality estimation using ETX. By simultaneously considering the transmission cost and the packet delivery ratio QoF provides more comprehensive estimation.

**Scenario 2.** Now I consider the second scenario where $r = 0$ and there are node with PDR less than 1. For simplicity, we assume only one node with PDR value less than 1. If $PDR_2 = 1/2$, we have $QoF_3 = 1/57$ on $P_2$ using the similar calculation as that in Scenario 1. If $PDR_1 = 1/2$, $QoF_3 = 1/76$ for path $P_2$. This scenario first shows that the node’s forwarding quality indeed affects the transmission cost. Even for paths with the same path PDR, bad link which resides on different position of the path influence the path quality differently and thus the cost of a successful packet delivery can be quite different. Such an impact is also overlooked in path quality estimation using ETX. QoF captures the impact of nodes’ forwarding quality and link quality for path quality estimation.

**Scenario 3.** In this scenario, we assume $r = 1$ for all links and PDR = 1 for all nodes. For $P_1$, $QoF_3 = 19/1190$. For $P_2$, $QoF_3 = 37/1064$, which is about two times of $P_1$’s QoF, while originally QoF of $P_2$ is almost 3 times of that of $P_1$. Such a result further implies increasing retransmission count can improve the path quality while the improvement brought by retransmissions appears to be diverse with different paths.
The computation of $Q_{n}$ depends on $PDR_{n \rightarrow 1}$, $ETC_{n,n-1}$, and $Q_{n-1}$. $ETC_{n,n-1}$ can be measured locally. $PDR_{n \rightarrow 1}$ can be computed from $PDR_{n-1 \rightarrow 1}$, $PDR_{n \rightarrow 1}$ and local measurement of $PDR_{n,n-1}$. $Q_{n-1}$ can also be obtained from previous hop. A node can compute a QoF value for all neighbor nodes and uses the largest one as the final QoF value to the destination.

5.3 Implementation

My implementation is based on TinyOS 2.1 in NesC. I implement the QoF and incorporate it in state-of-the-art data collection protocol, CTP. The hardware platform is the TelosB mote with MSP430 MCU and CC2420 radio.

![Diagram](image)

**Figure 5-7: Integrating QoF in CTP**

![Diagram](image)

**Figure 5-8: Component diagram**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>setAmType</td>
<td>Set the AM type for corresponding traffic</td>
</tr>
<tr>
<td>setRetrans</td>
<td>Set the retransmission threshold for link PDR</td>
</tr>
<tr>
<td>IncIncoming</td>
<td>Add an incoming packet to traffic monitor</td>
</tr>
<tr>
<td>IncOutgoing</td>
<td>Add an outgoing packet to traffic monitor</td>
</tr>
<tr>
<td>getNodePDR</td>
<td>Return the PDR of the node</td>
</tr>
<tr>
<td>getLinkPDR</td>
<td>Return PDR for a specified link</td>
</tr>
</tbody>
</table>

The overview architecture is shown in Figure 5-7. In order to incorporate the QoF metric in CTP, I use the QoF estimator, QofC, to interact with CTP for path selection. The QoF estimator is based on PDR estimators for the node (NodePdrC) and for the link (LinkPdrC). It combines the node and link PDR provided by PDR estimators for path quality computation and provide QoF information to the routing protocol for path selection. The link PDR
The estimator is built upon the four-bit link estimator. The interface provided by the PDR estimators is shown in Table 5-1, and the interface provided by the QoF estimator contains only one command, getQoF(), with the path QoF as the return value.

### 5.3.1 Measuring PDR within a Node

The PDR within a node can be computed as

\[
PDR = \frac{\text{outCtr}}{\text{inCtr}}
\]

where inCtr is the number of packets received by NodePdrC during a packet window of length w, outCtr is the number of packets that is passed down to NodePdrC for transmission. As shown in Figure 5-8, the component, NodePdrC, sits above the MAC layer and below the networking layer, and it is responsible for monitoring the incoming traffic and outgoing traffic across the two layers. When NodePdrC receives a packet from the MAC layer, it increases the incoming traffic counter (inCtr), and then forwards the packet to the upper layer. When NodePdrC receives a packet from the upper layer, it increases the outgoing traffic counter (outCtr), and then transfers the packet to the lower layer for actual transmission.

Since NodePdrC is used to monitor the forwarding capability of a node, it should exclude packets destined for the node (also broadcast packets) since those packets would not be further forwarded to other nodes. Hence, NodePdrC counts the packets of particular types that can be designated by application programmers. I use the AM type field of TinyOS packets for monitoring traffic and check if they are the type of packet I am interested in so that unnecessary packets such as broadcast packets can be eliminated.

In order to accurately measure the potential loss at the upper layers, we need to count the unique incoming packets and unique outgoing packets. We should count the unique incoming packets because the networking layer will usually forward the same packets only once (e.g. by duplicate detection in CTP). Also, we should count the unique outgoing packets because the same packet can be either considered as transmitted or dropped. To achieve this goal, I modify the routing architecture of CTP. The functions of duplicate detection and packet retransmissions are moved to a separate layer, RetransC, which is directly above the TinyOS CSMA MAC layer. I put the NodePdrC layer above the RetransC layer so that I can obtain accurate information about duplicate receptions and packet retransmissions.

There is a packet window of size w for recording whether the incoming packets have been passed down to the MAC layer for transmission. The short-term PDR value, PDRw, is calculated according to the number of incoming and outgoing packets in the current window.
The long-term PDR value is calculated as follows. \( PDR = (1 - \alpha) \times PDR + \alpha \times PDR_w \). I set \( \alpha = 9/10 \) in my current implementation. Since the node PDR estimation requires a sufficient number of incoming packets, it will not work well if there is few incoming traffic. In order to address this issue, I actively inject packets by a component, InjectorC, which is above the CTP component. I cannot inject packets directly at the NodePdrC layer because I need to effectively measure packet loss at upper layers. The InjectorC component is triggered when there are few incoming packets. It periodically injects packets to the networking layer, i.e. CTP in my case.

5.3.2 Measuring PDR over a Link

In order to measure the PDR over a link, I first need to measure the PRR over a link. Then I use the specified link retransmission threshold (=r) to obtain the link PDR as follows: \( PDR = 1 - (1 - PRR)^{r+1} \). The LinkPdrC component provides link PDR estimations, relying on state-of-the-art link estimation methods (for estimating link PRR). In my current implementation, the link PRR estimation is provided by the four-bit link estimation component in the TinyOS distribution.

The four-bit estimator uses information from 3 layers encoded by 4 bits. It first uses physical signal strength to filter out the link with very low signal strength. The ACK bit is used to denote whether a packet is acknowledged. The ACK bit can be further used to calculate the PRR of a link. The compare bit in the network layer indicates whether a better link exists in the routing table and the pin bit is used to fix an entry in the routing table. Combining those 4 bits, the estimator efficiently measures the link quality.

5.3.3 QoF Calculation

To calculate the QoF of the path, each node broadcasts a packet with the QoF and PDR fields to its neighbors. The QoF field is the QoF value from the broadcaster to the root and the PDR field is the node PDR value of the broadcaster. The receiver can use the information in the received packet to calculate the QoF (to the root) through the broadcaster, as I describe in Section 5.2. The beacon interval of the broadcast is controlled by a Trickle timer [107], which increases the time interval exponentially when the network is steady and decreases the interval to the minimum when there is new information for update.
5.3.4 Collaborative Reporting

Normally, each node actively reports its QoF to the neighboring node. Sometimes, however, some nodes cannot actively report due to software bugs (observed in the GreenOrbs application). To address this issue, I use an additional Collaborative Reporting (CR) mechanism to let some neighboring nodes report the QoF on behalf of the problematic node.

When a node does not receive report messages from a neighbor node for a predefined period of time (e.g., 10min), the neighbor node will be considered as problematic and the CR mechanism is triggered. The QoF of the problematic node is locally calculated as the link layer traffic destined for it (of particular AM types) divided by the outgoing traffic transmitted by the problematic node.

As described in Subsection 5.3.1, however, we should only count the unique incoming packets and the unique outgoing packets. The packet duplication of the neighbor node can be locally filtered using the similar method in Section 5.3.1. I also adopt a suppression mechanism in CR for preventing heavy collisions in QoF and PDR reporting:

When a node overhears that another node is reporting QoF for a problematic node, it simply suppress its reports for the problematic node.

5.4 Evaluation

In this section, I evaluate effectiveness of my design extensively. Section 5.4.1 introduces the evaluation methodology. Section 5.4.2 examines how QoF improves the routing performance. Section 5.4.3 reveals further observations on QoF. Section 5.4.4 summarizes the experimental results.

Figure 5-9: The picture of test-bed


5.4.1 Methodology

I use a network consisting of 50 TelosB nodes to evaluate the efficacy of my design. Figure 5-9 depicts the testbed. I integrate QoF with CTP (CTP-QoF) for evaluating the performance of QoF in supporting the routing. In the network, I set the power level of the transmissions to 1 to construct a multi-hop network. I mainly compare CTP-QoF with the original CTP protocol (CTP-ETX) in following two cases.

Case I: streaming application case. In this case, I set the retransmission threshold to 1 and the transmission frequency per node to 3Hz. In such a case, I explore the performance of QoF in supporting data streaming applications that pursue low latency and high traffic throughput without reliability guarantee on individual packets.

Case II: real-world deployment case. In this case, I set the retransmission threshold to 30 and the transmission frequency per node to 3Hz, which is the same with my settings in the real-world deployment. Based on my experience with GreenOrbs, I introduce some “problematic” nodes that artificially drop a portion of incoming packets. Some faulty nodes can still report QoF to its neighboring nodes while other nodes keep silent all the time.

I use three key metrics to compare CTP-QoF and CTP-ETX.

Data yield: the percentage of successfully received packets at the sink.
Transmission cost: the total number of packets transmitted in the network.
Normalized transmission cost: the number of transmissions normalized by the data yield.

5.4.2 Performance Comparison

I present the experimental results in this section, comparing the performance of CTP-QoF and CTP-ETX.

(1) Data Yield

I first investigate the improvement of using QoF to the data yield in the network. In this experiment, I use the setting in case I and case II. Figure 5-10 depicts the node yield of CTP-QoF and CTP-ETX for case I. The node yield gives the data yield of a specified node. I find from Figure 5-10 that most nodes have a high node yield in CTP-QoF than nodes in CTP-ETX. There are only four nodes (ID 9, 12, 41, 45) which have lower node yield. I find that link qualities of those nodes in CTP-QoF are very low and among all outgoing links the node cannot find an alternative path to avoid packet drops. The average improvement is around 12%. The improvement of CTP-QoF in case I is mainly due to the fact that as
retransmission threshold is low, PDR of links is low and thus packets are likely to be dropped over links. This type of packet drops cannot be captured by ETX estimation. The packet drops due to limited retransmissions affect the entire path and such effect, while is not quantified by existing method, can be captured by the QoF metric.

Figure 5-10: Comparison of node yield for CTP-ETX and CTP-QoF for case I with high traffic pressure and low retransmission threshold

![Figure 5-10: Comparison of node yield for CTP-ETX and CTP-QoF for case I with high traffic pressure and low retransmission threshold](image)

Figure 5-11: Comparison of node yield for CTP-ETX and CTP-QoF for case II with problematic nodes

Figure 5-11 shows the node yield of CTP-ETX and CTP-QoF for case II. Those faulty nodes randomly drop 30% packets it receives. Again I find from Figure 5-11 that almost all nodes in CTP-QoF have a higher node yield than those in CTP-ETX. This is because that QoF measures the forwarding qualities of both the nodes and the links. I find from Figure 5-11 that
the yield of some nodes in CTP-ETX is even only one third of that in CTP-QoF. I look into the experimental data. I find that with ETX estimation, most faulty nodes exist on the optimal data delivery paths chosen by CTP-ETX. In CTP-QoF, however, most of the faulty nodes are excluded from the optimal routing paths. For example, as shown in Figure 5-12 (a), I select node 20 near the faulty nodes on the test-bed and investigate its behavior. Links from 20 to node 88, 89, 14, 15 are shown in the Figure 5-12 (a). Here node 88 and node 99 are two faulty nodes. Figure 5-12 (b) shows the ETX values of node 20 to 4 neighbors. By using the ETX values, the node cannot distinguish faulty nodes from normal nodes. However, Figure 5-12 (c) shows that path passing the faulty node has a low QoF. Using QoF values can distinguish those faulty nodes.

![Figure 5-12: Path-ETX and path-QoF from node 20 through node 14, 15, 88, 89](image)

(a) Topology (b) ETX (c) QoF
I further investigate the behaviors of faulty nodes in comparison with the normal nodes in CTP-QoF. The normal nodes and faulty nodes are selected close to each other such that they have similar external conditions.

I find from Figure 5-13 a clear trend that in CTP-QoF the incoming traffic for faulty nodes does not accumulate much, as the QoF information help to choose paths that avoid the faulty nodes of much lower forwarding quality. On the other hand, CTP-ETX overlooks the internal problems on the faulty nodes and chooses delivery paths simply according to the link estimation. As shown in Figure 5-14, the incoming traffic on faulty nodes grows similarly with the traffic on normal nodes. It implies that other nodes are blind of the packet loss inside the faulty nodes and keep sending packets to them, resulting in degraded routing performance.
(2) Transmission Cost

In this set of experiments, I evaluate the total data transmissions incurred within the entire network for CTP-QoF and CTP-ETX. I do experiments for both cases I and II. Figure 5-15 (a) shows the network traffic cost for case I and Figure 5-15 (b) shows the network traffic cost for case II. In both two cases, CTP-QoF saves nearly 30% transmission cost compared with CTP-ETX.

(3) Normalized Transmission Cost

In this section, I evaluate the average transmissions for a successful E2E delivery (i.e. normalized transmission cost), which the QoF metric tries to minimize. Figure 5-16 compares CTP-QoF with CTP-ETX in both cases I and II. The average cost for the end-to-end data delivery in CTP-QoF is much less than the cost in CTP-ETX. For case I, the CTP-QoF
reduces the average cost/yield by 28%. For case II, the CTP-QoF reduces the average
cost/yield by 34%. This is mainly because of the more comprehensive estimation on the path
quality in CTP-QoF.

![Figure 5-17: CDF of node PDR for low traffic](image)

5.4.3 Observations on QoF

In order to further explore the detailed reasons for the improvement in CTP-QoF, I
investigate the details of PDR on node. I test the network at high traffic pressure (3
packets/s/node) and (low traffic 0.3 packet/s/node) at different retransmission thresholds (r=1
and r=30).

(1) Low Traffic Pressure

When the network traffic is low, most nodes are with a high forwarding quality, as
suggested in Figure 5-17. This is because the main source of packet loss is the packet drop
due to retransmission timeout. Figure 5-17 shows the PDR on nodes for low traffic. I find that
with low traffic, the PDRs on nodes with both high retransmission and low retransmission
thresholds are relatively high. For the network with low traffic and low retransmission
threshold, almost all nodes are with a PDR value close to 1. This can be explained as for low
traffic pressure, nodes are less likely to suffer from resource constraint, e.g., the queue
overflow is less likely to happen. In contrast, some nodes with low retransmission threshold
have a slightly higher PDR than nodes with high retransmission threshold. This may be due to
that for low retransmission threshold, nodes will more easily discard the packets due to
retransmission timeout. Fewer packets accumulate within the nodes and therefore the packet
loss is less likely to happen.
(2) High Traffic Pressure

Figure 5-18 shows the PDR on nodes with high traffic pressure. I find that there are more nodes with low PDR values when the traffic pressure is high. For the 330-node network with high traffic and high retransmission threshold, there are about 40% of nodes with PDR lower than 60%. This is mainly because more packets accumulate within the nodes and thus more likely a node suffers from resource constraints (e.g., receive queue overflow). The experiment data also validate that a main source of packet drop inside node is receive queue overflow.

5.4.4 Summary of Results

The above experimental results reveal the following findings.

QoF can effectively capture the data yield while minimizing the transmission cost. In the application cases I & II, QoF based routing improves the performance of ETX-routing in terms of data yield, transmission cost, and normalized transmission cost.

While link layer retransmission can improve link PDR, it will decrease node PDR as retransmission consumes system resources. The retransmission threshold should be carefully chosen so that we can obtain a satisfactory PDR along a path.

QoF will be more effective when the node PDR is rendered less reliable, e.g. under high traffic pressure and with high retransmission threshold. For example, Figure 5-18 shows that QoF can be effective for network with high traffic. Figure 5-17 shows that QoF is effective for high retransmission while the traffic is low.
6 Canopy Closure Estimates with GreenOrbs

Canopy closure, defined as the percentage of ground area vertically shaded by overhead foliage [123], is widely utilized as a critical indicator of the condition of a forest ecosystem. As illustrated in Figure 6-1, canopy closure in a forest refers to the ratio of the area shaded by the trees to the area of the entire ground. Having many significant uses for ecosystem management and disaster forecast, canopy closure estimates, however, are practically non-trivial [123-126]. Most existing approaches suffer the scalability problem, due to the limited measurement capacity and high costs. The status of canopy closure largely depends on the weather (e.g. the effect of strong winds) and can be highly dynamic. The measurement procedures are also restricted by the subjectivity of the surveyor, the landform, and the undergrowth. As a result, conventional approaches can only provide inaccurate estimates.

To address this issue, I present GreenOrbs [45], a wireless sensor network (WSN) system in the forest, and its application for canopy closure estimates. In GreenOrbs, a number of commercial off-the-shelf sensor motes are programmed, enclosed, and deployed in the forest. With every node equipped with light, temperature, and humidity sensors, GreenOrbs supports various ecological applications. In this chapter, I focus on the application for canopy closure estimates. My contributions are summarized as follows:

1. I propose a sensor network design that enables accurate canopy closure estimates using inexpensive sensors randomly deployed in the forest;
2. I propose a technique to calibrate the light sensors and discriminate the states between light versus shade;
3. I design light-weight mechanisms for node state monitoring, which reduces communication overhead;
4. I present a detailed evaluation of GreenOrbs and compare it with conventional forestry methods. The results demonstrate the advantages of WSN techniques and their great potential benefits by introducing WSNs into the traditional forestry field.

6.1 Application Background

Canopy closure is a valuable forest inventory factor, which plays an important role in ecosystem management. It is actually a fundamental factor to define a forest. The current global definition of forest is an area of land that is more than 0.5 hectares with more than 10% canopy closure [127]. Canopy closure is being seen as more important recently in the sense of environmental protection. For instance, it is a key indicator of the water conservation capacity of a forest. When logging in a forest with non-uniform vegetation, the canopy closure of different areas in the forest must be considered to maintain sustainable development. In a process of urban greening, canopy closure is useful in characterizing the forest stand structure.

In ecological forestry and agriculture, canopy closure is used to estimate various indexes, such as penetration of light to the understory, absorbance of carbon dioxide, and release of oxygen, which are closely associated with photosynthesis. Continuous measurements of canopy closure reflect the growth of vegetation, and thus can be used to assist ecological planning in forestry and agriculture. Moreover, regulations for certain regional wildlife species require maintenance of certain levels of canopy cover. Real-time data of canopy closure can be used to construct precise computational models of rain interception, so that people can better predict disasters like floods and mud-rock flows.

Despite the significance of canopy closure, the usage of canopy closure is often restricted due to the lack of accurate and efficient measurement approaches. The existing approaches fall into two categories: ground measurement and aerial measurement.

With ground measurement, canopy closure is manually estimated by people or with auxiliary devices. Approaches like ocular estimate, line-intercept, and crown mapping fall into this category. Ocular estimate and line-intercept are traditional approaches using simple inexpensive instruments [123, 124]. Ocular estimate relies on an experienced surveyor conducting a long training process. Line-intercept estimates canopy closure by the percentage of shaded length on a line placed on the ground. It is efficient for estimating the profile of large crowns but often fails to capture the interspaces among the leaves inside a crown, due to the limited breadth of a line. Crown mapping [126] maps the crowns of trees with a spherical densitometer or a vertical point sampling device. Canopy closure can thus be estimated by
scanning and processing the maps of crowns. Crown mapping requires careful mounting of
devices, and may yield detailed estimates, while it suffers poor scalability due to the
inconvenient deployment and prohibitive measuring cost. Fisheye lenses that are able to take
hemispherical photographs can be used to measure the canopy of a very small area but cannot
measure a vast forest.

Ground measurement approaches have two common limitations: First, various factors can
often interfere in the estimated results, such as the subjectivity of the surveyor, the landform,
and the undergrowth. Second, they can only measure a small portion of forest, and thus lack
scalability in large-scale applications.

In order to eliminate the ground measurement limitations, aerial measurement and
satellite imaging are proposed for canopy closure estimates [124, 125]. Aerial measurement
estimates the canopy closure based on bird's-eye view photos of a forest. Since the photos are
taken from the air at limited heights, the results are often overestimated due to the
non-vertical visual angles on a large canopy. The quality of photos is highly dependant on the
weather. Photos taken on a sunless day have poor contrast, while strong sunlight results in
high reflection from the vegetation. Regardless, both cases lead to poor estimates of canopy
closure.

Satellite imaging is the latest proposal for canopy closure estimates, but has not yet
reached a mature approach. It overcomes the drawbacks of aerial measurement, but still
suffers many difficulties. For example, in a satellite image, it is hard to accurately
discriminate the canopy from the undergrowth.

The accuracies of aerial measurement and satellite imaging are further affected by the
artificially constructed computation models. The resulting estimates usually have large
variations.

6.2 Theoretical Foundation

Canopy closure estimates with GreenOrbs are based on the Monte Carlo Theory [128].
Throughout this chapter, the real canopy closure is denoted by \( R \). Suppose a number of
sensors \( V(n) \) with size \(|V(n)|=n\) are randomly deployed at sampled locations, I partition the
sensors into two sets \( V_{\text{light}} \) and \( V_{\text{shade}} \), where \( V_{\text{light}} \cup V_{\text{shade}} = V \) and \( V_{\text{light}} \cap V_{\text{shade}} = \emptyset \).
Abusing notations, I use \( V \) instead of \( V(n) \). Then in practice \( R \) is estimated as \( R_V \), where

\[
R_V = |V_{\text{shade}}| / |V|
\]
In this section, I disclose the relationship between the accuracy of $RV$ and the number of randomly sampled sensors $n$. These will serve as the guideline in parameter selections in my experiments and deployments.

First, I give the following well known lemma which is useful in my further proof:

**Lemma 6-1 (Binomial Distribution):** Consider $n$ independent variables $X_i$,  

$$X_i \in \{0,1\}, \ p = \Pr(X_i = 1), \ X = \sum_{i=1}^{n} X_i \ ,$$

then

$$\Pr(X > \varepsilon) < \frac{e^{(1-p)}}{(e^{-n \cdot p})^2} \ \ \text{when} \ \ \ varepsilon > n \cdot p$$

$$\Pr(X \leq \varepsilon) \leq \frac{(n-varepsilon)p}{(n \cdot p - e)^2} \ \ \text{when} \ \ 0 < \varepsilon \leq n \cdot p$$

The following theorem discloses the relationship between the accuracy of $RV$ and the number of sensors $n$. Specifically, I use the ratio between $RV$ and $R$ as a metric to measure how accurate $RV$ is.

**Theorem 6-2:** Under the above system settings,

$$\Pr(c_1 < \frac{RV}{R} \leq c_2) \geq 1 - (\theta_1 + \theta_2) \cdot \frac{1}{n}$$

where $0 < c_1 \leq 1$ and $c_2 > 1$ are some constant and

$$\theta_1 = \frac{c_1(1-R)(c_1-1)^2}{R} \cdot \theta_2 = \frac{1-c_2 R}{(1-c_2)^2 R} \cdot$$

**Proof:** According to the property of Lemma 6-1, $X$ is a binomially distributed random variable, $n$ is number of trials, and $p$ is success probability. In my specific case, $X = |V_{shade}|$ is the number of sensors in the shade, $n$ is total number of randomly deployed nodes, and $p$ is the probability of a node being in the shade. Note that $p = R$, according to the definition of canopy closure, it follows that,

$$\Pr(|V_{shade}| > c_2 R \cdot n) > \frac{c_2 R \cdot n(1-R)}{(Rn-c_2 Rn)^2} \ \ \text{when} \ c_2 > 1,$$

$$\Pr(|V_{shade}| \leq c_1 R \cdot n) \geq \frac{c_1 R \cdot n(1-R)}{(Rn-c_1 Rn)^2} \ \ \text{when} \ 0 < c_1 \leq 1.$$

$$\Pr(|V_{shade}| \leq c_1 R \cdot n) = \Pr(R_V \leq c_1 R),$$

$$\Pr(|V_{shade}| > c_2 R \cdot n) = \Pr(R_V > c_2 R).$$

Then together with the property of union probability, we immediately get

$$\Pr(c_1 < \frac{RV}{R} \leq c_2) \geq 1 - (\theta_1 + \theta_2) \cdot \frac{1}{n}$$
where \( \theta_1 = \frac{1-c_4R}{(c_1-1)^2 R} \), \( \theta_2 = \frac{c_2(1-R)}{(1-c_2)^2 R} \).

Theorem 6-2 represents the ideal case when every sensor can be placed into its corresponding subset (either \( V_{\text{light}} \) or \( V_{\text{shade}} \)) correctly. Such an assumption, however, does not always hold in practice, due to the uncertainties in sensor readings and environmental dynamics (e.g. the sunlight). This is one of the major challenges I address in the following subsections. Nevertheless, Theorem 6-2 is useful to estimate the best attainable result with \( n \) sampled sensors.

### 6.3 Design with GreenOrbs

![Diagram of GreenOrbs]

Figure 6-2: Work flow of GreenOrbs includes four main phases: training, deployment, operation, and data translation

Figure 6-2 illustrates the entire work flow of GreenOrbs. Prior to the deployment, a training process (Section 6.3.1) is conducted on all the sensor nodes to analyze the correlations among their readings and obtain a linear discrimination model to discriminate their states. The enclosed sensor nodes are then installed into the forest. During the daily operations (Section 6.3.2), a node keeps monitoring its own state after determining its initial state, according to the guidelines from training. Whenever the node state changes, it will inform the sink of the change. By filtering the environmental noise, the sink translates the collected data into estimates of canopy closure (Section 6.3.3).

The following key issues need to be addressed in the design of GreenOrbs. First, the sensors in GreenOrbs have diverse instrumental errors in their readings. We need to devise an effective calibration method and a model which best separates the sensors into two subsets, \( V_{\text{light}} \) and \( V_{\text{shade}} \). Second, considering the energy constraints on the nodes, we need an energy-efficient mechanism for state monitoring and data collection. Third, considering the varying solar altitude, a universal model is required to translate the measured percentage of sensors in the shade into an estimated canopy closure. The design of GreenOrbs mainly...
consists of three components: pre-deployment training, online data processing, and sink-side data translation.

![Figure 6-3: Simultaneous readings of 21 sensors that are placed under different illuminance in the forest](image)

![Figure 6-4: Simultaneous readings of five sensors that are placed under the same illuminance](image)

6.3.1 Pre-deployment Training

The training process yields the guidelines of online sensory data processing, namely calibration and node state discrimination. Accordingly, the training process is divided into two phases, correlation analysis and linear discrimination analysis.

(1) Correlation Analysis

Sensor readings are intrinsically error-prone. Due to the diverse instrumental errors, the raw sensor readings without calibration often result in incorrect measurements. I conduct an
observational experiment to disclose the importance of calibration. I first place 21 sensors at different locations (some in the light and the others in the shade) when the environmental illuminance is 114.75Klux. Here environmental illuminance is defined as the illuminance at a location without any shade, denoting the maximum illuminance in the environment. Figure 6-3 shows the perceived illuminance of the sensors. The maximum difference between any two readings is 8.41KLux.

I then place the 21 sensors under exactly the same illuminance and let them sense the illuminance once every second. Note that the illuminance varies with time. The readings of five typical sensors are plotted in Figure 6-4. Surprisingly, the maximum difference between two sensor readings is around 6Klux. I take more rounds of the experiments and obtain similar results, indicating that the instrumental errors of sensor readings cannot be overlooked. Unless the sensor readings are appropriately calibrated, we cannot correctly estimate canopy closure.

Figure 6-4 indicates another fact that the instrumental errors on each individual sensor are consistent over time. Thus an intuitive guess comes into my mind: *Are the readings of sensors linearly correlated with each other? Yes.* To validate this answer, I take a randomly selected sensor in the above experiment as the reference and analyze the correlation between the reference and any other node.

There are a total of 21 sensors in the experiment, denoted by $S_0, S_1, S_2, ..., S_{20}$. Let $n$ denote the total number of readings a sensor produces. $k$ ($1 \leq k \leq n$) denotes the counter of sensor reading. $N_k$ denotes the reading of the reference sensor $S_0$ while $M_{ik} (i=1, 2, ..., 20)$ denotes the reading of $S_i$. The Pearson product-moment correlation coefficients [129] between $S_0$ and $S_i$ are calculated using Equation (6-1).

$$
 r_i = \frac{n \sum_{k=1}^{n} M_{ik} N_k - \sum_{k=1}^{n} M_{ik} \sum_{k=1}^{n} N_k}{\sqrt{n \sum_{k=1}^{n} M_{ik}^2 - (\sum_{k=1}^{n} M_{ik})^2} \sqrt{n \sum_{k=1}^{n} N_k^2 - (\sum_{k=1}^{n} N_k)^2}}
$$

(6-1)

The calculation results are shown in the second column of Table 6-1, which validate that the sensor readings are almost linearly correlated. Taking $S_0$ as the reference, we can thus calibrate the readings of all the other sensors through linear transformations. Let $M'_{ik}$ denote the calibrated reading of $S_i$. The linear transformation from $M_{ik}$ to $M'_{ik}$ is

$$
 M'_{ik} = a_i M_{ik} + b_i
$$
called *calibration formula*, where $a_i$ and $b_i$ are the calibration coefficients for $S_i$. To minimize the instrumental error differences between $S_0$ and $S_i$, $a_i$ and $b_i$ need to satisfy the following conditions. First, the expectation of $M'_{ik}$ equals that of $N_k$.

$$\frac{1}{n} \sum_{k=1}^{n} (a_i M_{ik} + b_i) = \frac{1}{n} \sum_{k=1}^{n} N_k \quad (6-2)$$

Second, $a_i$ and $b_i$ minimize the deviation of $M'_{ik}$ to $N_k$.

$$(a_i, b_i) = \arg \min_{a,b} \sum_{k=1}^{n} ((a_i M_{ik} + b_i) - N_k)^2 \quad (6-3)$$

**Table 6-1: Correlation coefficients between $S_0$ and $S_i$ and the calibration coefficients**

<table>
<thead>
<tr>
<th>Node ID ($i$)</th>
<th>$r_i$</th>
<th>$a_i$</th>
<th>$b_i$ (KLux)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.9645</td>
<td>1.09</td>
<td>-7.76</td>
</tr>
<tr>
<td>2</td>
<td>0.9512</td>
<td>1.10</td>
<td>-12.93</td>
</tr>
<tr>
<td>3</td>
<td>0.9525</td>
<td>1.22</td>
<td>-25.62</td>
</tr>
<tr>
<td>4</td>
<td>0.9543</td>
<td>1.19</td>
<td>-21.98</td>
</tr>
<tr>
<td>5</td>
<td>0.9329</td>
<td>1.21</td>
<td>-22.83</td>
</tr>
<tr>
<td>6</td>
<td>0.9421</td>
<td>0.99</td>
<td>0.67</td>
</tr>
<tr>
<td>7</td>
<td>0.9412</td>
<td>1.03</td>
<td>-2.60</td>
</tr>
<tr>
<td>8</td>
<td>0.9585</td>
<td>0.88</td>
<td>14.23</td>
</tr>
<tr>
<td>9</td>
<td>0.9478</td>
<td>0.94</td>
<td>7.14</td>
</tr>
<tr>
<td>10</td>
<td>0.9666</td>
<td>1.07</td>
<td>-5.50</td>
</tr>
<tr>
<td>11</td>
<td>0.9635</td>
<td>1.12</td>
<td>-14.36</td>
</tr>
<tr>
<td>12</td>
<td>0.9637</td>
<td>1.01</td>
<td>-2.48</td>
</tr>
<tr>
<td>13</td>
<td>0.9555</td>
<td>1.18</td>
<td>-18.41</td>
</tr>
<tr>
<td>14</td>
<td>0.9555</td>
<td>1.11</td>
<td>-11.04</td>
</tr>
<tr>
<td>15</td>
<td>0.9780</td>
<td>1.04</td>
<td>-3.77</td>
</tr>
<tr>
<td>16</td>
<td>0.9498</td>
<td>0.91</td>
<td>8.68</td>
</tr>
<tr>
<td>17</td>
<td>0.9443</td>
<td>0.91</td>
<td>10.89</td>
</tr>
<tr>
<td>18</td>
<td>0.9363</td>
<td>1.22</td>
<td>-22.04</td>
</tr>
<tr>
<td>19</td>
<td>0.9283</td>
<td>0.98</td>
<td>2.46</td>
</tr>
<tr>
<td>20</td>
<td>0.9700</td>
<td>0.95</td>
<td>3.94</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>0.9528</td>
<td>1.06</td>
<td>-6.17</td>
</tr>
<tr>
<td><strong>Standard Deviation</strong></td>
<td>0.0129</td>
<td>0.11</td>
<td>12.16</td>
</tr>
</tbody>
</table>

Solving Equations (6-2) and (6-3) yields the value of $a_i$ and $b_i$ as follows.

$$a_i = \frac{\sum_{k=1}^{n} M_{ik} \sum_{k=1}^{n} N_k - n \sum_{k=1}^{n} M_{ik} N_k}{(\sum_{k=1}^{n} M_{ik})^2 - n \sum_{k=1}^{n} M_{ik}^2} \quad b_i = \frac{\sum_{k=1}^{n} N_k - a_i \sum_{k=1}^{n} M_{ik}}{n}$$
The last two columns of Table 6-1 list the calibration coefficients. Through a similar training process, I obtain the calibration formulas of all sensors to be deployed. The training results are omitted here due to the page limit. Subsection 6.4.1 has more details on the effectiveness of calibration.

(2) Linear Discrimination Analysis

A sensor cannot determine its status merely based on the perceived illuminance, even it is calibrated. For example, I observe that at one time a sensor in the shade perceived illuminance of 106.3KLux, while at another time a sensor in the light perceived illuminance of only 102KLux. The discriminating point between light and shade actually depends on the environmental illuminance. It requires both the sensor’s calibrated reading and the current environmental illuminance to get an accurate determination of the sensor’s current state. Hence, in the second phase of the training, I derive a linear discrimination model from a comprehensive training data set. Node states in the set are recorded through manual observations. The corresponding calibrated sensor readings are then collected at the sink. The reference sensor is placed under the sun to perceive the environmental illuminance. As shown in Figure 6-5, Point \((x, y)\) denotes a sensor reading of \(y\) when the environmental illuminance is \(x\). Clearly, there exists a separating boundary between \(V_{\text{light}}\) and \(V_{\text{shade}}\). I use the method of Least Linear Squares to derive the linear discrimination model, denoted by \(Y = d_0 + d_1X\). We have

\[
\phi = \sum (Y_i - d_0 - d_1X_i)^2
\]

Figure 6-5: The linear discrimination model
\[
\frac{\partial \phi}{\partial d_0} = -2 \sum (Y_i - d_0 - d_i X_i) = 0 \tag{6-4}
\]

\[
\frac{\partial \phi}{\partial d_1} = -2 \sum (Y_i - d_0 - d_i X_i) X_i = 0 \tag{6-5}
\]

Solving Equations (4) and (5) with the training data set, we get \(d_0=17.45, d_1=0.81\). Using this model, a node is able to exactly determine its current state under any environmental illuminance.

Note that as shown in Figure 6-5, the discrimination model produces a very small portion of false judgments when the environmental illuminance is over 115KLux. I have more discussions on this problem in Subsection 6.3.2.

6.3.2 Online Data Processing

The calibration formulas and the linear discrimination model are loaded into all the nodes before deployment. Since the nodes all perceive nearly zero illuminance at night, canopy closure estimates are conducted only in the daytime. Every day sensors start operation at 8:30 a.m. and switch off at 15:30 (I explain in Subsection 6.3.3 why I set such a measurement period).

Whenever in operation, the raw readings are calibrated once they are perceived. Without special declaration, the sensor readings refer to the calibrated ones.

(1) Initial State Determination

The initial node states are determined based on the linear discrimination model. As soon as GreenOrbs starts in operation, nodes exchange their initial readings through gossip [130]. Every node forwards the highest reading it obtains (either its own reading or a reading from the others). The gossip process converges in \(O(d)\) time where \(d\) is the network diameter.

At the end of gossip, all the nodes possess an identical highest reading, which denotes the current environmental illuminance, denoted by \(X_0\). According to the discrimination model, the value separating \(V_{light}\) and \(V_{shade}\) is calculated by \(Y_0=d_0+d_1 X_0\). Let \(y_0\) denote the initial reading of Node \(V_i\). Its initial state is determined as follows: If \(y_0 > Y_0\), \(V_i\) is in the light. Otherwise, \(V_i\) is in the shade.

(2) Node State Monitoring and Collection

The sensing frequency of nodes in GreenOrbs is set at once per minute. Considering the degree of dynamics in the forest, such a sensing frequency is sufficient to capture all possible changes in the environment at relatively low cost.
A straightforward but inefficient method (called naive method) to monitor the node state is to let each node periodically send every reading directly to the sink. Such a method obviously incurs a huge amount of network traffic. Besides, considering the multi-hop transmission in a large WSN, concurrent data collection from all the nodes will probably cause congestion over bandwidth-constrained wireless links, which will lead to frequent packet loss and extra retransmission cost [103].

A possible minor improvement is to execute distributed state monitoring (called gossip-based method), as I do in the initial state discrimination. Nevertheless, it requires a gossip process in the whole network to propagate the environmental illuminance in each period and still incurs considerable network traffic.

GreenOrbs realizes a light-weight mechanism for node state monitoring and data collection, which overcomes all the above drawbacks. During the whole measurement period after the initial state discrimination, a node updates its state to the sink node only when it detects a state transition from light to shade, or vice versa. I have observed that the course of a state transition lasts at least five minutes. Hence I set the periodical interval of state monitoring as five minutes. At the end of every interval, a node calculates the average of the latest five readings. Let $I$ and $I'$ denote the current average and the average of the previous interval, respectively. The varying rate of readings is calculated by $V_{rate} = (I - I')/5$. The unit is KLux/min.

Note that the varying rate of sensor readings is not directly associated with the node state. A large variation in the perceived illuminance does not necessarily imply a state transition; however, a state transition always corresponds to a large variation in the perceived illuminance on a node. The ultimate goal of my algorithm is to identify the large variations of sensor readings which indeed cause state transitions.

If the variation is caused by the swaying of leaves, the phenomenon is usually transitory so the node state stays unchanged over a relatively long period. One can filter such noises by averaging the latest sensor readings. That’s why I set the interval of state monitoring at five minutes. If the variation is caused by reflection or refraction (the change in direction of light when it irradiates or passes through the vegetation), it is really a problem that my current design cannot solve. I leave it for future work.

Great variations in environmental illuminance (i.e. the sunlight) also cause large variations in the perceived illuminance on a sensor node. For example, a floating cloud sometimes blocks out the sunshine. In this case, the perceived illuminance on the nodes greatly varies, but actually the node states stay unchanged.
Figure 6-6: The pseudo code of node state monitoring

The real cause of node state transitions is the change of the solar altitude. In this case, only a small portion of the nodes, namely the nodes lying near the boundary of the light and shaded areas, change their states. In this case, we would expect the $V_{rate}$ of such a node to differ significantly from the average $V_{rate}$ of its neighbors. This is the key idea behind my algorithm, shown in Figure 6-6.

Figure 6-6 shows the pseudo code of node state monitoring. I define two thresholds for the detection of state transitions, namely $S_1=6$KLux/min and $S_2=5$KLux/min. The thresholds are obtained from practical observations on node state transitions. If $V_{rate}$ exceeds $S_1$, it
means the readings vary greatly and triggers the node to check whether the state has changed. Through one-hop communications, the node compares its \( Vrate \) with the \( AverageVrate \) of its neighbors. If their difference exceeds \( S_2 \), it indicates that the node state has changed. Otherwise, the node state remains unchanged and probably the environmental illuminance has varied greatly.

(3) Discussion

Here I have a brief discussion on the effectiveness of the mechanism for node state monitoring.

First, except for the daily starting phase, node state monitoring is based solely on local information and the data from its one-hop neighbors. Compared to the naive and gossip-based methods, it saves a large amount of energy cost.

Second, according to the observation, the daily frequency of node state transitions is 2.25 times on average, of which I have more details in Section 6.4. A daily measurement period includes \( 7 \times 60/5 = 84 \) intervals of state monitoring. Assuming that node transitions are uniformly distributed over the course of a day, then on average less than 3% of the nodes change their state at every interval. Hence the concurrency of data transmissions is significantly reduced and the data collection process becomes essentially asynchronous. Consequently, GreenOrbs avoids congestion over the links and further enhances the energy efficiency.

Third, as I mention in Subsection 6.3.1, the linear discrimination model produces a very small portion of false judgments when the environmental illuminance is over 115KLux. In GreenOrbs, however, this model is used only for initial state discrimination. Another fact is the environmental illuminance in the early morning is never over 110KLux. Therefore, those false judgments are avoided.

6.3.3 Sink-side Data Translation

Canopy closure is calculated using the collected states of all the nodes. Recall that canopy closure is defined as the percentage of ground area “vertically” shaded by overhead foliage. Here I define the solar altitude, denoted by \( \alpha \) (\( 0 \leq \alpha \leq 90 \degree \)), as the angle between the direction of sunlight and the horizontal plane. Note that \( \alpha \) is a time-varying parameter. Observed in our deployment area, \( \alpha \) is usually less than 83\degree and varies with the seasons and the time of a day. Hence, the calculated percentage of sensors in the shade cannot be directly regarded as canopy closure of the forest. Calculated results at different times cannot be merged, either.
introduce a translation model, with which the measured percentage of sensors in the shade ($R_V$) is translated into an estimate of canopy closure $R$. I further define $S_p$ as the area of the crown’s vertical projection on the ground, and $S_\alpha$ as the area of the shade when the solar altitude is $\alpha$.

$$R = R_V \times \frac{S_p}{S_\alpha} \quad (6-6)$$

![Figure 6-7: Tree shadows with varying solar altitude](image)

![Figure 6-8: Vertical and non-vertical shades of a conic crown](image)

![Figure 6-9: The case of non-overlapping tree shadows](image)

Here I need to compute $S_p/S_\alpha$. As conventionally assumed in forestry [131], the crown of a tree can be modeled as a certain stereo shape, depending on the tree species. Figure 6-7 depicts the shadows of different trees at different times of a day, which indicates that given the tree species and the solar altitude, the tree shade is uniquely determined. Although this is
an approximate model for trees, I show through the performance evaluation that estimates based on this modeling indeed have satisfactory accuracies.

Without loss of generality, I use the conic crown as an example. Figure 6-8 illustrates a conic crown, its vertical projection, and its shade when the solar altitude is $\alpha$. We have $S_p = \pi r^2$ and

$$S_\alpha = r^2 \left(\pi - \arccos \frac{r \tan\alpha}{h}\right) + \frac{r^2}{\tan(\arccos \frac{r \tan\alpha}{h})},$$

where

$$S_\alpha = \pi r^2, \text{ when } \frac{\pi}{2} - \arctan \frac{r}{h} \leq \alpha \leq \frac{\pi}{2}$$

The tree species and the solar altitude at a given time of day are relatively static information about a forest. The sink node stores such information as offline data. By substituting the above results into Equation (6-6), the sink node translates the measured $R_V$ into an estimate of canopy closure $R$. Actually $S_p/S_\alpha$ is computable given the tree species and slope of the ground [66]. We assume horizontal ground here for ease of discussion.

Note that in theory, tree shadows possibly overlap each other. Here I briefly discuss this issue. Figure 6-9 plots two neighboring trees, where $d$ is the distance between two neighboring trees, $h$ is the height of the trees, $h'$ is the height of the crown, and $r$ is the radius of the crown. Then the necessary condition for non-overlapping tree shadows is

$$d - r + (h - h') \cot \alpha \geq h \cot \alpha,$$

that is

$$\alpha \geq \arctan \frac{h'}{d - r} \quad (6-7)$$

Figure 6-10: The enclosure of mote and the deployment area
Note that we assume the two trees are the same shape and size, which is the case in a normal forest. I use such an assumption just to deduce the necessary condition, for example in our deployment, \( d \approx 4\text{m}, \; r \approx 1.6\text{m}, \; h' \approx 2.8\text{m} \). According to (7), \( \alpha \geq 49.3^\circ \). This condition is basically satisfied from 8:30 a.m. till 15:30 everyday in the summer. Thus I set this period as the daily measurement period. Using a similar method, we may also deduct the measurement period for any given type of forest.

### 6.4 Evaluation

Figure 6-10 shows the enclosure of sensor mote and the deployment area of GreenOrbs, where I carried out experiments to evaluate the performance of my design.

![Figure 6-11: Calibrated readings of 5 nodes under the same illuminance](image)

![Figure 6-12: Standard deviations of sensor readings before and after calibration](image)

#### 6.4.1 Effectiveness of Design

This section evaluates the effectiveness of the design including calibration, initial state discrimination, and node state monitoring.

1. **Calibration**
Figure 6-11 shows the calibrated readings of five typical sensors, corresponding to Figure 6-4. After calibration, the five sensor readings stay nearly consistent under various environmental illuminances. For further comparison, I calculate the standard deviations (STD) of all sensor readings under the same illuminance. These data are collected in the training process. Figure 6-12 plots the cumulative distributions of STD before and after calibration. We can see that the variations among sensor readings are significantly reduced through calibration. As a fundamental element in my design, calibration guarantees that different sensors, although having diverse instrumental errors, produce almost the same readings as long as they are under the same illuminance.

Table 6-2: Confusion matrix of initial state discrimination

<table>
<thead>
<tr>
<th>Result of discrimination</th>
<th>Real state</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In light</td>
</tr>
<tr>
<td>In light</td>
<td>0.998</td>
</tr>
<tr>
<td>In shade</td>
<td>0.002</td>
</tr>
</tbody>
</table>

(2) Accuracy of Initial State Discrimination

Every morning, I organized a group of surveyors to manually annotate the initial states (i.e. in the light or in the shade) of each sensor for 20 consecutive days. Altogether 1000 node states were recorded. They are regarded as the real initial sensor states and compared with the results of initial state discrimination collected at the sink node. Table 6-2 is the corresponding confusion matrix. The accuracy of initial state discrimination is very high. The slightly higher false rate in judging the sensors in the shade was caused by the reflection and refraction from the surroundings. While they are an infrequent phenomenon, I will address them in my future work.

Figure 6-13: Daily frequency of node state transitions

(3) Node State Monitoring
Figure 6-13 illustrates the daily frequency of node state transitions in GreenOrbs. The average frequency is 2.25 times per day, which indicates that state transitions are quite infrequent in nature. As a consequence, a large portion of the energy costs can be saved if a node only updates its state to the sink node when a state transition happens. Based on the above observation, I conduct trace-driven simulations with 100 nodes to evaluate the communication cost (measured by the average number of daily radio messages) on a node using different state monitoring methods, namely the naive method, the gossip-based method, and GreenOrbs’ light-weight method.

**Table 6-3: Communication cost of state monitoring, measured by the average number of daily radio messages**

<table>
<thead>
<tr>
<th>Method</th>
<th>Initial state discrimination</th>
<th>State updates</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive</td>
<td>0</td>
<td>289.2</td>
<td>289.2</td>
</tr>
<tr>
<td>Gossip-based</td>
<td>0</td>
<td>266.8</td>
<td>266.8</td>
</tr>
<tr>
<td>GreenOrbs</td>
<td>6.5</td>
<td>91.7</td>
<td>98.2</td>
</tr>
</tbody>
</table>

Note that the naive and gossip-based methods do not need initial state discrimination, while state updates involve both the process of detecting state transitions and the process of sending the data to the sink. The simulation results in Table 6-3 show that GreenOrbs greatly reduces the communication costs by 66.0% and 61.2%, compared to the other approaches respectively.

### 6.4.2 Estimates of Canopy Closure

**Figure 6-14: Canopy closure map of the deployment area**

This section presents the evaluation of canopy closure estimates with GreenOrbs and compares it with other conventional approaches. I further evaluate the impact of miscellaneous factors, such as the environmental illuminance, dynamics, and sample size.

(1) Overall Results
Figure 6-14 shows the canopy closure map of the area where GreenOrbs was first deployed. The map is manually drawn by experienced surveyors. Note that a point in the map has only binary states (in the light or in the shade). The varying color depths represent macroscopic canopy closure in different local areas. In other words, Figure 6-14 is an intuitive display of canopy distribution and does not represent the ground truth.

The real canopy closure is measured via a manual process: The surveyors identify every tree shade and profile it on the ground (usually on a large piece of white cloth or paper placed on the ground under the tree). By collecting all the cloth (or paper), canopy closure is then given by the ratio of the aggregated area inside the profiles to the total area of the forest. That means the surveyors need to carefully measure the area inside the profile on each piece of cloth (or paper). Obviously, this is a labor-intensive job and can only be used in a relatively small forest to obtain the ground truth.

Table 6-4: Results in the first deployment *

<table>
<thead>
<tr>
<th>Day</th>
<th>9:00</th>
<th>10:00</th>
<th>11:00</th>
<th>12:00</th>
<th>13:00</th>
<th>14:00</th>
<th>15:00</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.49</td>
<td>0.47</td>
<td>0.41</td>
<td>0.41</td>
<td>0.45</td>
<td>0.46</td>
<td>0.48</td>
</tr>
<tr>
<td>2</td>
<td>0.50</td>
<td>0.46</td>
<td>0.44</td>
<td>0.43</td>
<td>0.42</td>
<td>0.45</td>
<td>0.46</td>
</tr>
<tr>
<td>3</td>
<td>0.51</td>
<td>0.45</td>
<td>0.48</td>
<td>0.44</td>
<td>0.41</td>
<td>0.46</td>
<td>0.48</td>
</tr>
<tr>
<td>4</td>
<td>0.50</td>
<td>0.47</td>
<td>0.41</td>
<td>0.44</td>
<td>0.43</td>
<td>0.42</td>
<td>0.46</td>
</tr>
<tr>
<td>5</td>
<td>0.54</td>
<td>0.49</td>
<td>0.47</td>
<td>0.45</td>
<td>0.45</td>
<td>0.50</td>
<td>0.52</td>
</tr>
<tr>
<td>6</td>
<td>0.52</td>
<td>0.46</td>
<td>0.46</td>
<td>0.42</td>
<td>0.45</td>
<td>0.48</td>
<td>0.44</td>
</tr>
<tr>
<td>7</td>
<td>0.45</td>
<td>0.44</td>
<td>0.43</td>
<td>0.39</td>
<td>0.44</td>
<td>0.44</td>
<td>0.48</td>
</tr>
<tr>
<td>8</td>
<td>0.49</td>
<td>0.42</td>
<td>0.38</td>
<td>0.41</td>
<td>0.41</td>
<td>0.41</td>
<td>0.47</td>
</tr>
<tr>
<td>9</td>
<td>0.47</td>
<td>0.44</td>
<td>0.44</td>
<td>0.40</td>
<td>0.44</td>
<td>0.46</td>
<td>0.52</td>
</tr>
<tr>
<td>10</td>
<td>0.51</td>
<td>0.44</td>
<td>0.41</td>
<td>0.45</td>
<td>0.44</td>
<td>0.44</td>
<td>0.48</td>
</tr>
<tr>
<td>11</td>
<td>0.51</td>
<td>0.46</td>
<td>0.43</td>
<td>0.42</td>
<td>0.43</td>
<td>0.42</td>
<td>0.47</td>
</tr>
<tr>
<td>12</td>
<td>0.48</td>
<td>0.50</td>
<td>0.48</td>
<td>0.45</td>
<td>0.44</td>
<td>0.46</td>
<td>0.48</td>
</tr>
<tr>
<td>13</td>
<td>0.52</td>
<td>0.41</td>
<td>0.46</td>
<td>0.42</td>
<td>0.44</td>
<td>0.45</td>
<td>0.48</td>
</tr>
<tr>
<td>14</td>
<td>0.52</td>
<td>0.45</td>
<td>0.44</td>
<td>0.44</td>
<td>0.46</td>
<td>0.48</td>
<td>0.47</td>
</tr>
<tr>
<td>15</td>
<td>0.50</td>
<td>0.46</td>
<td>0.42</td>
<td>0.43</td>
<td>0.46</td>
<td>0.45</td>
<td>0.47</td>
</tr>
<tr>
<td>16</td>
<td>0.50</td>
<td>0.42</td>
<td>0.43</td>
<td>0.40</td>
<td>0.44</td>
<td>0.48</td>
<td>0.50</td>
</tr>
<tr>
<td>$S_p/S_a$</td>
<td>0.83</td>
<td>0.97</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.97</td>
<td>0.86</td>
</tr>
<tr>
<td>$R$</td>
<td>0.41</td>
<td>0.44</td>
<td>0.44</td>
<td>0.43</td>
<td>0.44</td>
<td>0.44</td>
<td>0.41</td>
</tr>
</tbody>
</table>

* Each of the 7*16 cells (9:00~15:00, Day 1 ~ Day 16) denotes a measured percentage of sensors in the shade. Each cell in the second last row denotes an estimate of canopy closure at the corresponding time of day, with respect to the time-varying translation parameters ($S_p/S_a$). The last row is the average of the data in the second last row, denoting the overall estimate of canopy closure.

Table 6-4 lists the measurement results in 16 consecutive days in the first deployment. Indeed, I observed in July 2008 that the maximum difference among the solar altitudes in 16
days does not exceed 1°. Thus I use an identical solar altitude in calculating canopy closure. I changed the node locations everyday during the 16 days. Everyday the new coordinates of every node were regenerated on a PC. According to the generated data, I went into the forest and moved the nodes to their new locations. To give a clear picture, I only include the statistical data on the seven hours.

According to Equation (6-6), I calculate the canopy closure at different hours as shown in the second last row. Taking the average, I get the overall estimated canopy closure as 0.43. The real canopy closure is 0.44. Compared to the real canopy closure, the relative error of GreenOrbs’ estimate is only 2.27%.

Note that the estimated results at 10:00, 11:00, 12:00, 13:00, and 14:00 are very accurate, while the estimates at 9:00 and 15:00 are a bit smaller. This is due to the partially overlapping shadows of the trees. Recall the necessary condition in Section 6.3.3, the tree shadows do not overlap when $\alpha \geq 49.3^\circ$. On the other hand, $\alpha = 45^\circ$ at 9:00 and 15:00, which leads to slightly overlapping tree shadows in some places. Thus $S_\alpha$ is overestimated and the estimated results become slightly smaller.

In order to examine the accuracy of GreenOrbs’ estimates, I further divide the entire deployment area into two sub-regions as shown in Figure 6-14, which apparently have different canopy closures. Table 6-5 compares the real and estimated canopy closures with GreenOrbs. When the canopy closure gets larger, the estimation accuracy becomes slightly higher too. This is consistent with my theoretical conclusion in Section 6.2.

<table>
<thead>
<tr>
<th></th>
<th>Real Canopy Closure</th>
<th>Estimated Canopy Closure</th>
<th>Relative error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region 1</td>
<td>0.368</td>
<td>0.353</td>
<td>4.1%</td>
</tr>
<tr>
<td>Region 2</td>
<td>0.512</td>
<td>0.506</td>
<td>1.2%</td>
</tr>
<tr>
<td>Overall</td>
<td>0.44</td>
<td>0.43</td>
<td>2.27%</td>
</tr>
</tbody>
</table>

(2) Comparison

Now I compare GreenOrbs with the conventional approaches for canopy closure estimates, such as ocular estimates, line-intercept, crown mapping, and satellite imaging. The ocular estimates are carried out by experienced surveyors 10 times. The line-intercept estimates are conducted three times, by placing a line along three different directions on the forest ground. The crown mapping estimate is conducted only once, because of the prohibitive measuring cost. Estimates with digital photography are conducted nine times based on nine different sample areas in the forest. The result of GreenOrbs corresponds to the overall estimate in Table 6-4.
Figure 6-15: Estimated results with different approaches

Table 6-6: Definition of illumination index of a day

<table>
<thead>
<tr>
<th>Weather</th>
<th>Illumination index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine</td>
<td>1</td>
</tr>
<tr>
<td>Cloudy</td>
<td>2</td>
</tr>
<tr>
<td>Showery</td>
<td>4</td>
</tr>
<tr>
<td>Rainy</td>
<td>8</td>
</tr>
</tbody>
</table>

Figure 6-15 compares the estimated results of all the above methods to the real canopy closure. The maximum errors of each method are marked as well, if applicable. We can see that GreenOrbs outperforms all the other methods with the best accuracy and the highest consistency. Clearly, crown mapping has comparable accuracy with GreenOrbs, while it is not scalable in large-scale measurement due to the prohibitive cost.

(3) Impact of Environmental Factors

In this subsection, I evaluate the impact of environmental illuminance on the estimation accuracy. For this purpose, I define the illumination index of a day as shown in Table 6-6. If there are two or more states of weather in a day, the illumination index is calculated as the average of multiple indexes. For example, a cloudy to showery day has an illumination index of 3.

Figure 6-16 plots the daily estimated results and illumination indexes of 16 consecutive days. Interestingly, I see strong correlation between them. Specifically, GreenOrbs achieves more accurate estimates on sunny days, because the nodes in the shade differ more distinctly from the nodes in the light in stronger sunlight. When a node state transition happens, the perceived illuminance on the sensor also presents a larger varying rate. Consequently, the accuracy of GreenOrbs’ state monitoring algorithm is higher, too.
(4) Impact of Sample Size

Now I perform offline analysis based on the collected data to evaluate the impact of sample size on the estimation accuracy. Everyday during the first GreenOrbs deployment, I randomly assigned the node locations, the number of sampled locations (denoted by $N$) represented by the collected data set is much larger than the system size ($n$).

For each sample size, I conduct 105 times the random samplings from the whole collected data set and therefore obtain 105 estimates of canopy closure. Figure 6-17 shows the cumulative distribution of relative errors. The results demonstrate that GreenOrbs is able to provide satisfactory estimates, even when I use only 50 samples. The average relative error is 4.69% when $N=50$. Meanwhile, the estimation accuracy increases when the sample size is
increased. However, the benefit from involving more samples is slight, especially when the sample size exceeds 200. When $N=200$ and $N=800$, the average relative errors are 3.12% and 2.97%, respectively.

Here I give a brief summary of the evaluation results. I have validated the design of GreenOrbs. Specifically, calibration eliminates the diverse instrumental errors among different sensors and the results of the initial state monitoring are very accurate. The light-weight mechanism for state monitoring saves the communication cost. The results clearly demonstrate that GreenOrbs outperforms all the conventional methods. It produces accurate and consistent estimates of canopy closure with various parameter settings, such as different canopy closure, varying solar altitude, different environmental illuminance, and different sample sizes. The impacts of the above factors are also carefully assessed.
7 Conclusions

7.1 Summary

WSNs have been proposed since a decade ago. Despite of the substantive research, engineering, and industrial efforts that have been put into this field, there are still a number of issues left to explore. Under the circumstances of global climate changes and environmental pollution, we launch the GreenOrbs project to address the fundamental challenges and explore the potential design space in long-term large-scale WSNs.

I lead the implementation and deployment of GreenOrbs systems, including two outdoor WSNs and an indoor testbed. The system scale by far is 330 nodes and the duration of the longest deployment has reached one year. GreenOrbs incorporates the techniques in both fields of WSNs and forestry, and plays as the foundation to support various forestry applications as well as research on WSNs.

In the infrastructural level of WSNs, my work to address the issue of localization includes two phases. The first approach PI, which is regarded as the preliminary attempts to address the ranging errors in localization, observes the monotonicity of RSSI and utilizes it in a mobile-assisted localization process. The CDL approach, which is motivated by the real-world observations on GreenOrbs, inherits the advantages of both range-free and range-based methods. It keeps pursuing better ranging quality throughout the localization process. Both PI and CDL have been implemented with real systems. The evaluation results demonstrate that they indeed improve the accuracy, efficiency, and applicability of localization in complex environments.

In the network layer, data collection is a basic task for WSN applications. In the light of many existing schemes and protocols of link estimation and routing, I reveal the limitations of existing link-based estimation methods like ETX and ETF in measuring the forwarding quality of intact paths, based on the real-world experience from GreenOrbs. My proposal called QoF is a new metric which comprehensively estimates the chances for a packet to be successfully forwarded, through either a link or a node. I have implemented QoF based on TinyOS 2.1 and incorporate it in the current CTP implementation. The performance of QoF-based routing is evaluated in the GreenOrbs test-bed. The results demonstrate that using QoF as the metric of routing path quality enhances the network throughput and efficiency of packet delivery.
In the application field, based on the GreenOrbs, I conduct study on introducing WSN technology into forestry measurements. My work focus on the classical but crucial application called canopy closure estimates. My proposal enables large-scale canopy closure estimates using inexpensive sensors randomly deployed in the forest. Specifically, I design appropriate techniques for calibration, computation, and communication, which make my proposal superior to conventional forestry methods with respect to cost and accuracy.

7.2 Future Work

The experiences in the system implementation, deployments, and application study with GreenOrbs enable me to foresee broader design space and more opportunities in my future research work.

I plan to continue my study that aims at further improving the applicability and manageability of WSNs for forestry applications. With regard to applicability, we may conduct interdisciplinary studies to investigate better solutions, such as application-driven query optimization, data fusion, and detection of data outlier. Such solutions are expected to be more adaptive and suitable to the scenario of forestry applications.

As for the manageability, it generally involves the available techniques for testing, debugging, and diagnosing a WSN. The state of arts in the WSN field is still far from being mature. My research plan on manageability includes two aspects. On one hand, I plan to design and develop a more powerful testing and debugging software framework for WSN systems. The existing solutions are mostly restricted to function above the network layer. Hence, the key issue left is how to acquire sufficient environmental information and integrate such featured factors into the testing and debugging process.

On the other hand, I plan to conduct study on the visibility problem of WSNs. Lack of visibility to in-network information is a major challenge to WSN management. The state of arts mainly resort to collecting the network information as much and efficient as one can, and executing diagnosis on top of the collected data. My viewpoint is that the lack of visibility can be reduced from the side of protocol designs. It is non-trivial challenge because by far there is not a widely recognized metric of network visibility. Thus it is worth studying how to define, quantify, and evaluate the visibility of a WSN running certain protocols.

When the WSN for forestry application become relatively mature, I also plan to extend the use of WSN technology into other fields that really demand it, such as disasters forecast, structural monitoring, health care, and industrial automation. The success in any of those mentioned fields will make significant progress in that field and bringing remarkable benefit
to people and the society. To push it one step further, I believe in the near future when WSNs are successfully applied in many different fields, deployed at different areas, with heterogeneous modalities, and serving different sets of people, there will be a global fabric of a number of collaborative sensor networks. Considering the developing trend in networking, internet of things is attracting increasing attention from governments, scientists, researchers, industry, and ordinary citizens. Applications with collaborative sensor networks will spawn a new cutting edge of networking and system research fields.
8 References


[42] How GPS receiver works,

[63] eXtreme Scale Mote, [www.ece.osu.edu/~bibyk/ee582/XscaleMote.pdf](http://www.ece.osu.edu/~bibyk/ee582/XscaleMote.pdf)


