Designing localized algorithms for large-scale wireless sensor networks : a geometric perspective

Li, Feng

2014

Li, F. (2014). Designing localized algorithms for large-scale wireless sensor networks : a geometric perspective. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/62064>

<https://doi.org/10.32657/10356/62064>

**DESIGNING LOCALIZED ALGORITHMS FOR
LARGE-SCALE WIRELESS SENSOR NETWORKS:
A GEOMETRIC PERSPECTIVE**

LI FENG

School of Computer Engineering
Nanyang Technological University

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy (Ph.D)

2014

To my parents, for their unconditional love and endless support.

Acknowledgement

First and foremost I offer my heartfelt thanks to my supervisor, Prof. Jun Luo, for his support, guidance, encouragement, and patience. It has been a great pleasure working with him, whose personality inspired me a lot as well as his knowledge did. I also would like to express my sincere gratitude to my co-supervisor, Prof. Ying He, for the continuous support of my Ph.D study and research, for his patience, motivation and immense knowledge.

Special thanks to my co-authors, Prof. Shiqing Xin, Prof. Wenping Wang and Prof. Xianfeng Gu, for their invaluable suggestions, and for our fruitful collaborations.

I would also like to express my sincere thanks to my labmates and friends, for their lovely company and encouragement. I always appreciate the precious moments we shared together. I also thank my colleagues and technicians in CeMNet for their patient and kind help.

Finally, My deepest gratitude goes to my family for their continuous love and support throughout my life. I owe to them every step of my progress. I must thank my wife, Huan Yang. She was always there cheering me up and stood by me through the good times and bad.

Abstract

Wireless Sensor Networks (WSNs) provide the service of monitoring and sensing physical environment. With the pervasive applications of WSNs in various fields, the increasing network scale brings considerable challenges in many aspects, from high-level applications to low-level networking and communication. In this thesis, we try to understand large-scale WSNs from geometric perspective. In particular, by exploiting the multi-fold geometric properties of WSNs (e.g., sensor nodes' locations and connectivity as well as their unique sensing/communicating models), we can build geometric models and apply geometric algorithms to address perplexing problems of large-scale WSNs. In contrast to typical geometric techniques which require global information as input, we design localized algorithms which can be performed by individual sensor nodes in a distributed manner using only locally available information, to benefit large-scale WSNs in terms of time cost, system adaptivity and robustness, energy efficiency, as well as network performance.

In many applications, a large-scale WSN is deployed to monitor and survey time-variant events, which demands a fast boundary detection technique. Considering sensor nodes may be deployed in a 3D volume, the first focus of this thesis is boundary detection for large-scale 3D WSNs. We model a 3D WSN as *point cloud* sampled from a hypothetical 3D volume. Borrowing the idea of *direct visibility*, we present an on-line boundary detection algorithm to identify the sensor nodes on the boundary surfaces of the 3D volume using only local position information. Additionally, we propose a localized parametrization algorithm to regulate the detected arbitrarily shaped network boundaries into spheres, thereby supporting other network functionalities, e.g., distinguishing internal boundaries from the external one, and geographic routing. We evaluate our strategies of boundary detection and regulation with extensive implementations and simulations.

By employing mobile sensor nodes, we are allowed to autonomously deploy a large-scale WSN in a remote and inaccessible targeted field, for example, to adaptively monitor time-variant events. On the other hand, multiple-coverage is developed to ensure the robustness and accuracy of the monitoring service. Therefore, our second topic in this thesis is to investigate k -coverage autonomous sensor deployment problem based on a general omnidirectional sensing model, with our on-line boundary detection serving as an underlying infrastructure. Specifically, we formulate the sensor deployment problem as a *space tessellation* problem, and adapt the theory of k -order *Voronoi Tessellation* to autonomously conduct mobile sensor nodes towards k -coverage with local optimality in the sense of load balancing. Apart from 2D area coverage, we also extend our deployment strategy to 3D surface coverage under the same framework, which significantly broadens the application scopes of our algorithm. Our extensive experiments deliver convincing results to confirm our theoretic analysis for our deployment strategy.

The increasing network scale also poses high demand on network throughput and transmission quality, especially under an optimized network deployment (e.g., the above load balancing multiple-coverage). Unfortunately, the performance of WSNs is jeopardized by the interference from surrounding WiFi networks. We hereby aim at maximizing network throughput with transmission quality also guaranteed, by improving the utilization of frequency and temporal resources. Inspired by *Centroidal Voronoi Tessellation*, we “tessellate” the sensor nodes in frequency-location space, to continuously allocate the spectrum less interfered by WiFi to nodes/links in a near-optimal manner, which enhances the transmission parallelism. We, on the other hand, efficiently exploit temporal resource by adaptively tune the usage of CSMA in an opportunistic fashion; thereby ensuring transmission quality while maintaining as high transmission throughput as possible in face of residual WiFi interference. We demonstrate the efficacy of our algorithms by extensive implementations on a MicaZ-based platform with different network settings.

Keywords

Wireless sensor networks, localized geometric algorithms, boundary detection, autonomous deployment, area/surface coverage, continuous frequency allocation, CSMA.

Contents

Acknowledgement	iii
Abstract	v
List of Figures	xi
List of Tables	xvii
1 Introduction	1
1.1 Background	1
1.2 Scope of Research	3
1.3 Contributions	4
1.4 Thesis Organization	6
2 Literature Review	9
2.1 Boundary Detection for Wireless Sensor Networks	9
2.1.1 Geometric or Topological	9
2.1.2 The “Pain” of Geometric Approaches	10
2.1.3 Event Boundary vs. Network Boundary	11
2.2 Area/Surface Coverage of WSNs	12
2.2.1 Area Coverage with Random Deployments	12
2.2.2 Area Coverage with Deterministic Deployments	12
2.2.3 Mobile Assisted Area/Surface Coverage	13
2.3 Improving Co-existence of ZigBee and WiFi	14
2.3.1 Channel Allocations in WSNs	14
2.3.2 Survive in WiFi Interference for WSNs	15

3	LBDP: Localized Boundary Detection and Parametrization for 3D Sensor Networks	17
3.1	Introduction	18
3.2	UNFOLD: Detecting Boundary in a “Mirror” Image	20
3.2.1	UNFOLD in a Nutshell	20
3.2.2	Transformation and Boundary Definition	22
3.2.3	On-Line Boundary Detection Algorithm	26
3.2.4	Performance Analysis	28
3.3	Localized Boundary Sphericalization (LBS) and its Applications	30
3.3.1	Boundary Sphericalization	30
3.3.2	Applications	32
3.4	Simulations and Experiments	33
3.4.1	Simulations for UNFOLD	34
3.4.2	Implementation and Experiments for UNFOLD	37
3.4.3	Simulations for LBS	38
3.5	Summary	40
4	Autonomous Deployment for Load Balancing k-Surface Coverage in Sensor Networks	41
4.1	Introduction	42
4.2	Problem Definition and Mathematical Background	44
4.2.1	System Model	44
4.2.2	Problem Formulation	45
4.2.3	High Order Voronoi Diagram	46
4.2.4	Generalization to 3D Surfaces	47
4.3	APOLLO: Localized k -Coverage Node Deployment Algorithm	50
4.3.1	Optimal Conditions	50
4.3.2	The Algorithm	51
4.3.3	Discussions	57
4.4	APOLLO on 3D Surfaces	58
4.4.1	Computing Dominating Regions	59
4.4.2	Computing Chebyshev Centers	59
4.5	Simulations	61
4.5.1	Convergence	61

4.5.2	Energy Consumption during Deployments	63
4.5.3	Energy Consumption after Deployments	64
4.5.4	Comparisons with Min-Node k -Coverage	66
4.5.5	Performance in Maximum k -Coverage	68
4.5.6	Adaptability to Obstacles	69
4.5.7	Extension to 3D Surfaces	70
4.6	Summary	72
5	ART: Adaptive fRequency-Temporal co-existing of ZigBee and WiFi Networks	73
5.1	Introduction	74
5.2	Motivations and Background	77
5.2.1	The Impact of CSMA on Throughput	77
5.2.2	Frequency-Distance Tradeoff without CSMA	79
5.2.3	WiFi Interference to ZigBee under Continuous Spectrum	81
5.2.4	Adapting to the Temporal White Space of WiFi Interference	82
5.2.5	Mathematical Background on Spatial Tessellation and CVT	84
5.3	ART: Co-Existing in Frequency and Time	85
5.3.1	System Model	85
5.3.2	FAVOR: A Location-Aware Frequency Allocation Scheme	86
5.3.3	P-CSMA: Randomizing the Deployment of CSMA	91
5.3.4	Application: Tree-based Data Collection	94
5.4	Evaluation	95
5.4.1	Experiment Settings	96
5.4.2	ART on A Single Link	97
5.4.3	A Five-Link Scenario	98
5.4.4	Tree-Based Data Collection	101
5.4.5	Convergence of FAVOR	104
5.5	Summary	104
6	Conclusion	107
6.1	Research Contributions	107
6.2	Future Directions	108

CONTENTS

References	111
A Proofs for the propositions of LBDP	123

List of Figures

2.1	A node (white) and all its one-hop neighbors (black) are shown. Whether the white one is identified as on the boundary or not depends on the specific geometric interpretation. Precisely, the triangulation on the left indicates the node as on the boundary, but the answer is negative for the case on the right. The shaded ball is used to illustrate the idea of α -shape [1].	11
3.1	A transformation that “blows up” a boundary. The length of a certain arrow in the left figure shows the “force” applied to that part of the boundary. The darker nodes in the right figure are those being blown up to the new convex boundary.	21
3.2	Illustrating the effect of $f(\cdot)$ (defined by the viewpoint v and the “mirror” with radius R) on point clouds. For $\mathcal{N}_i$ of an arbitrary point p_i , the transformation effectively “bends” the locally concave boundary to a convex one in the transform domain (a). Given a point cloud (representing a WSN) $\mathcal{P}$ in (b) and let $\mathcal{P}' = \mathcal{P}$, we show the image $f(\mathcal{P})$ in the transform domain, along with part of the detectable boundary. As demonstrated by the two amplified one-hop neighborhoods, points on non-convex surfaces (of the original 2D area) are also detected as boundary points.	23
3.3	The definition of Depth of Notch (DoN) is shown in 3D (a). However, to simplify the interpretation, we analyze the 2D case in (b), which can be considered as a projection of the 3D case on a certain plane.	25

3.4	The rationale of distinguishing internal boundaries from the external one for 3D WSNs. Though both boundaries should be unit spheres under ϕ , we abuse the size a bit to differentiate them.	33
3.5	To clearly illustrate the “shapes” of the WSNs, We attach their original 3D volumes to respective point clouds (a). For boundary detection with a common viewpoint, boundary nodes are marked on top of the original clouds (b). We remove the internal nodes when local viewpoints are applied to detecting the whole boundary (c).	35
3.6	Boundary detection under location errors. We vary the error radius ε as different fractions of the transmission range r , and we both statistics (a–c) and an example (d–f).	36
3.7	CPU times for MicaZ Mote.	38
3.8	Illustration of the diffusion process of LBS with about 800 nodes and $\pi = 0.1s$	39
3.9	Convergence of LBS under different values of π	40
4.1	k -order Voronoi partition for $k = 1, 2, 3, 4$. The disks at the backdrop represent the (overlapping) sensing ranges of individual sensor nodes. . .	48
4.2	The log/exp maps at point u introduces a mapping between a vector originating at u on $\mathcal{T}_u$ and a geodesic beginning at u on $\mathcal{M}$. The circles on $\mathcal{T}_u$ are mapped to the contours of the geodesic function on $\mathcal{M}$	49
4.3	The dominating region of the central node in k -order Voronoi diagram for k ranging from 1 to 12. The central node needs to collect location (or range) information from its neighboring nodes (the dark nodes) via multi-hop communication according to Algorithm 4 . Additionally, we illustrate multi-hop transmission range using red circles in (a). While the cases for $k = 1$ can be handled by involving only the 6 closest nodes (1-hop neighbors) to the central node, computing the 2-, 3- and 4-order dominating regions requires 2-hop neighbors. When $k > 4$, all sensor nodes within 3 hops are involved.	53

4.4	Recognizing the network boundary (the green dashed curve), a boundary node (dark) determines a searching ring (the outer circle) and check the half-radius arc within the network coverage area (the inner red arc). It then calculates its dominating region (the blue area) with $\mathcal{N}(n_i, \rho)$, while the searching ring helps to determine part of the boundary.	55
4.5	Notations in the proof for Proposition 10.	56
4.6	Initial deployments and k -coverage ($k = 1, 2, 3, 4$) deployments as the output of APOLLO.	62
4.7	The convergence of APOLLO.	63
4.8	Energy consumption during the autonomous deployments for 100 nodes	65
4.9	Total time cost.	66
4.10	Moving loci of a 20-node deployment with APOLLO for 3-coverage. Starting locations are blue and ending locations are red.	67
4.11	Energy consumption in the final deployments of 100 nodes	68
4.12	Adaptability of APOLLO to arbitrarily shaped areas and obstacles as well.	69
4.13	k -coverage deployments on 3D surfaces, $k = 2, 3, 4$	70
4.14	4-coverage ratio.	71
5.1	We overlap the WiFi signal strength observed through inSSIDer in our research center with the 16 channels of ZigBee. Clearly, apart from channels 25 and 26, all other channels are subject to multiple WiFi interferences.	75
5.2	Experiment configuration. The distance from transmitter to receiver is 3.6m.	78
5.3	The throughputs and PRRs for two co-channel links with different distances.	78
5.4	The lower throughput and PRR for two links with different frequency-distance combinations. CSMA is disabled for both links.	79
5.5	The throughputs and PRRs of both links as functions of frequency (a), (b) and distance (c), (d).	80
5.6	The impact of CSMA on throughput and PRR in different frequencies under WiFi interference.	82

5.7	Throughputs and PRRs under different CSMA probabilities.	83
5.8	Allocating (center) frequencies to a 2D WSN based on the locations of nodes. The black points on the 2D plane denotes the nodes, while the “poles” of different heights on the points indicate different frequencies.	87
5.9	Frequency allocation based on Voronoi tessellations. A density function $\Phi(f)$ is defined on the frequency-axis. The warm and cold colors indicate the high and low values of $\Phi(f)$, respectively. Given a regular (ascending) initial allocation (a), our algorithm outputs a much better allocation (b). In both cases, the corresponding Voronoi tessellations are also shown using red line segments as cell boundaries.	91
5.10	An example of P-CSMA. In each transmission window, whether CSMA is enabled for individual data packets is determined by the current CSMA probability. If n_r correctly receives the last data packet in a transmission window (a DPRQ), it will feed the PRR of this window back to n_t . In case that this DPRQ or its corresponding PRR report gets loss, n_t keeps sending inter-window data packets (another type of DPRQ) with CSMA deterministically enabled until the expected PRR report arrives. For every DPRQ from n_t , n_r responds with the PRR of the latest window.	93
5.11	Density function $\Phi(f)$ defined for [2450 – 2458] MHz band.	96
5.12	A MicaZ-based WSN testbed on the ceiling of our research center.	97
5.13	The throughput (a) and PRR (b) of a transmission link at 2455 MHz with different transmission window sizes. The green solid lines indicate the performance of CSMA-based <i>BuzzBuzz</i> , while the red dashed lines represent the performance of deterministically disabling CSMA.	98
5.14	A five-link deployment.	99
5.15	Experiment results for the five-link scenario.	100
5.16	Data collection tree with FAVOR frequency allocation (a) and min-delay transmission schedule (b). The node locations in (a) are plotted roughly proportional to their actual locations in our ceiling testbed.	102
5.17	Performance of six different schemes in a data collection tree.	103
5.18	The convergence of FAVOR in a 30-node WSN.	104

A.1	Illustrations for proofs. (a) Depth of Notch (DoN) in 2D – the extreme case, and (b) p_i is “visible” from at least one of the three vertices of an equilateral triangle enclosing $\mathbf{conv}(\mathcal{N}_i)$	125
A.2	Illustration for proof of Lemma 2.	126

LIST OF FIGURES

List of Tables

4.1	The minimum number of sensor nodes to achieve 2-coverage	68
4.2	The number of sensor nodes to achieve k -coverage with the strategy proposed in [2] for $k = 3, 4, \dots, 8$	70
4.3	The maximum and minimum sensing ranges for three surface deployments	71
5.1	Frequency allocations for the five-link scenario based on different schemes.	99

ABBREVIATIONS

Chapter 1

Introduction

Wireless Sensor Networks (WSNs) have been attracting great interests in both academic and industrial communities, given their pervasive applications in many fields [3, 4], e.g., military, health, environment and commerce, etc.. One of the main trends delivered by the ubiquitousness of WSNs is the increasing network scale [5, 6]: hundreds, even thousands of sensor nodes are deployed and networked in a targeted area to survey or monitor events of interest. It is evident that centralized algorithms are not adapted to the large-scale WSNs due to their heavy cost in terms of energy and time for performing global coordination within the networks. In addition, centralized computations within networks usually lead to serious load unbalance thereby threatening system lifetime and robustness. Therefore, we aim at designing localized algorithm for large-scale WSNs in this thesis, which can be performed by individual sensor nodes in a distributed manner using only locally available information.

In this chapter, We first introduce the backgrounds and motivations behind this thesis to drive our research work in Sec. 1.1, before presenting the innovation of this thesis in designing localized algorithms in Sec. 1.2. We then summarize our contributions and outline this thesis in Sec. 1.3 and 1.4 respectively.

1.1 Background

The boundary of a WSN is one of its important geometry properties, based on which network users can derive network sensing area, even the event boundaries when the targeted area is fully covered by the WSN. The boundary detection also plays a fundamen-

tal role in many other network functionalities (e.g., geographic routing and network deployment that we will discuss later). Since many up-to-date applications require WSNs to monitor time-varying events (e.g., bio-geo-chemical processes, streams/currents, and pollution in atmosphere or waterbodies), or even WSNs themselves are movable (e.g., mobile sensor networks), the boundary detection should be performed in an on-line manner. However, such an on-line boundary detection cannot be realized through centralized algorithms due to its conspicuous cost especially for WSNs of large scale. On the other hand, in many application scenarios, sensor nodes may be deployed in a 3D volume (e.g., water bodies), instead of a 2D plane. The complexity induced by 3D geometry also increases the difficulties of detecting 2-manifold network boundaries. Furthermore, some network functionalities, e.g., geographic routing, may fail in face of the arbitrarily shaped boundaries of 3D WSNs.

As mentioned above, boundaries of a WSN can profile its coverage area, but only boundary information is not sufficient to design a sensor deployment strategy to serve certain coverage goals. In fact, sensor deployment is a fundamental problem for WSNs. Considering a large-scale WSN may be deployed in an inaccessible area (e.g., a remote disaster field), mobile sensor nodes are developed [7], such that the initial random deployments can be improved by autonomously adjusting the locations of the nodes [8, 9]. Additionally, another advantage of such an autonomous deployment is the adaptability to the dynamics of targeted events. However, the existing autonomous deployment strategies only deal with 1-coverage, which are prone to suffering from the vulnerability of the sensor nodes. Although multiple-coverage has the advantage of high tolerance to node failures, existing proposals are confined by their centralized execution. How to realize multiple-coverage in an autonomous manner with certain objective optimized (e.g., maximizing the network lifetime through load balancing) is still an open problem. Furthermore, when a WSN is deployed on a 3D terrain, extending typical 2D area coverage to 3D surface coverage is a highly challenging issue as well and has not been studied in depth.

Under a well designed network deployment (especially a multiple-coverage deployment), the sensing potential of a WSN is highly utilized, such that a large amount of data may be collected by sensor nodes and get ready to be fed back to network users through a sink node for further analysis and process. Moreover, the increasing scale of

sensor node deployments also poses higher demands in network throughput and transmission quality. For example, when the data sensed by a WSN are carried to a sink through a tree-like structure, a high data rate is desired especially for the upstream sensor nodes, while the data loss rate should be as low as possible to ensure data usability. Therefore, the multi-channel capability of ZigBee radios (e.g., CC2420 of MicaZ Mote [10]) is intensively exploited (e.g., [11, 12]) to enable concurrent transmissions by utilizing orthogonal ZigBee channels [13]. However, multi-channel WSNs still face the contentions from other networks that also operate in the 2.4 GHz ISM band. In particular, the pervasive deployments of WiFi hotspots (used to provide ubiquitous WiFi service to publics) introduce heavy interference which severely restricts the availability of the limited orthogonal channels. On the other hand, although many solutions have been proposed to avoid WiFi interference in either time [14], or packet structure [15], or frequency [16], their fundamental component, *Carrier Sense Multiple Access* (CSMA) prominently sacrifices transmission throughput for quality.

1.2 Scope of Research

In this thesis, we try to understand WSNs mainly from the perspective of geometry. In particular, we build geometric models for WSNs, and apply localized algorithms to address challenging problems in WSNs (e.g., the ones proposed above). The locations and connectivity of the sensor nodes, their embedded space, even their channel allocations can be utilized to perform geometric modeling. The conventional geometric algorithms usually entail global information as input (e.g., the sensor nodes' locations in a global coordination system), whereas we hereby develop localized geometric algorithms that can be carried out by individual sensor nodes in a distributed fashion using locally available information.

Our first focus is on on-line boundary detection and regulation for 3D WSNs. We build a *point cloud* model to characterize the WSN deployed in a 3D volume, where the sensor nodes can be regarded as the “points” sampled from the 3D volume. Hence, the boundary detection problem is reduced to recognizing the boundary surface of the volume. One geometric property of the volume's boundary surfaces is “direct visibility”: they can be seen from certain (local) viewpoints, whereas the interior of the volume cannot be observed. Exploiting such local visibility information, we accordingly

design an on-line algorithm to identify the sensor nodes on the boundary surfaces. We also investigate how to apply localized parametrization technique to regularize the arbitrarily shaped boundaries of the 3D WSNs, which can be utilized to support many other network functionalities.

The on-line boundary detection also provides fundamental infrastructure for our following work on autonomous deployment. In this thesis, we aim at designing an autonomous deployment algorithm for load-balancing k -coverage. From geometric viewpoint, the deployment problem can be formulated as a *space tessellation* problem. For example, in 2D coverage, the sensing range of a sensor node is generally modeled as a disk in geometry; therefore, the deployment problem is reduced to “tessellating” the disks in the 2D targeted area such that each point in the targeted area can be covered by at least k disks. On the other hand, to prolong the network lifetime, the load distribution within the network should be balanced through minimizing the required maximum sensing disk. Furthermore, as the sensor nodes may be deployed on 3D surfaces rather than 2D planes in some applications (e.g., volcano monitoring), generalizing 2D plane coverage to 3D surface coverage under the uniform framework is another concern.

The concept of space tessellation also has the potential of being applied to (partially) solve our third topic in this thesis. To guarantee both transmission throughput and quality for large-scale WSNs in face of WiFi interference, we attempt to raising the utilization of the frequency and time resources. Since the existing multi-channel techniques (e.g., [16]) are usually confined by a limited number of discrete ZigBee channels, we hereby attempt to allocate continuous center frequencies (thus channels) to nodes/links; thereby better exploiting the spectrum less interfered by WiFi to support transmission parallelism. Geometrically speaking, such frequency allocation can be resolved by optimally “tessellating” the sensor nodes in a continuous frequency space. On the other hand, considering the most popular mechanism to deal with interference, CSMA, significantly sacrifices transmission throughput for transmission quality, we expect a more flexible way to adapt the utilization of CSMA to dynamic WiFi interference.

1.3 Contributions

In this thesis, we have made the following contributions

- We propose *Localized Boundary Detection and Parametrization* (LBDP) consisting of two components, *UNiform Fast On-Line boundary Detection* (UNFOLD) and *Localized Boundary Sphericalization* (LBS).
  - Borrowing the idea of *direct visibility*, the first component, UNFOLD, applies an inversion to local node coordinates such that a “notched” surface is locally “unfolded” into a convex one, such that the locally visible boundary surfaces (thus the sensor nodes) can be recognized by a localized convexity test.
  - LBS employs parameterization technique to maps an arbitrarily shaped boundary into a unit sphere in a localized fashion, which in turn supports functionalities such as distinguishing inter boundaries from external one and geographic routing.
- We propose a deployment strategy, *Autonomous dePlOyment for Load baLancing k-surface cOverage* (APOLLO), for mobile sensor networks equipped with tunable sensing ranges.
  - In light of *k-order Voronoi tessellation*, we propose an autonomous deployment strategy such that the mobile sensor nodes can be deployed for *k*-coverage, while the maximum sensing range among them is minimized for load balancing purpose.
  - We also generalize our preliminary 2D *k*-area coverage to 3D *k*-surface coverage under the same framework by replacing distance metric.
  - We prove the termination and the (local) optimality of APOLLO. We also demonstrate the close relationship between our optimization objective and other frequently considered objectives.
- we propose *Adaptive fRequency-Temporal co-existing* (ART) as a framework to improve the co-existence of ZigBee and WiFi in terms of both transmission throughput and quality.
  - Inspired by *Centroidal Voronoi Tessellation* (or CVT), we first propose *Frequency Allocation for Versatile Occupancy of spectRum* (FAVOR) to optimally allocate continuous (center) frequencies; thereby supporting more parallel transmissions.

- We propose *Probabilistic CSMA* (P-CSMA) to opportunistically tune the use of CSMA to leverage transmission throughput and quality, based on a real-time assessment of transmission quality.
- We implement ART in MicaZ platforms, and demonstrate the efficacy of ART in enhancing both throughput and transmission quality through extensive experiments under different scenarios.

1.4 Thesis Organization

This thesis is organized as follows:

In Chapter 2, we survey the literature related to the topics covered by this thesis, including boundary detection, sensor coverage as well as ZigBee-WiFi co-existence. We exploit the motivations to drive our research through carefully reviewing a few representative papers. We also give a brief discussion upon the differences between our research threads and the existing works to highlight our innovations.

In Chapter 3, we present our first work on boundary detection and regulation for 3D WSNs. We present a new understanding of boundary detection problem from geometry viewpoint. Using the proposed on-line boundary detection algorithm, each sensor node can real-timely be aware of whether it is on the network boundaries relying on only local computations. We also report a localized parameterization algorithm to regulate the detected network boundary surface, thereby supporting other network functionalities.

In Chapter 4, we propose an autonomous deployment framework for load-balancing k -coverage. In our algorithm, each sensor node can individually determine its sensing range and location in a localized fashion, such that the targeted area is k -covered while the maximum sensing range among the sensor nodes is minimized. We also generalize our proposed deployment strategy to the WSNs deployed on 3D surfaces.

In Chapter 5, we jointly apply multi-channel technique and CSMA control strategy to ZigBee-based WSN platforms, in order to guarantee both transmission throughput and quality in face of WiFi interference. We unify location space and frequency space into one space, in which the sensor nodes are “tessellated”, to compute (locally) optimal continuous frequency allocation. Moreover, we develop a randomized algorithm to improve CSMA mechanism, thereby leveraging transmission throughput and quality.

In Chapter 6, we conclude the works we have done. We also outline some promising direction extending our current research thread.

Chapter 2

Literature Review

In this chapter, we first survey the related proposals concerning boundary detection in WSNs. We hereby do not survey the boundary regulation through distributed parametrization since it has never been done in literature, to the best of our knowledge. We then introduce the existing works about area/surface coverage of WSNs, including static random/deterministic coverage and mobile assisted coverage. We finally dive into reviewing the related literature about co-existence of ZigBee and WiFi in the aspects of network-based channel allocations and link-based medium access mechanisms.

2.1 Boundary Detection for Wireless Sensor Networks

In this section, we briefly discuss the existing proposals for boundary detection, which in turn serve as the motivations for our on-line approach. Note that, in computational geometry community, surface reconstruction from sample points has been extensively studied. Representative proposals include CRUST [17] and Cocones [18], which are based on Voronoi diagrams generated by the sample points. These approaches assume that the input points are densely sampled from some surface, so they are not suitable for boundary detection, whose objective is to identify boundary points from a set of points that form a 3D volume.

2.1.1 Geometric or Topological

The existing boundary detection approaches can be roughly classified into two categories, namely geometric (e.g., [19, 20, 21, 22]) and topological (e.g., [23, 24, 25, 26,

27, 28, 29]). While the former always requires the knowledge of nodes location or distance, the latter is often claimed as a location/range-free approach. For example, Ghrist and Muhammad [24] compute homology groups, algebraic topological invariants, to recognize “holes” within WSN coverage. Kröller *et al.* [26] define boundary based on chordless cycles and propose a series of fairly sophisticated algorithms to identify subgraphs that satisfy the boundary criterions. [23, 28] and [29] are similar in the sense that they rely on global connectivity information (e.g., shortest path tree or primary boundary circle) to “guide” further boundary refinements. In [30] and [31], boundary nodes are detected by coarse-to-fine approaches based on connectivity information. Landmark boundary nodes are first selected to delineate a 3D WSN, and then the resulting coarse boundaries are refined to accurate ones.

Generally, the price a topological approach pays to avoid relying on location/range information is a highly complicated procedure that often requires a large scale coordination among a WSN. For example, the algorithm proposed in [24] is actually centralized, while the ones in [30] and [31] requires communication flooding within the network.

Therefore, while the topological approaches do offer a one-time boundary detection for static WSNs, they are not adequate to on-line detection. Moreover, whether these topological approaches can be extended to distributed 3D boundary detection is still open. We thus focus on studying geometric boundary in this thesis.

2.1.2 The “Pain” of Geometric Approaches

Equipped with location or range information (precisely, each node knows either its own location that may come with a certain error or the distances between itself and close-by nodes), geometric approaches often involve fairly localized computations and hence have the potential to be performed on-line. However, the discrete nature of a WSN “volume” may lead to complicated local detections. On one hand, there is no commonly agreed definition for the boundary of a point cloud (the geometric representation of a WSN). For example, Fig. 2.1 shows that whether a node is at the boundary or not heavily depends on the specific geometric interpretation. On the other hand, the algorithms for boundary detection, albeit localized, can still incur a high time and/or message complexity.

Zhang *et al.* [20] propose to use two local geometric structures, namely *localized Voronoi polygon* and *neighbor embracing polygon*, for boundary characterization only

Figure 2.1: A node (white) and all its one-hop neighbors (black) are shown. Whether the white one is identified as on the boundary or not depends on the specific geometric interpretation. Precisely, the triangulation on the left indicates the node as on the boundary, but the answer is negative for the case on the right. The shaded ball is used to illustrate the idea of α -shape [1].

in 2D. It is shown in the paper that the detection procedure involves several rounds of interactions between (at least) all one-hop neighbors. Both [21] and [22] use a concept called α -shape [1] for boundary detection. In a nutshell, α -shape results from “erasing” the convex hull of a point cloud using a spherical “eraser” with a certain radius α : while 0-shape is the original point cloud, ∞ -shape is the convex hull. In Fig. 2.1, the left triangulation is actually an α -shape with α (roughly) equal to half of the transmission range. Although the α -shape construction leads to localized boundary detection, its computation cost is still non-negligible.

2.1.3 Event Boundary vs. Network Boundary

Some existing proposals tend to distinguish between event boundary and network boundary [22]. In this thesis, we are concerned with both boundaries for the WSN applications under consideration. In fact, network boundary can be considered as a particular event boundary, with the “event” being the WSN itself. It is true that, whereas the network boundary is a clear-cut concept, other event boundaries can be rather fuzzy, due to sensing errors or smooth changes (in terms of certain physical quantities indicating the event) around the boundary. Fortunately, relying on statistic approaches such as [32], each node can arrive at a (binary) indicator on whether it covers a certain event or not [33]. Moreover, using an event indicator may suggest a simple way to detect event boundaries: a boundary node has at least one of its neigh-

bors carrying an indicator value different from its own value. However, this method often leads to rather “thick” boundaries if the transmission range of nodes is relatively large. Therefore, we do not make a distinction between these two types of boundaries in this thesis.

2.2 Area/Surface Coverage of WSNs

Given the vast literature for WSN coverage, we focus only on the topics related to area/surface coverage and mobile assisted autonomous deployments, but leave out other interesting topics, such as point (or target) coverage (e.g., [34, 35, 36]).

2.2.1 Area Coverage with Random Deployments

In some applications, unmovable sensor nodes may be randomly deployed (e.g., in an air-drop manner) in an inaccessible area. One of the main concerns of the existing proposals is the limited energy supply of sensor nodes. One thread is to determine the minimum number of sensor nodes given their duty-cycle settings, e.g., [37] supposing the sensor nodes have independent low duty-cycle, while another more popular one is to schedule the duties of the redundantly deployed sensor nodes to prolong network lifetime. For example, a set of active sensor nodes can be selected (in a centralized manner) such that certain coverage and connectivity requirements are guaranteed with their total energy cost (in sensing and communicating) minimized [38, 39, 40]. Moreover, a polynomial-time distributed scheduling algorithm is proposed in [41], to dynamically activate the sensor node to maximize the network lifetime, according to the residual energy storage of the sensor nodes. Inspired by the above proposals, maximizing the network lifetime is also one of the main objective of our mobile assisted deployment strategy but in a quite different way.

2.2.2 Area Coverage with Deterministic Deployments

The static and deterministic area coverage problem is essentially a geometry problem; the results for 1-coverage with a minimum number of nodes can be directly taken from pure mathematical research [42]. In later research proposals for WSN coverage, the focus is more on minimum node 1-coverage with certain connectivity requirement (e.g., [43, 44, 45]). While it is known that a 1-covered WSN is also connected if the

transmission range R_t and the sensing range R_s satisfy $R_t \geq \sqrt{3}R_s$, a strip-based deployment strategy is proposed in [43] for other values of R_t , which is proven to be asymptotically optimal in [44]. In fact, adding more strips may allow a WSN to achieve higher degree of connectivity (or k -connectivity).

Compared with k -connectivity, the progress on k -coverage appears to be relatively slower. To the best of our knowledge, the only optimality result in terms of minimum node k -coverage is presented in [45], where $k = 2$. It appears that minimum node k -coverage (for $k > 2$) is better to be tackled indirectly due to its hardness. As we will show in Chapter 4, our objective of a k -coverage with mini-max sensing range may also imply minimum node k -coverage.

2.2.3 Mobile Assisted Area/Surface Coverage

The utilization of mobile sensor nodes poses increasing challenge in designing motion control strategy. A few 3-coverage heuristics that aim at bounding the minimum separation among sensor nodes is proposed in [46] to guide the movements of the sensor nodes; the paper also shows that bounding the min-separation may lead to lower coverage redundancy and is hence a good approximation to minimum node 3-coverage. Deploying WSNs for k -coverage using mobile nodes is also reported in [2, 47], but their approaches are not autonomous (hence the algorithms are not localized) as they all rely on a “blueprint” to guide the node mobility and to guarantee the coverage.

Considering that a WSN can be deployed on a 3D terrain for some application scenario, the surface coverage problem has also been investigated. Zhao *et al.* [48] shows that extending just 1-area coverage to 1-surface coverage already poses great challenges, and they provide approximation algorithms to the resulting NP-complete problem. Recently, Jin *et al.* [49] applies Centroid Voronoi Tessellation (CVT) [50] and discrete Ricci flow [51] to obtain some form of optimal 1-coverage on 3D surfaces.¹ However, the discrete Ricci flow is a global parametrization process, so the motion control cannot be implemented in a distributed manner using only local information.

Our work is also related to the sensing heterogeneity issue [9, 52]. Unlike the previous proposals that aim to cope with sensing heterogeneity or evaluate its impact,

---

¹In [49], the sensing range is assumed to be a continuous function (representing sensing accuracy attenuation with an increasing geodesic distance), hence the resulting coverage may only serve as an approximation to the combinatorial 1-coverage problem.

we actively exploit the sensing heterogeneity to construct our algorithm that guides the autonomous deployment.

2.3 Improving Co-existence of ZigBee and WiFi

Our solution to co-existence of ZigBee and WiFi consists of a network-based channel allocation and a link-based WiFi interference-resisted protocol. Therefore, we hereby survey the existing proposals concerning the two topics respectively.

2.3.1 Channel Allocations in WSNs

Recent years have witnessed the development of multi-channel techniques to improve the performance of ZigBee-based WSNs (especially in terms of network throughput) [11, 53]. Most proposals make use of graph-coloring heuristics to allocate channels in the whole network, but the work of [11] innovated in allocating channels to disjoint trees in WSNs. As the allocated channels are assumed to be orthogonal, the inter-channel interference is often neglected. However, the inter-channel interference does exist and its impact on link capacity and the performance of multi-channel protocols are systematically explored in [11, 54]. On the other hand, the existence of WiFi interference exerts more limitation to the availability of those orthogonal channels.

To further improve the spectrum utilization in WSNs, partially overlapped channels have recently been introduced [55]. However, the proposal in [55] is still too conservative in using partially overlap channels, and it fails to discover the continuous frequency-distance relation in a spatial-frequency space. Moreover, all the aforementioned multi-channel schemes still apply CSMA to handle possible transmission conflict, but as we have shown in Sec. 5.2.1, testing the performance of a multi-channel scheme on CSMA-enabled platforms may lead to wrong conclusions. As demonstrated by our FAVOR, CSMA may not be needed anymore for handling ZigBee interference; disabling CSMA has the potential to considerably improve the network throughput.

The multi-channel feature of ZigBee radios has also been applied to improve the quality of individual links [56] and to avoid the interference from WiFi devices [16]. However, the scalability of these proposals are still confined by the limited channels. We believe that the flexible spectrum utilization offered by FAVOR can also contribute

to tackling these problems, as well as problems under a joint routing and link scheduling framework (e.g., [57]) or for duty-cycled WSNs (e.g., [58]).

2.3.2 Survive in WiFi Interference for WSNs

Unfortunately, fully utilizing the whole ZigBee spectrum may cause certain ZigBee links to operate in the WiFi-interfering spectrum (which could be avoided should a single ZigBee channel, e.g., channel 25 or 26, be used). This co-existing issue has provoked certain interests from the research community [14, 15, 59, 60]. The proposals try to handle WiFi interference from every different perspectives. [59] proposes to model the correlation of interference and link burstiness in an off-line manner. Being aware of this model, ZigBee nodes could estimate the duration of the interference and then schedule the on-going transmissions by intelligently setting the backoff delay. [14] observes that WiFi frames are usually highly clustered in time, which results in a large number of temporal white spaces. Exploiting the self-similarity of the arrival process of WiFi frame clusters to predict the lengths of the white spaces, the data frames from MAC layer can be divided into sub-frames of smaller size and then scheduled into those “niches”. [15] presents Multiple-Header (MH) mechanism (where multiple headers are attached to each packet), based on the observation that the header has the highest probability to be corrupted in a packet. [15] also employs Reed-Solomon coding to help correcting bit errors in packets. These two FEC mechanisms are deployed to complement the existing ARQ component in ZigBee, adding on a considerable overhead. Finally, [60] jointly takes both interference intensity and density into account, and designs a new method for assessing WiFi interference in a certain channel through periodically sampling the RSSI reading. By assessing the WiFi interference in all available channels, the nodes are enabled to pick up the least interfered channel for receiving packets.

All the above existing proposals only aim at enhancing the transmission quality but overlook the need for high throughput. To pursuing a perfect PRR, CSMA is unanimously adopted as a foundation in these approaches for handling both internal ZigBee and external WiFi interference. However, as we have showed, CSMA heavily trades throughput for higher PRR. Moreover, the adaptations proposed by existing proposals have introduced extra overhead, which may further jeopardize network throughput. In this thesis, we innovate in efficiently utilize the frequency spectrum and in intelligently

deploying CSMA, which in turn allows it to achieve both excellent throughput and high transmission quality.

Chapter 3

LBDP: Localized Boundary Detection and Parametrization for 3D Sensor Networks

Many applications of wireless sensor networks involve monitoring a time-variant event (e.g., radiation pollution in the air). In such applications, fast boundary detection is a crucial function, as it allows us to track the event variation in a timely fashion. However, the problem becomes very challenging for a large scale sensor node deployment, as it demands a highly efficient algorithm to cope with the dynamics introduced by the evolving event. Moreover, as many physical events occupy volumes rather than surfaces (e.g., pollution again), the algorithm has to work for 3D cases. Finally, as boundaries of a 3D network can be complicated 2-manifolds, many network functionalities (e.g., routing) may fail in the face of such boundaries. To this end, we propose *Localized Boundary Detection and Parametrization* (LBDP) to tackle these challenges. The first component of LBDP is *UNiform Fast On-Line boundary Detection* (UNFOLD). It applies an inversion to node coordinates such that a “notched” surface is “unfolded” into a convex one, which in turn reduces boundary detection to a localized convexity test. We prove the correctness and efficiency of UNFOLD; we also use simulations and implementations to evaluate its performance, which demonstrates that UNFOLD is two orders of magnitude more time and energy efficient than the most up-to-date proposal. Another component of LBDP is *Localized Boundary Sphericalization* (LBS). Through purely localized operations, LBS maps an arbitrary boundary to a unit sphere, which

in turn supports functionalities such as distinguishing inter boundaries from external one and distributed coordinations on a boundary. We implement LBS in TOSSIM and using simulations to show its effectiveness.

3.1 Introduction

One of the main applications of WSNs is to constantly monitor physical events (or phenomena) either too widely spread or too remote to be accessed through conventional techniques. Boundary detection, as an enabling technique to such applications, becomes very crucial to the functionality of WSNs. It allows the network users to be aware of the geometry of the network, which infers either the sensing coverage (if the network only partially covers the targeted event [36]) or the boundary of the targeted event (if the network fully covers it [61]). Moreover, boundary detection is also an important component to support geographic data routing [23, 62, 63] as well as sensor node deployments (which we will present in Chapter 4) for mobile WSNs.

While most of the existing boundary detection approaches are designed just for a “one-time shot”, the detection actually has to be constantly conducted, given the time-varying nature of the events under surveillance. Such events, for example, can be bio-geo-chemical processes, streams/currents, or pollution in atmosphere or waterbodies (e.g., ocean). Depending on different deployments, a WSN can be either static (e.g., in smart buildings) to observe the event passing through or stuck to an event to keep monitoring it (e.g., for water monitoring). In both cases, boundary detection has to be performed on-line to keep tracking either the event or the network boundary. Unfortunately, the existing approaches have too high message or time complexity to be performed in an on-line manner.

Another feature of the events under consideration is that they often span a 3D volume rather than a 2D surface. Given the fact that very few existing proposals deal with 3D boundary detection and that extending the approaches designed for 2D surfaces to 3D volumes is highly nontrivial in geometry,¹ a clean-slate boundary detection algorithm needs to be designed for 3D WSNs. Note that, should a 2D boundary detection be ever needed, it would be really trivial to reduce a 3D detection approach to 2D.

---

¹For example, the well known *edge flip* algorithm for Delaunay triangulation in 2D does not converge in 3D [64].

Furthermore, 3D WSNs may have arbitrarily complicated but orientable 2-manifolds as their boundaries. This, on one hand, makes it difficult to tell internal boundaries from the external one from a local point of view. On the other hand, routing protocols may fail when facing or on such boundaries [65]. Therefore, if we do not regularize the boundary geometry, the boundary detection may be performed in vain without delivering necessary supports to networking functionalities.

In this chapter, we tackle the aforementioned challenges by proposing *Localized Boundary Detection and Parametrization* (LBDP). LBDP comes with two components: *UNiform Fast On-Line boundary Detection* (UNFOLD) for boundary detection and *Localized Boundary Sphericalization* (LBS) for boundary regularization. The underlying principle of UNFOLD is to apply a spherical inversion to the local coordinates of every node, such that a (locally) concave surface can be “unfolded” into a convex one. As a result, the painful procedure of identifying a boundary node on a “notched” surface is reduced to convexity test (which can be tackled with simple geometric tools). Though the idea of inversion is borrowed from [66], UNFOLD has substantially improved on it by implementing it in a distributed networking scenario. As UNFOLD entails only simple and uniform computation for every node, it can be performed super fast and hence enable on-line boundary detection. Building upon UNFOLD, LBS employs a diffusion process to regularize an arbitrary boundary into a unit sphere. The algorithm is purely localized and involves only arithmetic operations, and the resulting virtual coordinates on the sphere may be exploited by many networking functionalities, such as distinguishing internal boundaries from the external one. Our main contributions in LBDP are:

- The idea of 3D WSN boundary detection in a transformed domain.
- The on-line algorithm, UNFOLD, that entails only localized communications and computations.
- A real implementation of UNFOLD in MICAz Motes for time and energy efficiency evaluations.
- The parametrization algorithm, LBS, that efficiently regularizes an arbitrary boundary into a unit sphere.

- Distinguishing internal boundaries from the external one (from a local point of view) based on the virtual coordinates delivered by LBS.
- A full implementation of LBS in TOSSIM for realistic simulations.

In the following, we focus on UNFOLD in Sec. 3.2: we first discuss boundary definitions and properties, then we present UNFOLD in detail along with the corresponding analysis. We then present LBS in Sec. 3.3, as well as the descriptions of two crucial applications of LBS. We finally report the simulation and experiment results in Sec. 3.4, before concluding this chapter in Sec. 3.5. To maintain fluency, we postpone the proof details or sketches to Appendix A.

3.2 UNFOLD: Detecting Boundary in a “Mirror” Image

In this section, we first introduce our network model and give a brief overview of UNFOLD. Then we describe the transformation used by UNFOLD to transform a point cloud (geometrically representing a WSN), as well as our formal definitions of the boundaries for the point cloud. Finally, we present in detail the localized on-line algorithm for boundary detection, along with its performance analysis.

3.2.1 UNFOLD in a Nutshell

We model a 3D WSN as a *point cloud* $\mathcal{P} = \{p_i | 1 \leq i \leq n\} \subset \mathbb{R}^3$, with each point p_i representing a sensor node. We are not concerned with network topology, so the boundaries that we aim at detecting are purely geometric and only concern local network connectivity. We assume that each node has a convex *transmission volume*, such that a bi-directional communication link exists between this node and any other node within this volume. We denote by $\mathcal{N}_i$ the set of nodes within the transmission volume of node p_i , or p_i 's *one-hop neighbor set*. We also assume that node p_i is either aware of its geographic location or can measure the distance between itself and another node in $\mathcal{N}_i$. In this chapter, we follow the definition of orienting a boundary surface in [67] (p611-615), and draw a distinction between external and internal boundaries. Intuitively, the (unique) external boundary encloses the whole WSN, whereas an internal boundary encloses a “hole” inside the WSN (so it is also termed *hole boundary* [28]).

Similar to the existing work [22, 28, 62], we assume that these boundaries are all of genus-0.

As explained in Sec. 2.1.2, the difficulty of determining whether a node is on the boundary stems from the existence of notches and, more importantly, from the **absence of convex boundary** due to consecutive notches; one would need to rely on the fairly complicated α -shape construction to identify boundary nodes. Our UNFOLD applies a special transformation to “blow up” a boundary such that it becomes almost convex, and we define a boundary node based on its local convexity. We illustrate the idea by Fig. 3.1.

Figure 3.1: A transformation that “blows up” a boundary. The length of a certain arrow in the left figure shows the “force” applied to that part of the boundary. The darker nodes in the right figure are those being blown up to the new convex boundary.

Given such a transformation, the localized algorithm for UNFOLD to perform boundary detection becomes straightforward. Each node periodically exchanges location or range information with its one-hop neighbors to construct a local coordinate system. Then the transformation is applied to the coordinates of all nodes in $\mathcal{N}_i \cup \{p_i\}$, possibly from different directions. In the transformed domain, node p_i performs convexity test to check if it is on or out of the convex hull of $\mathcal{N}_i$, and it indicates itself as a boundary node if this test succeeds. Relevant questions we need to address are:

- Q1 What are the properties of the boundaries resulting from UNFOLD?
- Q2 How many transformations need to be applied?
- Q3 What if the location/range information for individual nodes comes with errors?

In the following, we present detailed principles and algorithms for UNFOLD, while addressing these questions.

3.2.2 Transformation and Boundary Definition

As we mentioned in Sec. 2.1.2, there is no commonly agreed definition for the boundary of a point cloud. Our definition is based on the assumption that a point cloud results from sampling a certain hypothetical 3D volume. Therefore, a point is on the **boundary** of the point cloud if it lies on the hypothetical **surface** of that volume. As such a surface is unknown, we borrow the idea of *direct visibility* from [66]: a surface (hence the points lying on it) is what we can see from a certain viewpoint. We first present a transformation that enables the recognition of points on such a surface, then we define boundary and its properties.

3.2.2.1 Definitions

Given any subset $\mathcal{P}' \subseteq \mathcal{P}$, we associate with $\mathcal{P}'$ a local coordinate system, and we place a *viewpoint* v that does not belong to the convex hull of $\mathcal{P}'$ at the origin and set a *spherical “mirror”* centered at v with radius R (where $R > \|p_i\|, \forall p_i \in \mathcal{P}'$). The transformation for a point $p_i \in \mathcal{P}'$ is given by:

$$\check{p}_i = f(p_i) = p_i + 2(R - \|p_i\|) \frac{p_i}{\|p_i\|} \quad (3.1)$$

where $\|\cdot\|$ can be any norm but we hereby take Euclidean norm $\|\cdot\|_2$. We illustrate the effect of this “spherical reflection” in Fig. 3.2.

Our definition of point cloud boundary is based on the transformation $f(\cdot)$.

Definition 1. For any point $p_i \in \mathcal{P}' \subseteq \mathcal{P}$, p_i is said to be **on the boundary** of $\mathcal{P}'$ with respect to a **common** transformation $f(\cdot)$ (defined by v and R) if $\mathcal{N}_i \subseteq \mathcal{P}'$ and $\check{p}_i$ lies on the convex hull of $f(\mathcal{N}_i) \cup \{v\}$.

We refer to Fig. 3.2 for the two extreme cases of $\mathcal{P}' = \mathcal{N}_i$ and $\mathcal{P}' = \mathcal{P}$. Note that, as the definition concerns the image of $\mathcal{N}_i$ in the transform domain, it focuses only on the local property of a point cloud. Actually, we have another definition of boundary for which even the transformation is made local:

Definition 2. For any point $p_i \in \mathcal{P}' \subseteq \mathcal{P}$, p_i is said to be **on the boundary** of $\mathcal{P}'$ with respect to a **particular** transformation $f_{p_i}(\cdot)$ (defined by v_{p_i} and R_{p_i} that depend on p_i) if $\mathcal{N}_i \subseteq \mathcal{P}'$ and $\check{p}_i$ lies on the convex hull of $f_{p_i}(\mathcal{N}_i) \cup \{v_{p_i}\}$.

Figure 3.2: Illustrating the effect of $f(\cdot)$ (defined by the viewpoint v and the “mirror” with radius R) on point clouds. For $\mathcal{N}_i$ of an arbitrary point p_i , the transformation effectively “bends” the locally concave boundary to a convex one in the transform domain (a). Given a point cloud (representing a WSN) $\mathcal{P}$ in (b) and let $\mathcal{P}' = \mathcal{P}$, we show the image $f(\mathcal{P})$ in the transform domain, along with part of the detectable boundary. As demonstrated by the two amplified one-hop neighborhoods, points on non-convex surfaces (of the original 2D area) are also detected as boundary points.

While the latter definition allows the transformation to be adapted to local geometry (hence requires only local range information), the former definition (which requires location information) may have certain practical significance.¹ In general, these two definitions, on one hand, are both amenable to the design of localized algorithm. On the other hand, the local properties stated in both definitions do have a global implication to some extent, according to the following result.

Proposition 1. *If p_i is a boundary point of $\mathcal{P}$ according to either definition, there exists a **nontrivial** $\mathcal{P}'$, i.e., $\mathcal{N}_i \subset \mathcal{P}' \subseteq \mathcal{P}$, and a transformation $f'(\cdot)$, such that $f'(p_i)$ lies on the convex hull of $f'(\mathcal{P}') \cup \{v'\}$. In particular, p_i is an **external** boundary point of $\mathcal{P}$, iff $f'(p_i)$ lies on the convex hull of $f'(\mathcal{P}) \cup \{v'\}$.*

¹Taking the recent BP oil spill as an example. Should a WSN be deployed to monitor the spill coverage, a question some Miami tourism authority might ask would be: how far is the spill frontline towards Key West? As the concerned boundary is only the section facing Key West, it makes more sense to perform a boundary detection based on a common viewpoint at Key West.

The transformation $f(\cdot)$ shown in Equation (3.1) is inspired by an inversion applied in a quite different context [66]. Katz *et al.* [66] applies this inversion to identify *visible* points on part of the external boundary of a point cloud, our extension involving only local convexity test, however, allows boundary detection for both internal and external boundaries of a point cloud from all directions.

3.2.2.2 Properties

First, it is straightforward to show that the boundary definitions preserve convexity, a safety property.

Proposition 2. *If p_i is on the convex hull of $\mathcal{P}' \subseteq \mathcal{P}$ and $\mathcal{N}_i \subseteq \mathcal{P}'$, then p_i is on the boundary of $\mathcal{P}'$ with respect to a certain transformation $f(\cdot)$.*

In other words, if a point is on a (internal or external) surface of the hypothetical volume represented by $\mathcal{P}$ and is definitely “visible” (due to locally convexity) from some viewpoint, it will be recognized as a boundary node.

Second, our definition assures that a node that is hidden behind a boundary node from the viewpoint v will not be recognized as a boundary node, i.e.,

Proposition 3. *If p_i is recognized as a boundary node for $\mathcal{P}' \subseteq \mathcal{P}$ with respect to a certain transformation $f(\cdot)$, $p'_i \in \mathcal{N}_i$ will not be recognized as a boundary node if v, p_i, p'_i are collinear.*

This is another safety property.

The third property quantifies the impact of the transformation on a notched surface to “blow it up”. We first define quantities to measure to what extent a point is **notched** locally.

Definition 3. *Given $p_i \in \mathcal{P}$ and $\mathcal{N}_i$, if p_i lies within the convex hull, $\text{conv}(\mathcal{N}_i)$, of $\mathcal{N}_i$, we define the **Depth of Notch (DoN)** d of p_i as the distance from p_i to the nearest plane on $\text{conv}(\mathcal{N}_i)$, and the **inner radius** ρ of $\mathcal{N}_i$ as the distance from p_i to the nearest vertex of $\text{conv}(\mathcal{N}_i)$.*

We illustrate the definition of DoN in Fig. 3.3(a), and we show that, given $f(\cdot)$ (with certain v and R), a node can be “blown” to the convex hull only if its DoN is below a certain threshold. The proposition, illustrated by Fig. 3.3(b), is stated in 2D for brevity, but it can be readily extended to 3D: where the 2D conditions need to be

Figure 3.3: The definition of Depth of Notch (DoN) is shown in 3D (a). However, to simplify the interpretation, we analyze the 2D case in (b), which can be considered as a projection of the 3D case on a certain plane.

satisfied for three non-identical planes containing v and p_i , and we sketch the idea in Appendix A.

Proposition 4. *For a given $f(\cdot)$, p_i is recognized as a boundary node for $\mathcal{P}' \subseteq \mathcal{P}$ iff its DoN d satisfies:*

$$d \leq \frac{r_i (r_j \sin \alpha_1 + r_k \sin \alpha_2) - r_j r_k \sin(\alpha_1 + \alpha_2)}{\sqrt{r_j^2 + r_k^2 - 2r_j r_k \cos(\alpha_1 + \alpha_2)}},$$

where $r_i \leq 2R - \frac{(2R-r_j)(2R-r_k) \sin(\alpha_1+\alpha_2)}{(2R-r_j) \sin \alpha_1 + (2R-r_k) \sin \alpha_2}$. If $\alpha_1 = \alpha_2 = \alpha, r_j = r_k = r$ in particular, $d \leq 2R(1 - \cos \alpha)$.

Given a common viewpoint transformation (**Definition 1**), (r_i, r_j, r_k) and (α_1, α_2) are fixed to each point, but R can be tuned for different requirements on compensating d . If a transformation is chosen for individual points (**Definition 2**), all the parameters of the transformation can be tuned.

Finally, we answer Q2 raised in Sec. 3.2.1 by showing that, no matter which direction a boundary surface is facing, only a constant number of transformations are needed to recognize points on this surface.

Proposition 5. *If p_i is on a boundary (internal or external) surface of $\mathcal{P}$ and it satisfies $\frac{d}{\rho} < \frac{1}{2}$, at most **four** transformations (hence four viewpoints) are needed to recognize p_i as a boundary point. Here ρ is the inner radius of $\mathcal{N}_i$ defined in Definition 3.*

Intuitively, these four viewpoints are the vertices of a tetrahedron that encloses $\text{conv}(\mathcal{N}_i)$.

Summarizing **Propositions 2 to 5**, our boundary definitions based on $f(\cdot)$ have the following three properties:

1. **Safety I:** Points that are surely on the boundary defined by the convex hull of $\mathcal{P}$ will be identified as boundary points.
2. **Safety II:** Points that are surely not on any boundary defined by the already identified boundary points and visible from v will not be identified as boundary points.
3. **Tunable Liveness:** Points on a non-convex surface but with reasonable DoN can be identified with a small number of viewpoints (hence transformations).

Remark: It is important to note that, for a hypothetical surface passing through p_i , the ratio $\frac{d}{\rho}$ is actually a discrete indicator of its curvature at p_i . As we set an upper bound for $\frac{d}{\rho}$, the identified boundary points form a boundary surface that is low-curvature filtered from the real boundary surface. Moreover, it is possible that many nodes are identified as boundary nodes in extremely sparse 3D WSNs, because the condition $\frac{d}{\rho} < \frac{1}{2}$ may often be satisfied. Therefore, our approach performs well in relatively dense WSNs. This is actually a common feature for localized detection mechanisms, as they are all not concerned with global topology.

3.2.3 On-Line Boundary Detection Algorithm

Given the definitions of transformation and boundary, we are ready to present our UNFOLD for on-line boundary detection. UNFOLD involves mainly three steps for each node in a WSN:

I) Local Interactions Each node p_i exchanges location or range information with its neighbors in $\mathcal{N}_i$ to construct a local coordinate system. This step is trivial if the location information is available (through, for example, [62, 68]); otherwise a certain 3D embedding algorithm (e.g., [69]) is used to create the coordinate system using mutual distances. The origin of the coordinate system is the viewpoint v .

1. If the location information is available, we could afford to have a common view-point v (**Definition 1**), which can be required by certain applications (such as the BP oil spill example we gave in footnote 2).
2. Viewpoint v_{p_i} (**Definition 2**) specific to every p_i can always be applied. This is preferred if only range information is available, as otherwise we have to perform a costly procedure for constructing a global coordinate system with local range information (e.g., [70]).

II) Transformation Having the coordinates for all nodes in $\mathcal{N}_i \cup \{p_i\}$ with respect to an origin v (or v_{p_i}), node p_i applies the transformation given in (3.1) to these coordinates and obtains their images. This step involves only simple computations.

III) Convexity Test Each node p_i performs a convexity test to decide whether or not its image $\check{p}_i$ is on the convex hull of $f(\mathcal{N}_i) \cup \{v\}$ (or $f_{p_i}(\mathcal{N}_i) \cup \{v\}$). A basic algorithm is shown in **Algorithm 1**. We use $\vec{x}$ to represent the vector form of a point $\check{x}$. The idea

---

Algorithm 1: CVX-TEST

Input: $f(\mathcal{N}_i)$, $\check{p}_i$, a coord. system with v as the origin

Output: Binary indicator $boundary(\check{p}_i)$

```

1 forall the distinct node pair  $m_1, m_2 \in f(\mathcal{N}_i)$  do
2    $\vec{n} \leftarrow (\vec{p}_i - \vec{m}_1) \times (\vec{p}_i - \vec{m}_2)$ ;  $\vec{n} \leftarrow \frac{\vec{n}}{\|\vec{n}\|}$ ;  $b = \vec{p}_i \cdot \vec{n}$ 
3    $boundary(\check{p}_i) \leftarrow \mathbf{true}$ 
4   forall the  $m \in f(\mathcal{N}_i) \setminus \{m_1, m_2\}$  do
5     if  $\vec{m} \cdot \vec{n} > b$  then  $boundary(\check{p}_i) \leftarrow \mathbf{false}$ ; break ;
6   end
7   if  $boundary(\check{p}_i) = \mathbf{true}$  then break ;
8 end
```

---

is to try all possible planes determined by $\check{p}_i$ and another two points in $f(\mathcal{N}_i)$ (lines 1–3), and to check whether one of them is a *supporting plane* for $f(\mathcal{N}_i) \cup \{v\}$ (lines 4–5), i.e., if $f(\mathcal{N}_i) \cup \{v\}$ lies on one side of that plane.

To improve the efficiency, we apply the *divide-and-conquer* strategy. Observe that, after each inner loop (lines 4–5), points in $f(\mathcal{N}_i)$ can be totally ordered under $\geq$, according to their dot products with the normal $\vec{n}$. Therefore, instead of arbitrarily

choosing a (m_1, m_2) pair, we replace only one of the current two points with what is ordered first in $f(\mathcal{N}_i)$ (ties broken arbitrarily), such that the replaced point lies on the same side as v with respect to the new plane. In fact, no sorting is needed; the maximum point is naturally obtained at the end of each inner loop (lines 4–5). The algorithm terminates if neither points can be replaced: either because replacing either of them separates another from v or because the node ordered first in $f(\mathcal{N}_i)$ lies on the current plane (i.e., the plane is a supporting plane). The algorithm returns $\text{boundary}(p_i) = \mathbf{true}$ if the current plane is a supporting plane and returns $\text{boundary}(p_i) = \mathbf{false}$ otherwise. We call this enhanced algorithm CVX–TEST–DC. Note that the algorithm is conducted by individual nodes without the need for time synchronization. Therefore, UNFOLD is a **localized** algorithm requires only asynchronous operations; this makes UNFOLD extremely efficient.

3.2.4 Performance Analysis

We analyze the performance of UNFOLD on two aspects: namely time complexity of UNFOLD that also represents the energy efficiency of the algorithm and the robustness of UNFOLD against location or range errors.

3.2.4.1 Complexity Analysis

Our analysis on UNFOLD focuses on the transformation and convexity test steps, as the first step either has a negligible complexity if location information is available or otherwise involves well known procedures that are commonly applied in other proposals (e.g., [22]).

Assuming $\eta = |\mathcal{N}_i|$, the complexity of the transformation is obviously $\Theta(\eta)$, as we basically apply the transformation (3.1) to all nodes in $\mathcal{N}_i \cup \{p_i\}$. The complexity of the basic convexity test, CVX–TEST, is also obvious: as the outer iteration has $\eta(\eta - 1)$ loops and the inner iteration has $\eta - 2$ loops, the complexity is $\Theta(\eta^3)$. Consequently, the complexity (both average and worst-case) [71] of UNFOLD with CVX–TEST is $\Theta(\eta^3)$. Although this is same as that of the α -shape based boundary detection [22], the actual CPU time (hence energy consumption) of UNFOLD is much less (as shown in Sec. 3.4.2). This is because convexity test involves only vector operations (which are basically arithmetics), whereas α -shape construction entails complicated operations/procedures such as square root and solving equation systems. Moreover, we may

further reduce the complexity of UNFOLD by applying the enhanced convexity test: CVX-TEST-DC.

Proposition 6. *The average-case complexity of CVX-TEST-DC is $\mathcal{O}(\eta \log \eta)$.*

This complexity holds under the condition that the input is introduced in a random order, and it effectively reduces the average-case complexity of UNFOLD to $\mathcal{O}(\eta \log \eta)$. Though the worst-case complexity is $\Theta(\eta^2)$, those cases rarely happen based on our experience.

3.2.4.2 Error Analysis

Given the fact that the geometric boundary of a point cloud is not well defined, it is impossible to perform error analysis rigorously, as there is no ground truth to be compared with. One may be able to create a set of artificial “ground truth” boundary points in simulations, by deliberately sampling on the surface of the volume from which a point cloud is derived (which is the method that we will apply in Sec. 3.4.1 to evaluate the robustness of UNFOLD). However, unless those points are sampled extremely dense, there are still chances that certain “under the surface” points are detected as boundary points, regardless of which boundary detection mechanism is used. It is definitely unreasonable to categorize these points as detection errors, as they may well be on the surface of another volume that results in the same point cloud. Consequently, the error analysis we discuss here is rather qualitative.

If location information is available to every node, the error can be characterized by a small ball with radius ε around the expected location of the node, where ε can be the *mean square error* resulting from certain localization mechanism (e.g., GPS). If only range information between neighboring nodes is available, the initial errors come from the given ranging technique. However, this error will eventually be translated into location errors through a 3D embedding algorithm (e.g., [69]). In either case, ε cannot be too large compared with the radius of $\mathbf{conv}(\mathcal{N}_i)$, as otherwise it could be corrected (thus reduced) using local connectivity relations. Therefore, we may safely assume that ε is bounded by the radius of $\mathbf{conv}(\mathcal{N}_i)$.

For a boundary node (based on our definitions), the location errors may either decrease or increase DoN, if we consider a node p_i on or out of its local convex hull $\mathbf{conv}(\mathcal{N}_i)$ as having a non-positive DoN. Apparently, a decreased DoN has no impact

on boundary detection, whereas an increased DoN is somewhat compensated by our transformation, according to **Proposition 4**. For a non-boundary node, there are two cases: either it is very close to a boundary node or it indeed lies in the very interior of the point cloud. The former case, compared with a boundary node, is inversely affected by location errors. As the node is anyway close to a boundary node, a false positive does not really compromise boundary detection. The latter case can hardly be affected by location errors, given the boundedness of these errors. In summary, UNFOLD is very robust against location errors; we will demonstrate this in Sec. 3.4.1.

3.3 Localized Boundary Sphericalization (LBS) and its Applications

The outcome of our boundary detection algorithm is that every node may tell whether it is on some boundary or not. However, such individual boundary awareness is not sufficient in supporting some WSN functionalities. In this section, we propose LBS as a localized boundary parametrization technique that maps each boundary surface to a sphere. This allows three immediate applications: distinguishing internal boundaries from the external one, enabling 3D greedy geographic routing, and supporting surface routing (hence distributed coordinations) on a certain boundary.

One major limitation of LBS is that it only works for boundary surfaces of genus-0. As a surface of higher genus always has negative Gaussian curvature for part of it, it is not topologically equivalent to a sphere. Coping with boundaries of arbitrary topology is still an open problem that remains to be tackled in our future work.

3.3.1 Boundary Sphericalization

Let $\mathcal{B}$ be a boundary of $\mathcal{P}$. By our assumption, it is of genus-0, i.e., topologically equivalent to a unit sphere $\mathbb{S}^2$. Our goal is to parameterize $\mathcal{B}$ to $\mathbb{S}^2$, for which we need to construct a bijective map between $\mathcal{B}$ and $\mathbb{S}^2$. This can be done by solving a harmonic function $\phi : \mathcal{B} \rightarrow \mathbb{S}^2$.

$$\frac{\partial \phi}{\partial t} = -\Delta \phi_{\parallel}, \quad (3.2)$$

where $\Delta \phi(p)_{\parallel} = \Delta \phi(p) - \langle \Delta \phi(p), \mathbf{n}(p) \rangle \mathbf{n}(p)$ is the tangent component of $\Delta \phi$, $\mathbf{n}(p)$ is the normal vector of vertex p and $\langle \cdot, \cdot \rangle$ is the dot product. Intuitively speaking, the Laplace

$\Delta\phi \in \mathbb{R}^3$ is a 3D vector that, in general, is not on the tangent plane of the sphere, $\Delta\phi_{\parallel}$ is the projection of $\Delta\phi$ to the tangent plane. Thus, the image of vertex $\phi(p)$ is only allowed to diffuse along the tangential direction of the sphere. The function ϕ is harmonic if and only if its tangential component is zero, i.e., $\Delta\phi_{\parallel} = 0$. In differential geometry, it is known that the above harmonic function $\phi : \mathcal{B} \rightarrow \mathbb{S}^2$ is a conformal mapping, i.e., bijective, C^∞ smooth and angle preserving [72].

In our case, the boundary is sampled by a set of points (the boundary nodes being detected). Therefore, we first re-construct the surface by a localized triangulation [22]; it leads to an edge set $E_{\mathcal{B}}$. Then we propose a localized algorithm to perform a diffusion process that solves (3.2), as shown by **Algorithm 2**. Line 1 in **Algorithm 2**

Algorithm 2: Localized Boundary Sphericalization

Input: For each $p \in \mathcal{P}$, the stopping tolerance ϵ

Output: A harmonic function $\phi : \mathcal{B} \rightarrow \mathbb{S}^2$

- 1 For every boundary node p , compute its normal vector $\mathbf{n}(p)$, and initialize $\phi(p) = \mathbf{n}(p)$
  - 2 For every boundary node p periodically:
  - 3 **if** $\|\Delta\phi(p)\| > \epsilon$ **then**
    - 4 $\Delta\phi(p) = \phi(p) - \sum_{(p,q) \in E_{\mathcal{B}}} \omega_{pq}\phi(q)$
    - 5 $\Delta\phi(p)_{\parallel} = \Delta\phi(p) - \langle \Delta\phi(p), \mathbf{n}(p) \rangle \mathbf{n}(p)$
    - 6 $\phi(p) \leftarrow \phi(p) - \Delta\phi(p)_{\parallel}$
    - 7 Normalize $\phi(p)$
    - 8 Broadcast $\phi(p)$ to all $q : (p, q) \in E_{\mathcal{B}}$
  - 9 **end**
-

initializes the map $\phi : \mathcal{B} \rightarrow \mathbb{S}^2$ by the Gauss Map [73], that provides a mapping from every vertex on $\mathcal{B}$ to a corresponding point on a unit sphere $\mathbb{S}^2$. Note that this initial map is not bijective in general, since $\mathcal{B}$ may be concave. Line 4 is to compute the Laplacian of $\phi(p)$, where Δ is the discrete Laplace-Beltrami operator and ω_{pq} is the weight defined for edge $(p, q) \in E_{\mathcal{B}}$. A commonly used scheme is the uniform weight, i.e., $\omega_{pq} = \frac{1}{\deg(p)}$, where $\deg(p)$ is the degree of p . Then Line 5 is to compute the tangential component of $\Delta\phi(p)$ followed by the diffusion along the tangential direction in Line 6. Next, Line 7 normalizes $\phi(p)$ to ensure that the image is on the unit sphere. The above diffusion process is repeated until the tangential component of $\Delta\phi(p)$ is less

than the user-specified threshold ϵ .

3.3.2 Applications

3.3.2.1 Distinguishing Internal from External

Telling internal boundaries from external ones usually requires global information on a 3D mesh. As localization in WSNs often needs a boundary detection mechanism that can distinguish the two types of boundaries [62], it benefits from having a localized algorithm that achieves this goal. Our UNFOLD+LBS makes it fairly easy to distinguish internal and external boundaries on the parametric domain by using only local information.

Let $p \in \mathcal{P}$ is an arbitrary boundary node with spherical coordinates $\phi(p) \in \mathcal{S}^2$. Find a boundary triangle $\mathbf{t}$ such that $p \in \mathbf{t}$. Without loss of generality, $\mathbf{t}$ is determined by the tuple (p, m_1, m_2) that terminates Algorithm 1. Let $\mathbf{h}$ be the unique tetrahedron¹ with $\mathbf{t}$ as the boundary face. We denote $q \in \mathbf{h}, q \notin \mathbf{t}$ the opposite vertex of face $\mathbf{t}$, as shown by Fig. 3.4(a). Let g be the geometric center of $\mathbf{t}$ and $\vec{n}$ be the normal of $\mathbf{t}$. The direction of the normal $\vec{n}$ is defined according to the order of the three vertices of $\mathbf{t}$: $\vec{n}$ points to the outward of the tetrahedron $\mathbf{h}$, i.e., $\vec{gq} \cdot \vec{n} < 0$.

Note that each vertex of $\mathbf{t}$ has been mapped to the unit sphere $\mathcal{S}^2$ by ϕ resulting from LBS (see Algorithm 2). Let $\vec{n}_{\phi(\mathbf{t})}$ be the normal of the $\phi(\mathbf{t}) \in \mathcal{S}^2$, where the order of the three vertices of $\phi(\mathbf{t})$ is the same as $\mathbf{t}$. Then the node p is on the external boundary if and only if $\phi(p) \cdot \vec{n}_{\phi(\mathbf{t})} > 0$, as shown in Fig. 3.4(b); otherwise p is on the internal boundary. Intuitively speaking, this means that a boundary is external (resp. internal) if there exists a node inside (resp. outside) of it. Given an arbitrary boundary, this inside or outside relation cannot be recognized in a localized manner. It is our LBS that makes such an intuitive recognition procedure possible.

3.3.2.2 Supporting Geographic Routing

LBS may serve as a building block for performing greedy geographic routing with a guaranteed delivery. In a nutshell, greedy routing may fail when facing an internal boundary (the boundary of a hole) due to the so called “local minimum phenomena” [23]. Our solution relies on LBS to regularize the internal boundary and hence to

---

¹It stems from the uniqueness of Delaunay triangulation for general point set $\mathcal{P}$.

Figure 3.4: The rationale of distinguishing internal boundaries from the external one for 3D WSNs. Though both boundaries should be unit spheres under ϕ , we abuse the size a bit to differentiate them.

eliminate the local minimum phenomena. As this aspect is presented in a companion work, we refrain from providing details here. Instead, we briefly discuss a geographic routing on the boundary surface. As boundaries (especially event boundaries) are important from data collection point of view (because those are where information is produced), data routing as well as distributed coordinations (e.g., constructing quorum systems for data sharing [74, 75]) among nodes that share the same boundary are crucial to WSN functionalities.

Given the virtual coordinates of the boundary nodes on a sphere, the shortest path routing between any two nodes follows the great circle that is uniquely determined by the locations of these nodes and the sphere center. Assume the coordinate of the sphere center is $[0, 0, 0]^T$ and those of the two nodes are $\vec{p}_1 = [x_1, y_1, z_1]^T$ and $\vec{p}_2 = [x_2, y_2, z_2]^T$, such a great circle has a very simple representation by the following parametric equations: $x = x_1 \cos t + \tilde{x}_2 \sin t$, $y = y_1 \cos t + \tilde{y}_2 \sin t$, and $z = z_1 \cos t + \tilde{z}_2 \sin t$, where $[\tilde{x}_2, \tilde{y}_2, \tilde{z}_2]^T = (\vec{p}_1 \times \vec{p}_2) \times \vec{p}_1$ and $\times$ is the cross product between two vectors. The routing path can be specified by the source node and a trajectory-based forwarding [76] is then applied to follow the parametric curve (the great circle) until reaching the destination.

3.4 Simulations and Experiments

We first evaluate UNFOLD through both simulations and experiments. With simulations, we demonstrate the efficacy of UNFOLD in large scale WSNs. Through experi-

ments based on an implementation in MICAz Motes, we confirm UNFOLD’s superiority in efficiency over the most up-to-date proposal [22]. We have also implemented LBS in TinyOS [77], illustrated the convergence of LBS and also evaluated the performance of LBS in terms of convergence time through simulations with TOSSIM [78].

3.4.1 Simulations for UNFOLD

We first construct several 3D volumes to represent the physical events to be monitored. Then we randomly deploy sensor nodes in each volume and on the volume surface. As explained in Sec. 2.1.3, we may treat network and event boundary equivalently without loss of generality. Therefore, the goal of our simulations is to verify if UNFOLD can correctly identify those points that we have deliberately put on the volume boundaries (which, according to Sec. 3.2.4.2, is an inevitable but artificial setting). Although UNFOLD applies to arbitrary convex transmission volumes, we assume a regular volume, a ball with radius r , for every node in order to simplify our presentation. We choose the value of r such that the average size of $\mathcal{N}_i$ is about 40.

The implementation of UNFOLD in our simulator follows the protocol description in Sec. 3.2.3. Specifically, each node first exchanges location information with its neighbors. As our high-level simulator neglects the MAC effect, this information exchange is considered to be reliable. In practice, the reliability can be achieved through, for example, ARQ. After collecting the neighbor information, all the computations left for a node are strictly localized: transformation and convexity test.

In Fig. 3.5, four examples of the simulated WSNs are shown. The WSNs, shown in the left column, are designed to exhibit various 3D shapes we may face in atmospheric or ocean monitoring. We apply UNFOLD with either common viewpoint or local viewpoints to perform boundary detection, and the results are shown in the central and right columns, respectively. It is clearly shown that, while a single common viewpoint only detects the boundaries that face the viewpoint, a few local viewpoints detect the whole boundary. Note that, though the inversion used in [66] only detects an external boundary, our extension also detects the internal boundary, as shown by the second and third WSNs.

We also evaluate the robustness of UNFOLD under location errors. We assume each node has an error radius ε , such that the location information available to a node may be uniformly distributed within a ball with radius ε and centered at the real location.

Figure 3.5: To clearly illustrate the “shapes” of the WSNs, We attach their original 3D volumes to respective point clouds (a). For boundary detection with a common viewpoint, boundary nodes are marked on top of the original clouds (b). We remove the internal nodes when local viewpoints are applied to detecting the whole boundary (c).

We then vary ε to different fractions of r , and apply UNFOLD to detect the boundary. In general, a node that is either a real boundary node or detected as a boundary node may have four states:

- **Found:** detected as a boundary node
- **Correct:** a boundary node and also detected as one
- **Mistaken:** not a boundary node and detected as one
- **Missing:** a boundary node and not detected as one

In Fig. 3.6(a), we report the statistics in terms of these four states based on all the WSNs that we have simulated. We also report the statistics on the distance (in hop) from a mistaken/missing node to the closest correct boundary node in Fig. 3.6(b) and (c). The same metrics are used in [22]; we reuse them to facilitate comparisons.

(a) Algorithm efficiency (b) Mistaken distribution (c) Missing distribution

Figure 3.6: Boundary detection under location errors. We vary the error radius ε as different fractions of the transmission range r , and we both statistics (a–c) and an example (d–f).

As shown in Fig. 3.6(a), both found and correct nodes decrease with an increasing error radius. However, even with the worst case error $\varepsilon = r$, correct nodes still account

from around 50% the total boundary nodes. Also, the distributions in Fig. 3.6(b) and (c) show that both mistaken and missing nodes are not far from the real boundary (at most two hops but mostly within one hop). We finally provide one example, corresponding the first WSN in Fig. 3.5, in Fig. 3.6(d–f). The message conveyed by all these figures is clear: although increasing location errors may marginally affect the detected boundaries (which is characterized by the found nodes), these boundaries still well characterize the geometry of the network volume. Compared with the statistics reported in [22], UNFOLD appears to be more robust against the location errors: higher correct nodes percentage and shorter distances (from the mistaken and missing nodes) toward the real boundary.

3.4.2 Implementation and Experiments for UNFOLD

We implement UNFOLD in MICAz Motes, and compare UNFOLD with the Unit Ball Fitting (UBF) algorithm proposed in [22] in terms of CPU time for boundary detection. Our experiments focus only on local computation steps, as the local communications to exchange location/range information are common to both approaches. Therefore, we use the location information sampled from our simulated WSNs and inject these data to a MICAz Mote, so that we can directly start the actual boundary identification steps: transformation/convexity tests for UNFOLD, and ball tests for UBF. As the microcontroller of our MicaZ (ATMEL ATmega 128L) is fully occupied (i.e., never in idle mode or being interrupted) during both computations, we may roughly use the CPU time to also represent the energy consumption spent for computations. Consequently, our experiments also compare UNFOLD with UBF in terms of energy efficiency.

We show two sets of results, internal and boundary nodes, in Fig. 3.7. The comparisons are made between UBF and two versions of UNFOLD that differ in the convexity test algorithms. As an internal node is often the worst case for all the three schemes, Fig. 3.7 (a) serves as a confirmation of the complexity analysis (Sec. 3.2.4.1). Because the complexity of both UBF and CVX-TEST are $\Theta(\eta^3)$, the more than 10 times difference between them stems from the simple computations incurred by CVX-TEST. Another more than 10 times improvement brought by CVX-TEST-DC follows from its $\mathcal{O}(\eta \log \eta)$ complexity.

For boundary nodes (Fig. 3.7 (b)), all the three schemes may face cases varying from the best to the worst. Therefore, the CPU times are rather dispersed, but the

(a) CPU times for internal nodes. (b) CPU times for boundary nodes.

Figure 3.7: CPU times for MicaZ Mote.

roughly 100-time advantage of CVX-TEST-DC over UBF still holds. Note that the fairly constant (and almost always the smallest) CPU times for CVX-TEST-DC mostly indicate the cost of the transformation $f(\mathcal{N}_i)$. In summary, UNFOLD with CVX-TEST-DC is far more time and energy efficient than UBF. In fact, with a CPU time of tens of seconds, UBF may not even be eligible for an on-line boundary detection.

3.4.3 Simulations for LBS

We have also implemented LBS in TinyOS. However, as the effect of LBS can only be observed collectively for relatively large scale WSNs and our current MICAz-based testbed only has tens of nodes, we use simulations in TOSSIM (rather than experiments with our testbed) to evaluate the performance of LBS. Let π denote the period specified in Algorithm 2. We will evaluate the convergence time of LBS against different values of π .

We first illustrate the evolution of the diffusion process in Fig. 3.8. Given the original boundary (obtained by performing triangulation on the outcome of UNFOLD) shown in Fig. 3.8(a), the initial map on a unit sphere, Fig. 3.8(b), is rather messy, due to the many non-triangular edges (those induced by the non-bijectivity, i.e., foldover, of the Gauss map). The number of such edges decreases with the number of execution round until no such edge exist, which is shown by Fig. 3.8(c-f).

Figure 3.8: Illustration of the diffusion process of LBS with about 800 nodes and $\pi = 0.1s$.

We also show the impact of π on the convergence time in Fig. 4.7 for a 800-node WSN boundary. It is clear that the time to convergence increases monotonically in π (22s, 3s, and 2s for $\pi = 100ms$, 10ms, and 5ms, respectively), whereas the total number of rounds decreases with π (410, 300, 220 for $\pi = 100ms$, 10ms, and 5ms, respectively). As the number of messages sent is proportional to the number of rounds, the algorithm becomes more energy efficient but less time efficient with large values of π . The intuitive explanation is the following. When π gets small, the potential (broadcast) packet collisions become intensive. Consequently, the information each node collects about its neighbors decreases, which in turns requires more rounds to terminate the diffusion process. Therefore, we have a clear tradeoff between the time complexity and the energy efficiency of LBS, and we may tune the value of π to obtain a required (by

a certain application) balance between reducing latency and saving energy.

Figure 3.9: Convergence of LBS under different values of π .

3.5 Summary

In this chapter, we have investigated the challenging problems of on-line boundary detection and boundary regularization in 3D WSNs, and we have proposed LBDP as a concrete solution to these problems. As one component of LBDP, UNFOLD significantly speeds up the boundary detection by performing it in a transformed domain. This makes it perfectly suitable for on-line boundary detection in 3D WSNs that are deployed for monitoring time-variant physical events. We have demonstrated the efficiency and efficacy of UNFOLD through both simulations and experiments (using a MICAz-based testbed). Compared with an up-to-date proposal, UNFOLD is, on one hand, more robust against the location errors, and on the other hand, UNFOLD is far more efficient in terms of both computation time and energy consumption. As another component of LBDP, LBS serves as a distributed parametrization procedure to regularize an arbitrary boundary surface. The resulting virtual coordinates may enable many WSN functionalities, such as distinguishing internal boundaries from the external one, as well as geometric routing on a boundary.

Chapter 4

Autonomous Deployment for Load Balancing k -Surface Coverage in Sensor Networks

The coverage of sensor nodes is very crucial to the functionality of WSNs. It is well acknowledged that 1-coverage sensor deployment may suffer from the vulnerability of sensor nodes. Therefore, multiple-coverage (k -coverage) is often applied to enhance the fault tolerance in face of node failures. Although the problem of k -area coverage has been intensively investigated for static WSNs, how to arrive at an autonomous k -coverage sensor deployment that optimizes certain objectives in mobile WSNs still faces both theoretical and practical difficulties. Moreover, only a handful of centralized algorithms have been proposed to elevate 2D area coverage to 3D surface coverage. In this chapter, we present a practical algorithm APOLLO (*Autonomous dePlOyment for Load baLancing k -surface cOverage*) to move sensor nodes toward k -surface coverage, aiming at minimizing the maximum sensing range required by the nodes. With the help of our on-line boundary detection technique, APOLLO enables purely autonomous node deployment as it only entails localized computations. We prove the termination of the algorithm, as well as the (local) optimality of the output. We also show that our optimization objective is closely related to other frequently considered objectives for 2D area coverage. Therefore, our practical algorithm design also contributes to the theoretical understanding of the 2D k -area coverage problem. Finally, we use extensive simulation results to both confirm our theoretical claims and demonstrate the efficacy

of APOLLO.

4.1 Introduction

One of the major functions of WSNs is to monitor a certain area in terms of whatever physical quantities demanded by applications [3]. In achieving this goal, a basic requirement imposed onto WSNs is their *area/surface coverage*:¹ it indicates the monitoring quality of WSNs. Whereas many research proposals focus on either analyzing the performance of static sensor deployments (e.g., [37, 45]) or scheduling sensor activity to retain the coverage of given deployments (e.g., [38, 79]), there exists an unfailing trend in seeking autonomous deployments assisted by mobile sensor nodes to arrive at certain predefined objectives (e.g., [8, 9]). Our proposal in this chapter falls into this later trend.

Due to the vulnerability of sensor nodes, multiple-coverage (k -coverage) is often applied to enhance the fault tolerance in face of node failures (e.g., [38, 45]). In addition, k -coverage may yield higher sensing accuracy through data fusion [80]. Existing approaches in achieving k -coverage rely on either randomized (e.g., [37, 81]) or regular (e.g., [2, 45]) deployments. Whereas randomized deployments require a substantially denser network (e.g., [37, 81]), regular deterministic deployments serve only as theoretical guidelines [2, 45] as they often require centralized coordinations and may not accommodate irregular network regions. Also, if the physical phenomena under surveillance change after the deployment, the cost of re-deployment can be huge. Therefore, autonomous deployments, when being made possible by employing movable nodes [7], are good complements to the randomized or regular deployments: they may achieve a density comparable to that of regular deployments, while being more adaptive to irregularity and variations of the network regions.

However, existing techniques for autonomous deployments may only handle 1-coverage, and extending them to k -coverage is highly nontrivial. First of all, autonomous deployments through (node) motion control require each node to compute its coverage in a localized manner (i.e., relying as much as possible on close-by nodes).

---

¹For WSNs deployed on 2D planes, we only focus on approaches concerning area coverage (e.g., [37, 45]), as opposed to the *point coverage* (e.g., [34, 35, 36]). Surface coverage is an elevation of area coverage from 2D planes to 3D surfaces (terrains), making the deployment more useful in practice (e.g., for monitoring volcanos).

Although quite a few localized algorithms have been proposed to perform such computations for 1-coverage (e.g., [8, 82]), no algorithm, to the best of our knowledge, exists for localized k -coverage computations. Secondly, even if the k -coverage computations can be performed locally, there is no guarantee whether a motion control strategy may converge, due to the significant difference between 1-coverage and k -coverage. Thirdly, existing approaches are almost heuristics that offer no provable guarantee on the quality of the eventual deployment. Finally, as indicated in [48], extending 2D area coverage to 3D surface coverage for WSNs deployed on 3D terrains may incur new challenges even for 1-coverage.¹ By far, only a handful of proposals employ (partially) centralized algorithms to handle the deployment for 1-surface coverage [48, 49]; no pure autonomous deployment mechanism has ever been proposed. These are exactly the problems we want to tackle in this thesis.

In this chapter, we consider the problem of moving sensor nodes towards k -coverage. In particular, we assume that nodes are randomly deployed initially and that each node has a tunable sensing range. Our goal is to cover a certain *monitored area or surface* to the extent that every point in this area/surface is at least monitored by k sensor nodes and that the maximum sensing range used by the nodes is minimized. As a larger sensing range implies a larger energy consumption of a node, our APOLLO approach aims at balancing the sensing load (thus prolonging network lifetime) while guaranteeing k -coverage, with the help of mobile nodes. The main contributions we are making in this chapter are:

- We design the APOLLO algorithm such that it executes in a localized manner, i.e., relying only on information from close-by nodes.
- We prove the termination of APOLLO as well as the (local) optimality of its output.
- We discuss the relation between the output of APOLLO and other commonly used optimization objectives for 2D area coverage, which provides a better understanding of optimal k -coverage deployments whose theoretical characterizations are hard to obtain under general settings.

---

¹The 3D *volume coverage* problem for WSNs deployed in 3D spaces can actually be solved by straightforward extensions from the 2D area coverage techniques, as the same (Euclidean) metric applies to both cases.

- We show that, by using geodesic distance [83] to replace the conventional Euclidean metric, APOLLO can be naturally extended to handle autonomous deployments on 3D surfaces.
- Since motion and communication cost cannot be neglected, our algorithm allows for a tradeoff between the two factors. Through extensive simulations driven by realistic power consumption data, we show that the existing hardware platform can afford the energy consumption of our algorithm in terms of motion and communication.

To the best of our knowledge, we are the first to tackle the problem of k -coverage autonomous deployment, for both 2D areas and 3D surfaces.

The remaining of this chapter is organized as follows. We formally define our model and problem in Sec. 4.2, in which we also review the basic mathematical tools. In Sec. 4.3, we present our APOLLO algorithm for 2D k -area coverage and analyze its performance, we also discuss the interpretation of our solution with respect to other optimization frameworks. We then extend APOLLO to handle 3D k -surface coverage in Sec. 4.4. The efficacy of APOLLO is further confirmed by extensive simulation results reported in Sec. 4.5. We finally conclude this chapter in Sec. 4.6.

4.2 Problem Definition and Mathematical Background

In this section, we first present the system model and define our optimization problem. We also briefly introduce the mathematical basics relevant to our algorithm design. To simplify the exposition, the above discussions are all carried out for a 2D plane with Euclidean metric. Then, we show that a 2D plane can be generalized to a 3D surface by replacing Euclidean metric with geodesic distance, and we also introduce the mathematical tools to perform this map locally, and hence to allow a straightforward migration of our model, problem, and algorithm from area coverage to surface coverage.

4.2.1 System Model

We assume a WSN consisting of a set $\mathcal{N} = \{n_1, \dots, n_N\}$ of sensor nodes, and $|\mathcal{N}| = N$. Let $\mathcal{U} = \{u_1, \dots, u_N\}$ denote the locations of sensor nodes. The nodes are initially deployed arbitrarily on a 2D *targeted area* $\mathcal{A}$. Each node n_i is equipped with certain

mechanisms (e.g., motors plus wheels) that allow it to gradually change its location u_i [7]. We also suppose that nodes are equipped with bumper sensors to detect and avoid obstacles in the targeted area [84]. All nodes have an identical transmission range γ , and we denote by $\mathcal{N}(n_i) = \{n_j | \|u_i - u_j\|_2 \leq \gamma, i \neq j\}$ the one-hop neighbors of node n_i , i.e., the nodes within the transmission range of n_i .

We define the omnidirectional sensing model as a disk centered at u_i with sensing range r_i . We assume the sensing ranges are adjustable according to different application requirements [38, 85]. A point $v \in \mathcal{A}$ is said to be covered by node n_i iff the Euclidean distance between v and node location u_i is no longer than r_i , i.e., $\|v - u_i\|_2 \leq r_i$. We use $f(v, u_i, r_i)$ to indicate if v is covered by node n_i .

$$f(v, u_i, r_i) = \begin{cases} 1 & \text{if } \|v - u_i\|_2 \leq r_i \\ 0 & \text{otherwise} \end{cases} \quad (4.1)$$

In other words, if v is covered by n_i , $f = 1$; otherwise, $f = 0$. We rely only on the ranging ability of each node to construct a local coordinate system for motion control, so location information is not essential to our algorithm.

In terms of energy cost, we consider only the cost induced by the sensing activities of a node. Because our network deployment strategy aims to achieve a **constant** (and long-term) coverage by moving sensor nodes in the **initial** phase, the communication cost becomes negligible as the data transmission activities only take place sporadically,¹ while the energy spent in moving is only a one-time investment. We assume that the energy consumed by a sensor node n_i is an increasing function $E(r_i)$ of its sensing range r_i , and this function is identical for all nodes. In other words, with a certain amount of energy $E(r_i)$, n_i can only cover the points $\{v \in \mathcal{A} | \|v - u_i\|_2 \leq r_i\}$.

4.2.2 Problem Formulation

The node locations and sensing ranges, $\{u_i, r_i\}_{i=1, \dots, N}$, define a network deployment with a certain coverage.

Definition 4. A network deployment $\{u_i, r_i\}$ is said to achieve k -coverage iff for any point $v \in \mathcal{A}$, there exist **at least** k sensor nodes covering it, or $\sum_i f(v, u_i, r_i) \geq k$.

---

¹In many applications, data transmissions are invoked only when *anomalies* or *events of interest* are detected [85]. Even for applications collecting all sensed data, temporarily storing the data at individual nodes for a proper compression is desirable for energy efficiency purpose [86]. Also, we note that a sensing task consumes energy in many components, including sensing, sampling, processing, and memory accessing. The total power consumption can be higher than that of wireless transmissions.

To allow sensor nodes properly cover the targeted area, we divide $\mathcal{A}$ into several disjoint areas $\{\mathcal{A}_j^k\}_{j=1,2,\dots}$, and at least k sensor nodes are allocated to take charge of each area. In other words, each sensor node n_i takes care of multiple subareas, and we indicate this relation by $n_i(\mathcal{A}_j^k)$: it equals 1 if n_i covers $\mathcal{A}_j^k$; otherwise 0. We also denote by $\mathcal{A}_{n_i}^k$ the area covered by n_i : we have $\mathcal{A}_{n_i}^k = \bigcup_{n_i(\mathcal{A}_j^k)=1} \mathcal{A}_j^k$. Obviously, the sensing range r_i of n_i is determined by the farthest point in $\mathcal{A}_{n_i}^k$ from u_i , i.e., $r_i = \max_{v \in \mathcal{A}_{n_i}^k} \|v - u_i\|_2$, so that $\mathcal{A}_{n_i}^k$ can be totally covered by n_i .

Our k -Coverage Sensor Deployment Problem (k -CSDP) can be formulated as follows.

$$\begin{aligned} & \text{minimize} && R \\ & \{u_i\}, \{\mathcal{A}_j^k\}, \{n_i(\mathcal{A}_j^k)\} \end{aligned} \quad (4.2)$$

$$\text{subject to} \quad \sum_{i=1}^N f(v, u_i, r_i) \geq k, \quad \forall v \in \mathcal{A} \quad (4.3)$$

$$\|v - u_i\|_2 \leq R, \quad \forall i, v \in \mathcal{A}_{n_i}^k \quad (4.4)$$

$$\mathcal{A}_{j_1}^k \cap \mathcal{A}_{j_2}^k = \emptyset, \quad \bigcup_j \mathcal{A}_j^k = \mathcal{A} \quad (4.5)$$

Literally, k -CSDP aims at determining the node locations $\{u_i\}$, the area partition $\{\mathcal{A}_j^k\}$, and the covering relations $\{n_i(\mathcal{A}_j^k)\}$, such that the maximum sensing range among all nodes is minimized. As energy consumption is an increasing function of sensing range, k -CSDP is equivalently balancing the energy consumption over a whole WSN and hence maximizing the lifetime of the WSN. As the problem is generally not convex due to its non-convex feasible region, we have to be contented with **local minimum**.

4.2.3 High Order Voronoi Diagram

We hereby briefly introduce the ideas and theories on *high order Voronoi diagram* [87]. They are key to our autonomous deployment strategy.

In a k -order Voronoi diagram, the targeted area $\mathcal{A}$ is segmented into $\hat{N}^k$ disjoint areas $\{\mathcal{V}_j^k\}_{j=1,\dots,\hat{N}^k}$ ¹, each of which is associated with k closest generators (sensor nodes in our case), i.e., a subset $\mathcal{N}_j^k \subseteq \mathcal{N}$ with $|\mathcal{N}_j^k| = k$. The k -order Voronoi cell $\mathcal{V}_j^k$ is defined as

$$\mathcal{V}_j^k = \{v \in \mathcal{A} \mid \|v - u_i\|_2 \leq \|v - u_{i'}\|_2, \forall n_i \in \mathcal{N}_j^k, n_{i'} \in \mathcal{N}/\mathcal{N}_j^k\} \quad (4.6)$$

---

¹In 1-order Voronoi diagram, the number of Voronoi cells equals the number of generators (i.e., $\hat{N}^1 = N$), while in generalized k -order Voronoi diagram ($k \geq 1$), $\hat{N}^k$ is $\mathcal{O}(k(N - k))$ [87].

The set $\mathcal{N}_j^k$ is called the *generator set* of $\mathcal{V}_j^k$. It is straightforward to see that each sensor node n_i is associated with multiple Voronoi cells. Let $\mathcal{V}_{n_i}^k$ denote the union of the Voronoi cells for which n_i serves as a generator; we term $\mathcal{V}_{n_i}^k$ the *dominating region* of n_i (hence n_i the *dominator* of $\mathcal{V}_{n_i}^k$). We also have the following proposition.

Proposition 7. *A point $v \in \mathcal{A}$ is said to belong to $\mathcal{V}_{n_i}^k$ iff there exist **at most** $k - 1$ other generators such that their distance to v is less than $\|v - n_i\|_2$.*

Proof. Assume $v \in \mathcal{V}_{n_i}^k$ but there were another k nodes $\{n_j\}_{j \in J, i \notin J, |J|=k}$ such that $\|u_j - v\|_2 < \|u_i - v\|_2$. Then the point v would **strictly** belong to the set of k -order Voronoi cells generated by $\{n_j\}$, which does not include n_i as a generator; a contradiction. Conversely, if there are at most $k - 1$ nodes, $\{n_j\}_{j \in J, i \notin J, |J| \leq k-1}$, closer to v than n_i , we can find the set of v 's k -nearest nodes (which obviously contains $\{n_j\} \cup \{n_i\}$) to generate a set of k -order Voronoi cells containing v . As n_i is a generator, $v \in \mathcal{V}_{n_i}^k$. $\square$

Based on the above proposition, assuming

$$\mathcal{S}_{n_i}^k(v) = \{n_j \in \mathcal{N} \mid \|u_j - v\|_2 < \|u_i - v\|_2, j \neq i\}$$

we can re-define the dominating region of n_i as

$$\mathcal{V}_{n_i}^k = \{v \in \mathcal{A} \mid |\mathcal{S}_{n_i}^k(v)| \leq k - 1\} \quad (4.7)$$

We illustrate k -order Voronoi partition ($k = 1, 2, 3, 4$) generated by 30 nodes in Fig. 4.1. The cells shown in each figure are $\{\mathcal{V}_j^k\}$. Taking 2-order Voronoi partition for example, as shown in Fig. 4.1(b), the area enclosed by red (resp. green) polygon is actually the dominating region of the red node (resp. green node). The hatched area is the Voronoi cell generated by the two nodes, i.e., the points in this area are closer to the two nodes than any other nodes.

4.2.4 Generalization to 3D Surfaces

For WSN deployments on a terrain modeled as a 3D surface, the conventional Euclidean metric is no longer appropriate. Serving as the generalization of “straight line” in curved surface (e.g., 3D surfaces), *geodesic* is the shortest path between two given points on the surface [83]. Therefore, geodesic distance metric is a natural choice for measuring distance on a 3D surface. By replacing Euclidean metric with geodesic distance, we may migrate the aforementioned model and problem definitions directly from 2D planes to 3D surfaces.

Figure 4.1: k -order Voronoi partition for $k = 1, 2, 3, 4$. The disks at the backdrop represent the (overlapping) sensing ranges of individual sensor nodes.

4.2.4.1 Problem Description on 3D Surfaces

Considering the case where a WNS $\mathcal{U}$ is deployed on the 3D surface $\mathcal{M}$, we use geodesic distance as the distance metric. In particular, we replace all the distant metric $\|u - v\|_2$ used for 2D planes by $g(u, v)$, the geodesic distance between u and v . As a result, all the definitions in Sec. 4.2.1 to 4.2.3 can be directly migrated to 3D surface. For example, k -CSDP and high order Voronoi diagram can be redefined using geodesic metric. The only difference here is that, while $\mathcal{A}$ does not need an explicit characterization, the 3D surface $\mathcal{M}$ is often represented by a triangular mesh that is given to all nodes during the initialization phase. In order to compute geodesic distance on $\mathcal{M}$, we adopt the Improved Chen & Han's (ICH) algorithm [88].

4.2.4.2 Logarithm and Exponential Maps

As we need a local coordinate system to compute k -order Voronoi diagrams in a localized manner for every node, we cannot rely on a parametrization method such as Ricci flow due to its global nature and high computational cost. We instead apply the ICH algorithm to compute an logarithm/exponential map [83] around a certain node, which constructs a (local) geodesic polar coordinate system on the curved surface as follows.

Given a point $u \in \mathcal{M}$, we designate a local coordinate system on its tangent plane $\mathcal{T}_u$ centered at u . A *logarithm map* $\exp_u^{-1} : \mathcal{M} \rightarrow \mathcal{T}_u$ maps the points on $\mathcal{M}$ to $\mathcal{T}_u$. In particular, for any other point $v \in \mathcal{M}$ in a sufficiently small neighborhood of u , there

is a unique geodesic $\mathfrak{G}_u(v)$ extending from u to v . As $\mathfrak{G}'_u(v)$, the tangent of $\mathfrak{G}_u(v)$, is obviously tangent to $\mathcal{M}$ at u , $\mathfrak{G}'_u(v) \in \mathcal{T}_u$. Therefore, v can be mapped to $\mathcal{T}_u$ with geodesic polar coordinates $\{g(u, v), \theta_{u, v}\}$ where $g(u, v)$ is the geodesic distance between u and v and $\theta_{u, v}$ is the polar angle of $\mathfrak{G}'_u(v)$ on $\mathcal{T}_u$. Consequently, the geodesic coordinates on $\mathcal{M}$ around u can be transformed to normal coordinates under any orthogonal basis $\{\mathbf{e}_1, \mathbf{e}_2\}$ with origin u . The inverse of logarithm map is called *exponential map*. We illustrate the log/exp map in Fig. 4.2. Here the ICH algorithm is used to trace the

Figure 4.2: The log/exp maps at point u introduces a mapping between a vector originating at u on $\mathcal{T}_u$ and a geodesic beginning at u on $\mathcal{M}$. The circles on $\mathcal{T}_u$ are mapped to the contours of the geodesic function on $\mathcal{M}$.

geodesic $\mathfrak{G}_u(v)$ such that any point v in the neighborhood of u on $\mathcal{M}$ can get its geodesic polar coordinates on $\mathcal{T}_u$. The ICH algorithm [88] maintains a “wavefront”, propagating outward from u . Suppose we need a local geodesic coordinate system within a radius r , the ICH algorithm takes $\mathcal{O}(n^2)$ time to complete the propagation, where n is the number of vertices of the triangular mesh $\mathcal{M}$ within geodesic distance r from u . As the ICH algorithm is the dominating computational component, constructing $\mathcal{T}_u$ has $\mathcal{O}(n^2)$ time complexity.

4.3 APOLLO: Localized k -Coverage Node Deployment Algorithm

In this section, we first develop two optimality conditions for k -CSDP. Then we present the APOLLO algorithm in detail. The correctness of APOLLO is then proven, and we finally discuss the relation between k -CSDP and other optimization problems related to k -coverage deployment, along with the corresponding properties of APOLLO.

4.3.1 Optimal Conditions

To motivate our algorithm design, we develop two optimality conditions for k -CSDP. Firstly, we show that, if we fix $\{u_i\}$, then k -order Voronoi diagram is the optimal solution to k -CSDP.

Proposition 8. *If we fix the sensor locations $\{u_i\}_{i=1,\dots,N}$, the k -order Voronoi diagram $\{\mathcal{V}_1^k, \dots, \mathcal{V}_{N^k}^k\}$ generated by $\{u_i\}_{i=1,\dots,N}$ is an optimal partition of $\mathcal{A}$. Also, $n_i(\mathcal{V}_j^k) = 1$ if $\mathcal{V}_j^k \subseteq \mathcal{V}_{n_i}^k$; otherwise $n_i(\mathcal{V}_j^k) = 0$.*

Proof. The proof is by contradiction. Suppose for fixed $\{u_i\}_{i=1,\dots,N}$, there exists an optimal solution to k -CSDP denoted by $R^*, \{\bar{\mathcal{A}}_j^k\}, \{n_i(\bar{\mathcal{A}}_j^k)\}$. Let $r_i^* = \max_{v \in \bar{\mathcal{A}}_{n_i}^k} \|v - u_i\|_2$ and $r_i^V = \max_{v \in \mathcal{V}_{n_i}^k} \|v - u_i\|_2$. Also assume that the optimal value is obtained for n_i , i.e., $R^* = r_i^*$. If $r_i^V = r_i^*$, then it is straightforward to see that $r_i^V = \max_i \{r_i^V\}$, otherwise a contradiction to the definition of $\mathcal{V}_{n_i}^k$: some regions are not covered by the k -closest nodes. Therefore, in this case the k -order Voronoi diagram is equally optimal. If $r_i^V > r_i^*$, it means that n_i could cede a certain region to have it covered by other nodes while reducing $\max_i \{r_i^V\}$. However, this again contradicts the definition of $\mathcal{V}_{n_i}^k$: as n_i is already one of the k -closest nodes that can cover the ceded region, ceding this region to some other nodes would increase $\max_i \{r_i^V\}$. $\square$

Before stating the second optimality condition, we need the following definition.

Definition 5. *Given an arbitrary set S in Euclidean space, the Chebyshev center u_c is defined as:*

$$u_c = \arg \min_{\hat{u}} \left(\max_{u \in S} \|u - \hat{u}\|_2 \right)$$

Using this definition, for an arbitrary partition $\{\mathcal{A}_j^k\}$ and its dominator allocation $\{n_i(\mathcal{A}_j^k)\}$, the optimal locations of $\{n_i\}$ can be obtained.

Proposition 9. *If we fix the partition $\{\mathcal{A}_j^k\}$ and its dominator allocation $\{n_i(\mathcal{A}_j^k)\}$, the optimal sensor location u_i^* for k -CSDP is given by the Chebyshev center of $\mathcal{A}_{n_i}^k$.*

Proof. As n_i needs to cover $\mathcal{A}_{n_i}^k$ and the objective of k -CSDP is to minimize the maximum sensing range among all sensors, the optimal solution under a fixed partition is achieved if each sensor individually minimizes its own sensing range. This exactly coincides with the property of Chebyshev center, hence the proposition follows. $\square$

4.3.2 The Algorithm

Given the two optimality conditions stated in Sec. 4.3.1, we immediately have an iterative algorithm to solve k -CSDP. The pseudo-codes of our APOLLO algorithm are presented in **Algorithm 3**. The algorithm proceeds in rounds. At the beginning of each round, the k -order Voronoi diagram is computed for the whole WSN, resulting in $\{\mathcal{V}_1^k, \dots, \mathcal{V}_{\hat{N}^k}^k\}$ along with $\{\mathcal{V}_{n_1}^k, \dots, \mathcal{V}_{n_N}^k\}$ (Line 2). Then each node computes the Chebyshev center of its dominating region (Line 3), and moves to that location to end this round (Line 4-6). The algorithm terminates if each node is indeed located at the Chebyshev center of its dominating region. As a perfect matching is impossible in face of numerical errors, we use a small value ε as the stopping tolerance: the algorithm terminates if the distance from the node's current location to the Chebyshev center of its dominating region is smaller than ε . Also, in order to avoid oscillation, a step size $\alpha < 1$ is chosen to confine the motion of the nodes. At the termination, each node tunes its sensing range to be the minimum value (the circumradius of its dominating region) that covers its dominating region. As a dominating region is a polygon, we

---

Algorithm 3: APOLLO

Input: For each $n_i \in \mathcal{N}$, initial location $u_i^{(0)}$, stopping tolerance ε

Output: $\{u_i^*\}$ and $\{r_i^*\}$

- 1 For every node $n_i \in \mathcal{N}$ periodically (every τ ms):
  - 2   Compute its dominating region $\mathcal{V}_{n_i}^k$
  - 3   Compute the Chebyshev center c_i of $\mathcal{V}_{n_i}^k$
  - 4   **if** $\|u_i - c_i\|_2 > \varepsilon$ **then**
  - 5 $u_i^+ \leftarrow u_i + \alpha(c_i - u_i)$ /* α is the step size*/
  - 6   **end**
  - 7 $u_i^* \leftarrow c_i, \quad r_i^* \leftarrow \max_{u \in \mathcal{V}_{n_i}^k} \|u - u_i\|_2$
-

apply Welzl’s algorithm [89] to compute the Chebysehev center by taking the vertices of the region as the input.

Similar algorithms have been applied in [8, 9], but they were used to indirectly optimize a different objective (see our discussions in Sec. 4.3.3). Consequently, their approaches do not abide by the optimality conditions and employ a very different termination condition. Therefore, their termination proofs do not apply to our case even for $k = 1$. Most importantly, as our algorithm deals with a more general k -coverage, we are facing the following new challenges.

- C1 How to compute $\mathcal{V}_{n_i}^k$ in a localized manner without involving all nodes in a WSN?
- C2 Does the algorithm terminate for $k \geq 1$? If yes, what is the relation between termination and the step size α ?

We tackle these two challenges in the following.

4.3.2.1 Localized Algorithm for Computing $\mathcal{V}_{n_i}^k$

Unlike 1-order Voronoi diagram that can be computed (mostly) by only interacting with one-hop neighbors $\mathcal{N}(n_i)$ of a given node n_i , $\mathcal{N}(n_i)$ may not be sufficient to obtain k -order Voronoi cells, especially when k is large. The reason is simple: at least $k + 1$ nodes should be involved to compute a dominating region of n_i [90]. Therefore, our localized algorithm for computing $\mathcal{V}_{n_i}^k$ works in an expanding ring manner, as shown in **Algorithm 4**.

Basically, we expand the search ring ρ with a granularity of the transmission range γ (line 2). As expending ρ beyond γ will need multi-hop communication and the hop number is always an integer, it makes no sense to apply a smaller granularity. We use the embedding algorithm proposed in [69] to construct a local coordinate system (line 4). If the location information is available, this step is not necessary. Under the constructed coordinate system, we check whether the circle centered at u_i with a radius $\rho/2$ is not dominated by n_i anymore (lines 5 to 8, based on **Proposition 7**). Because the Voronoi edges computed given $\mathcal{N}(n_i, \rho)$ divide the circle with radius $\rho/2$ into a finite number of arcs, each of which either fully dominated by n_i or not at all, we only need to check an arbitrary point on each arc in the actual implementation. The ring expending terminates if the answer becomes true. Finally, we compute $\mathcal{V}_{n_i}^k$ using only nodes falling into the current search ring (line 10).

Figure 4.3: The dominating region of the central node in k -order Voronoi diagram for k ranging from 1 to 12. The central node needs to collect location (or range) information from its neighboring nodes (the dark nodes) via multi-hop communication according to **Algorithm 4**. Additionally, we illustrate multi-hop transmission range using red circles in (a). While the cases for $k = 1$ can be handled by involving only the 6 closest nodes (1-hop neighbors) to the central node, computing the 2-, 3- and 4-order dominating regions requires 2-hop neighbors. When $k > 4$, all sensor nodes within 3 hops are involved.

We need another lemma to show that $\mathcal{V}_{n_i}^k$ computed by **Algorithm 4** is indeed the one that would be computed in a centralized manner using global information.

Lemma 1. *If the dominating region of n_i is enclosed by a circle centered at u_i with a radius of $\rho/2$, then it is fully determined by all the nodes located within another circle centered at u_i with a radius of ρ .*

Proof. For a disk $\odot(u_i, \rho/2)$ centered at u_i with radius $\rho/2$, if $\mathcal{V}_{n_i}^k \subset \odot(u_i, \rho/2)$, the boundary of $\mathcal{V}_{n_i}^k$ also belongs to $\odot(u_i, \rho/2)$. According to the definition of Voronoi cells, the cell boundary consists of bisectors, each of which is determined by two generators. For $\mathcal{V}_{n_i}^k$, one generator is n_i , and all other generators can be obtained by going through each segment (or bisectors) on the boundary of $\mathcal{V}_{n_i}^k$ and identifying another generator that determines this bisector along with n_i . Since the boundary of $\mathcal{V}_{n_i}^k$ belongs to $\odot(u_i, \rho/2)$, all generators of $\mathcal{V}_{n_i}^k$ belong to $\odot(u_i, \rho)$. $\square$

The correctness of our algorithm is immediate from this lemma: as the algorithm terminates when n_i is not dominating the circle centered at u_i with a radius $\rho/2$ anymore, the nodes falling into $\odot(u_i, \rho)$ are sufficient to compute $\mathcal{V}_{n_i}^k$. In Fig. 4.3, we

---

Algorithm 4: Localized $\mathcal{V}_{n_i}^k$ Computation

Input: For each $n_i \in \mathcal{N}$, initial ring radius $\rho = 0$

Output: $\mathcal{V}_{n_i}^k$

```

1 repeat
2    $\rho \leftarrow \rho + \gamma$ ;  $out \leftarrow \mathbf{true}$ 
3    $\mathcal{N}(n_i, \rho) \leftarrow \{n_j | \|u_j - u_i\|_2 < \rho\}$ 
4   Construct a local coordinate system using  $\mathcal{N}(n_i, \rho)$ 
5   foreach  $v \in \mathcal{A}$  s.t.  $\|v - n_i\|_2 = \rho/2$  do
6      $\hat{\mathcal{S}}_{n_i}^k(v) \leftarrow \{n_j \in \mathcal{N}(n_i, \rho) | \|u_j - v\|_2 < \|u_i - v\|_2, j \neq i\}$ 
7     if  $|\hat{\mathcal{S}}_{n_i}^k(v)| < k$  then  $out \leftarrow \mathbf{false}$ ; break ;
8   end
9 until  $out = \mathbf{true}$ ;
10 Compute  $\mathcal{V}_{n_i}^k$  based on  $\mathcal{N}(n_i, \rho)$ 

```

---

demonstrate this sufficiency using k -order dominating region ($k = 1$ to 12) in a regularly deployed WSN. The regular deployment is chosen to facilitate exposition, our algorithm works for any arbitrary deployments.

For a node n_i on the boundary, the search ring will never stop expanding, as the arc that is out of the network coverage will always need the domination of n_i . To cope with this issue, we first employ a boundary detection service (e.g., our UNFOLD algorithm presented in Chapter 3) to indicate which nodes are on or close to the boundary. Based on the boundary awareness, the boundary node executes the circle checking procedure (lines 5 to 8) only for the arc that lies within the network coverage area, as shown in Fig. 4.4. Finally, the boundary node calculates its dominating region, using $\mathcal{N}(n_i, \rho)$ as well as the searching ring to determine the boundary of the dominating region. For an initial random deployment in which nodes only occupy a small fraction of $\mathcal{A}$, this procedure has the effect of “pushing” boundary nodes outwards, hence expanding the network coverage to the whole $\mathcal{A}$. In fact, such a constrained checking procedure should always be used by nodes on the boundary of $\mathcal{A}$, the difference is that $\mathcal{A}$ ’s boundary, known in advance to the nodes, serves as a natural boundary for a dominating region.

Figure 4.4: Recognizing the network boundary (the green dashed curve), a boundary node (dark) determines a searching ring (the outer circle) and check the half-radius arc within the network coverage area (the inner red arc). It then calculates its dominating region (the blue area) with $\mathcal{N}(n_i, \rho)$, while the searching ring helps to determine part of the boundary.

4.3.2.2 Termination Analysis

Showing the termination of **Algorithm 3** appears to be highly non-trivial, as many k -order Voronoi cells are concerning a certain node, and the dominating region of a node is mostly probably a non-convex region. Fortunately, the termination can be shown by focusing on the boundary of a dominating region.

Proposition 10. *Algorithm 3 terminates.*

Proof. As shown in Fig. 4.5, u_i^l and $\mathcal{V}_{n_i}^{k,l}$ are the location and dominating region of node n_i at the beginning of the l -th round, respectively. We also denote by c_i^l and R_i^l the Chebyshev center and the circumradius, respectively, of $\mathcal{V}_{n_i}^{k,l}$ computed by n_i during the l -round. Let $\hat{R}_i^l = \max_{u \in \mathcal{V}_{n_i}^{k,l}} \|u - u_i^l\|_2$ be the farthest distance from u_i^l in $\mathcal{V}_{n_i}^{k,l}$. Finally, we define $R^l = \max_i \{R_i^l\}$ and $\hat{R}^l = \max_i \{\hat{R}_i^l\}$.

We first prove the termination for $\alpha = 1$ by contradiction. For each n_i , we put a disk $\odot(c_i^l, R^l)$ centered at c_i^l with radius R^l . Obviously, $\bigcup_{i=1}^N \odot(c_i^l, R^l)$ form a k -coverage over the targeted area $\mathcal{A}$. The termination is naturally justified if we can prove that $\mathcal{V}_{n_i}^{k,l+1}$ is inside $\odot(c_i^l, R^l)$ after u_i^l is updated to c_i^l (i.e., $u_i^{l+1} = c_i^l$). For each point q on the boundary of $\mathcal{V}_{n_i}^{k,l+1}$, it is straightforward to see that c_i^l is the location of the k -th nearest nodes. Assume that q is outside $\odot(c_i^l, R^l)$, there would be only $k - 1$ disks

covering q , which contradicts the fact that $\bigcup_{i=1}^N \odot(c_i^l, R^l)$ constitute a k -coverage over $\mathcal{A}$.

Figure 4.5: Notations in the proof for Proposition 10.

We then prove the termination for $0 < \alpha < 1$. According to line 7 of **Algorithm 3**, during the l -th round, $u_i^{l+1} = u_i^l + \alpha(c_i^l - u_i^l)$. We put disks $\odot(u_i^l, \hat{R}_i^l)$ and $\odot(c_i^l, \hat{R}_i^l)$ centered at u_i^l and c_i^l , respectively. Obviously, $\mathcal{V}_{n_i}^{k,l}$ is inside $\odot(u_i^l, \hat{R}_i^l) \cap \odot(c_i^l, \hat{R}_i^l)$, which implies $\mathcal{V}_{n_i}^{k,l} \subset \odot(u_i^{l+1}, \hat{R}_i^l)$. Hence, $\bigcup_{i=1}^N \odot(u_i^{l+1}, \hat{R}_i^l)$ constitute a k -coverage over $\mathcal{A}$. Following a similar argument as for $\alpha = 1$, we have $\mathcal{V}_{n_i}^{k,l+1}$ is involved in $\odot(u_i^{l+1}, \hat{R}_i^l)$, which completes the proof. $\square$

In summary, our APOLLO algorithm terminates for any $\alpha \in (0, 1]$. Usually, smaller α leads to slower convergence but smoother moving locus. As a byproduct of the proof, we also conclude that $\hat{R}$ is non-increasing iteratively and finally equivalent to R . Especially, when $\alpha = 1$, R is also non-increasing in the iterative process. Therefore,

Corollary 1. *Algorithm 3 terminates at a local minimum of k -CSDP.*

It is important to note that R^l and $\hat{R}_i^l$ are introduced only for our proof. During the algorithm execution, each node n_i can only compute its own R_i^l and $\hat{R}_i^l$. According to our earlier discussion in Sec. 4.3.2.1 (see Fig. 4.4 also), the evolution of the node positions often takes two phases: an *expanding* phase and a *converging* phase. The expanding phase exists if the initial node distribution is non-uniform, our APOLLO algorithm will force the node to spread out during this phase, as discussed at the end of Sec. 4.3.2.1. During this phase, both R^l and $\hat{R}_i^l$ are most probably achieved by

boundary nodes. The expanding phase ends when all the boundary nodes are at the boundary of $\mathcal{A}$, this is when the converging phase starts.

Difficulty in characterizing the gap between local and global minima is a general issue with any solution techniques for handling non-convex optimization problems. To the best of our knowledge, algorithms for finding optimal k -coverage or for providing guaranteed approximation ratio to the optimal solutions given $k > 2$ do not exist even with centralized computations. Therefore, we can only rely on (later) extensive simulation results to demonstrate the optimality of APOLLO.

4.3.3 Discussions

In this section, we show the relations between the output of our APOLLO algorithm and other optimization objectives often considered for area coverage problems in WSNs, as well as some desirable properties of APOLLO.

Min-Node k -Coverage One type of problem that is commonly tackled in the research community is to achieve k -coverage using a minimum number of nodes (e.g., [2, 45, 48]). As this problem often assumes that all nodes have a fixed and identical sensing range r_s , it appears that APOLLO may not suggest a direct solution to it. However, we can transform our algorithm to deliver a good approximation to this min-node k -coverage problem as follows. APOLLO is called iteratively¹ and R^* is compared with r_s at the end of each iteration. Nodes are added (resp. reduced) if $R^* > r_s$ (resp. $R^* < r_s$), until $R^* \leq r_s$ but adding one more node would make $R^* > r_s$. Although the solution may not be optimal, it yields very good approximation to the optimal solution, as we will demonstrate in Sec. 4.5. In fact, as the up-to-date algorithms are all approximations for $k > 2$ and they are not autonomous (e.g., [2]), our algorithm is also the first autonomous deployment for approximating min-node k -coverage with an arbitrary k .

Maximum k -Coverage Another type of problem aims at maximizing coverage under fixed sensing ranges, but existing proposals only focus on 1-coverage [8, 9, 49]. A natural definition of the general maximum k coverage problem is to maximize the area that is

---

¹If an application only requires a one-time (rather than autonomous) deployment, we may use APOLLO in a centralized fashion.

k -covered under a fixed sensing range. The major difference between $k = 1$ and $k > 1$ is that the former achieves maximum coverage if nodes are far apart from each other whereas the same principle does not apply to the latter. An obvious example is the following: assume only 3 nodes are used to 3-cover an area, the maximum coverage is achieved only if all three nodes are put at the same location. Consequently, the heuristic of bounding the minimum separation among nodes [46] fails. Intuitively, APOLLO may deliver a good approximation to the maximum k -coverage problem, as, for example, APOLLO terminates at the optimal solution for the aforementioned 3-node case.

Connectivity Although maintaining network connectivity is not our concern in designing APOLLO, it appears to be a natural outcome of achieving k -coverage for $k \geq 2$. Under k -coverage, it is easy to see that there should be at least k nodes in the sensing range r_i of a node n_i (including n_i), otherwise u_i is not k -covered. In reality, as shown in Fig. 4.3, there are at least 7 nodes in a certain sensing range for $k \geq 2$. Given the common assumption in the literature that $\gamma \geq r_i$ (e.g., [8, 9, 35]), each node in a WSN has at least a degree of 6, which is sufficient to guarantee network connectivity.

Min-Max Fair While our k -CSDP only requires that the maximum sensing range is minimized and hence does not concern nodes with non-maximum sensing ranges, the min-max fair utility (a Pareto optimal point) requires that a node n_i cannot further reduce its sensing ranges r_i without increasing the sensing range r_j ($r_j \geq r_i$) of another node n_j . According to the property of k -order Voronoi diagram, the output of APOLLO is at least locally optimal with respect to the min-max fair utility, i.e., if we reduce r_i , another node n_j that shares dominating region boundary with n_i should increase r_j to maintain k -coverage, but we know $r_j \geq r_i$ before r_i gets reduced. In fact, our simulation results in Sec. 4.5 show that, after APOLLO terminates, the maximum and minimum sensing ranges are almost the same for $k > 2$.

4.4 APOLLO on 3D Surfaces

Though replacing Euclidean metric by geodesic distance yields a straightforward extension of our problem from 2D planes to 3D surfaces (as we discussed in Sec. 4.2.4), our APOLLO algorithm has to be slightly tuned to adapt to the local coordinate maps

(i.e., the log/exp maps). As APOLLO (**Algorithm 3**) involves two main computations: dominating region and Chebyshev center, we present the APOLLO 3D extension with respect to these two separately.

4.4.1 Computing Dominating Regions

By redefining the k -order Voronoi diagram based on geodesic distance. **Algorithm 4** could be extended to handle computations on 3D surfaces, while still guaranteeing the locality of the computations. After constructing a local coordinate system on the 3D surface (see Sec. 4.2.4.2), each sensor node n_i expands its searching ring ρ with a granularity of transmission range γ , until the geodesic disk $\odot_i = \{v \in \mathcal{M} | g(u_i, v) \leq \rho/2\}$ is not dominated by n_i anymore. Sensor node n_i needs only to communicate with $\mathcal{N}(n_i, \rho) = \{n_j | g(u_i, u_j) \leq \rho\}$ if its dominating region lies in $\odot_i$, according to **Lemma 1**. This extension is pretty straightforward; the only difference is that, while we compute $\|u_i - u_j\|_2$ on 2D planes, we use the ICH algorithm to obtain $g(u_i, u_j)$ on 3D surfaces.

A seeming discrepancy here is that, while the sensing range is mostly determined by Euclidean metric, APOLLO operates on geodesic distance. Fortunately, it can be shown in [91] that $1 \leq \frac{g(u_i, v)}{\|u_i - v\|_2} \leq \beta$, where β is a constant determined by the geometric properties of a 3D surface (i.e., its maximum Gaussian curvature). Consequently, the sensor nodes simply assign the geodesic distance $g(u_i, v)$ (the upper bound of the Euclidean distance) to their Euclidean sensing ranges (i.e., $r_i = g(u_i, v)$). This leads to a feasible solution that does not compromise much of the optimality.

4.4.2 Computing Chebyshev Centers

After determining the dominating region $\mathcal{V}_{n_i}^k \in \mathcal{M}$, the next step is for n_i to compute the Chebyshev center of its dominating region. The problem is reduced to that, given a set of points (i.e., the vertices of $\mathcal{V}_{n_i}^k$ in our case) on a surface, how to compute the Chebyshev center of the point set. Unfortunately, compared with its 2D counterpart, computing Chebyshev centers on 3D surface (i.e., under geodesic distance) appears to be highly non-trivial; it has not been addressed in the literature to the best of our knowledge. What we propose here is an iterative algorithm to calculate an approximate Chebyshev center $c'_i \in \mathcal{M}$ of $\mathcal{V}_{n_i}^k$ using log/exp map. We present the pseudo-codes

in **Algorithm 5**: the basic idea is to first compute the mass center of the dominating region by iteratively applying log/exp map (lines 1–6), and then determine the Chebyshev center (line 7–8).

Algorithm 5: Approximating a Chebyshev Center on a 3D Surface

---

Input: For each $n_i \in \mathcal{N}$, a local geodesic coord. system on the surface, the dominating region $\mathcal{V}_{n_i}^k$, stopping tolerance ε

Output: c'_i

- 1 Initialize the mass center of $\mathcal{V}_{n_i}^k$ as $\omega_i \leftarrow u_i$
  - 2 **repeat**
  - 3   Compute the logarithm map, $\exp_{\omega_i}^{-1}(\mathcal{V}_{n_i}^k) \in \mathcal{T}_{\omega_i}$, of $\mathcal{V}_{n_i}^k$ at ω_i
  - 4   Compute the mass center $\tilde{\omega}_i \in \mathcal{T}_{\omega_i}$ of $\exp_{\omega_i}^{-1}(\mathcal{V}_{n_i}^k)$
  - 5   **if** $\|\tilde{\omega}_i\|_2 > \epsilon$ **then** $\Delta\omega_i \leftarrow \delta [\exp_{\omega_i}(\tilde{\omega}_i) - \omega_i]$;
  - 6 **until** $\|\tilde{\omega}_i\|_2 \leq \epsilon$;
  - 7 $\tilde{\mathcal{V}}_{n_i}^k \leftarrow \exp_{\omega_i}^{-1}(\mathcal{V}_{n_i}^k)$; compute the Chebyshev center, $\tilde{c}_i$, of $\tilde{\mathcal{V}}_{n_i}^k$ in the 2D tangent plane $\mathcal{T}_{\omega_i}$
  - 8 $c'_i \leftarrow \exp_{\omega_i}(\tilde{c}_i)$
-

As the difficulty in computing the Chebyshev center is the local shape distortion resulting from any 3D-to-2D map, we want to find a log/exp map that yields the smallest distortion within $\mathcal{V}_{n_i}^k$. Intuitively, the mass center of $\mathcal{V}_{n_i}^k$,

$$\omega_i = \arg \min_u \int_{\mathcal{V}_{n_i}^k} g^2(u, v) dv$$

may yield a log/exp map that has the smallest shape distortion. Therefore, we take $\mathcal{V}_{n_i}^k$ to the tangent plane $\mathcal{T}_{\omega_i}$ by applying the logarithm map to $\mathcal{V}_{n_i}^k$ at ω_i , and we compute the Chebyshev center of $\exp_{\omega_i}^{-1}(\mathcal{V}_{n_i}^k)$ (line 7), and we finally determine the Chebyshev center of $\mathcal{V}_{n_i}^k$ using the exponential map (line 8). As the shape distortion has been suppressed as far as possible, we believe that c'_i is a good approximation to the real Chebyshev center of $\mathcal{V}_{n_i}^k$. Here $\delta [\exp_{\omega_i}(\tilde{\omega}_i) - \omega_i]$ in line 5 is computed with respect to a geodesic from $\exp_{\omega_i}(\tilde{\omega}_i)$ to ω_i on the 3D surface.

The question here is whether replacing $\{c_i\}$ by $\{c'_i\}$ in **Algorithm 3** still allows the algorithm to terminate. Fortunately, we have the following result:

Proposition 11. *Algorithm 3 terminates with $\{c'_i\}$ if the step size α is sufficiently small.*

Proof. Let $\mathcal{V}$ on $\mathcal{M}$ be contained in a geodesic ball $B(y, r)$ centered at $y \in \mathcal{M}$ with radius $r < \frac{\pi}{2 \max\{0, \kappa\}}$, where κ is the maximal Gaussian curvature of points inside $B(y, r)$. It is proven in [92] that the function $\Phi(\omega) = \int_{\mathcal{V}} g^2(\omega, v) dv$ for $\omega \in \mathcal{V}$ is convex and achieves a unique minimum $\omega^* \in B(y, r)$. A simple computation shows that $0 = \nabla \Phi(\omega^*) = 2 \int_{\mathcal{V}} \exp_{\omega^*}^{-1}(v) dv$. In other words, the mass center of $\mathcal{V}$ is uniquely defined and independent of the initial value, hence the iterative computation (lines 1-6) converges to the mass center. With the exponential map at the mass center ω^* , a tangent plane $\mathcal{T}_{\omega^*}$ is constructed and the Chebyshev center $\tilde{c}$ is computed on that plane. This ends one round for **Algorithm 3**. Suppose we take a sufficiently small step size α , the next round for **Algorithm 3** will be done on almost the same tangent plane $\mathcal{T}_{\omega^*}$. Therefore, from a node i 's point of view, the computations involved in two consecutive rounds l and $l+1$ are done in 2D. So we can apply the proving method for **Proposition 10** to show $\tilde{\mathcal{V}}_{n_i}^{k, l+1} \subset \tilde{\odot}(c_i^l, R^l)$. As the log/exp map is a bijection within $B(y, r)$, we have $\mathcal{V}_{n_i}^{k, l+1} \subset \odot(c_i^l, R^l)$, which completes the proof. $\square$

The computations of **Algorithm 5** are done by individual nodes; they entail neither communications nor motions. Since $\Phi(\omega)$ has C^2 smoothness, the algorithm converges very fast, causing a negligible computational cost. In reality, $\alpha < 1$ almost always guarantees the overall convergence.

4.5 Simulations

In this section, we report our simulation results. We first present the convergence of APOLLO. Then, we study the energy consumptions during and after the autonomous deployments. We also evaluate the output of APOLLO in *Min-Node k-Coverage* and *Maximum k-Coverage*, followed by the adaptability to network irregularities. Finally, we validate the effectiveness of APOLLO 3D extension by deploying sensor nodes on three 3D terrains.

4.5.1 Convergence

As convergence results we obtain from our extensive experiments are all similar, we present only two case to demonstrate the convergence of our algorithm. We consider a targeted area of 1 km², and initially deploy 100 sensor nodes either at the bottom-left corner, shown by Fig. 4.6(a), or separated into two disjoint groups located at the bottom-left and upper-right corners, shown by Fig. 4.6(f). According to the following

four sub-figures for both cases, it is obvious that our algorithm leads to an “even” node distribution in the sense of multiple coverage. Specifically, in the multiple coverage cases with $k = 2, 3, 4$, nodes tend to cluster in groups of size k , in contrast to the pure even distribution for $k = 1$. This is not a surprise as such an “even clustering” distribution yields more overlaps of the dominating regions among every cluster, which in turn reduces the required sensing range. Interestingly, this appears to also meet the needs of maximum k -coverage. As we discussed in Sec. 4.3.3, APOLLO leads to a co-location deployment for the extreme example of using three nodes to achieve 3-coverage. The second case also shows that two disconnected clusters will eventually merge into a concocted network. According to our experience with extensive simulations, the initial deployment does not have significant impact on the algorithm output.

Figure 4.6: Initial deployments and k -coverage ($k = 1, 2, 3, 4$) deployments as the output of APOLLO.

We also show the convergence process of APOLLO in Fig. 4.7. Since a sensor node finally reaches the Chebyshev center of its dominating region and the sensing range is equivalent to the circumradius of the dominating region, we illustrate the relationship between execution rounds (of length τ ms each) and the maximum/minimum

Figure 4.7: The convergence of APOLLO.

circumradii. As the nodes are deployed at the corner of the targeted area initially, the maximum circumcircle usually appears on the network boundary, which is mainly determined by the searching ring (Fig. 4.4). Consequently, the maximum circumradii for $k = 1, 2, 3, 4$ are almost the same at the beginning. Corresponding to our proof of termination, the maximum circumradius is monotonically decreasing with the execution rounds of APOLLO, while the minimum radius is increasing in general. In the end, the maximum and minimum radii are very close to each other, especially for larger k . While the monotonic decreasing in maximum circumradius shows the termination of APOLLO, the fact that the minimum circumradius coincides with the maximum one further confirms the balanced sensing load in the deployed WSN.

4.5.2 Energy Consumption during Deployments

While the convergence results reported in Sec. 4.5.1 are only about the performance of APOLLO as an algorithm, the actually cost of autonomous deployments conducted by APOLLO is actually more complicated. In this section, we use TOSSIM [78] simulations driven by realistic power consumption data to evaluate the energy consumption of the whole deployment process. We assume that a mobile sensor node is equipped with a Micromo coreless DC motor (www.micromo.com/coreless-dc-motors-data-sheets.aspx). The power consumption of this motor is 120 mW and it may move a MicaZ Mote in a speed of 0.2 m/s. For communication cost, we get the data from the specification of the popular CC2420 radio [10]: transmit power 52.5 mW and receiving

(or idle-listening) power 56.4 mW. We assume that, during the i -th round (see **Algorithm 3**), the radios are shut down (zero power consumption) when nodes move, and the communication (and computation) session starts only when nodes have moved to their new locations $\{u_i^+\}$ (see **Algorithm 3**).

Based on the same scenario studied in Sec. 4.5.1 (i.e., 100 nodes on 1 km² area), Fig. 4.8 demonstrates the actual energy consumption of six autonomous deployments. It is evident that a smaller step size α results in more rounds but shorter total moving distances; this is shown by a decreasing communication consumption in Fig. 4.8(a) and an increasing motion consumption in Fig. 4.8(b) as functions of α . Therefore, given certain power consumption specifications for motion and communication, we may tune α to obtain an energy efficient deployment. For our current settings, the best step size shown by Fig. 4.8(c) is around 0.3 to 0.7. We also pick up nodes that consume the highest energy in each WSN to illustrate the energy consumed by individual nodes in Fig. 4.8(d). To put these consumptions into perspective, a 2450mAh Energizer (www.energizer.com) AA battery contains 13kJ, so the (maximum) individual node consumption of 200J only accounts from a small part of the node's energy storage.

We also report the time cost APOLLO in Fig. 4.9. As the time cost stems from both communication and motion, the general trend is similar to the total combined consumption shown by Fig 4.8(c): the time cost is minimized around $\alpha \in [0.3, 0.7]$ as well, for which APOLLO terminates within around 25 minutes and is reasonable for practical applications.

In order to give a better illustration on the effect of adjusting α , we also use a 20-node case to show the moving loci of nodes in a deployment in Fig. 4.10. Obviously, a large α often results in many detours, while a small α can make each moving locus almost straight (or optimal).

4.5.3 Energy Consumption after Deployments

In this section, we show the sensing energy consumption after APOLLO completes the deployments. We again consider a targeted area of 1 km², while scaling the network size (i.e., the number of sensor nodes) from 20 to 180. As the sensing range is modeled as a disk centered at u_i with radius r_i , we naturally define the energy consumption

Figure 4.8: Energy consumption during the autonomous deployments for 100 nodes

function as $E(r_i) = \pi r_i^2$: an increasing function of r_i ¹.

We illustrate the sensing energy consumption in terms of maximum load $\max\{E(r_i)\}$ and total load $\sum_{i=1}^N E(r_i)$ in Fig. 4.11. As we deploy more sensor nodes to a given targeted area, each node takes care of less area when achieving a certain coverage. The maximum sensing load is thus decreasing with the increasing number of nodes. Given a certain number of nodes, to achieve higher coverage degree, each sensor node is supposed to cover larger area thereby enhancing the maximum sensing load. We also observe that for k_1 -coverage and k_2 -coverage, the ratio of maximum loads between

¹ As data processing and memory accessing consume most of power in sensing and their frequency depends on the covered area, we model the energy consumption in sensing to be proportional to the area of the sensing disk.

Figure 4.9: Total time cost.

them is roughly k_1/k_2 , which can be explained as follows. Since APOLLO makes the minimum sensing range very close to the maximum one, each sensor node roughly covers the same area $k|\mathcal{A}|/N$, i.e., $E(r_i) = k|\mathcal{A}|/N$ where $|\mathcal{A}|$ is the area of the targeted region. Nevertheless, increasing the number of nodes does decrease the total sensing load of the WSN, shown by Fig. 4.11(b). Because using a less number of nodes requires each sensor node to cover a larger area, which implies a larger sensing disk. This in turn yields more overlap between sensing ranges (i.e., a larger sensing redundancy), thus a higher total load.

4.5.4 Comparisons with Min-Node k -Coverage

As mentioned in Sec. 4.3.3, our APOLLO algorithm results in a good approximation to min-node k -coverage problem (where all nodes have the same sensing range and the objective is to minimize the number of nodes used to achieve k -coverage). In this section, we compare our algorithm with the deployment strategies proposed in [45] and [2], in terms of the required number of nodes guaranteeing k -coverage ($k \geq 2$). As we can increase the minimum sensing range to the maximum one in the output of APOLLO without compromising coverage, we assign an identical sensing range to every node as the maximum sensing range R^* in our case.

Bai *et al.* have proven in [45] that, without considering boundary effect and with an

Figure 4.10: Moving loci of a 20-node deployment with APOLLO for 3-coverage. Starting locations are blue and ending locations are red.

identical sensing range r , the optimal **congruent** deployment density¹ for 2-coverage is $4\pi/3\sqrt{3}$. Given a targeted area $\mathcal{A}$, we thus compute the minimum number of sensor nodes $N_{k=2}^*$ for 2-coverage as:

$$N_{k=2}^* = \frac{|\mathcal{A}| \frac{4\pi}{3\sqrt{3}}}{\pi r^2} = \frac{4|\mathcal{A}|}{3\sqrt{3}R^{*2}},$$

here we use $|\mathcal{A}|$ to replace the area of Voronoi polygons generated by sensor nodes, which leads to an under-estimation of $N_{k=2}^*$ due to the boundary effect. We simulate large-scale WSNs with size ranging from 1000 to 1600 in a 1 km^2 targeted area. The result is shown in Table 4.1. In general, the number of nodes deployed by APOLLO is about 15% higher than the minimum value, and it is obvious that the boundary effect is the main reason for this difference. As the boundary effect is not considered in [45], extra nodes are needed to cover the vacancies on the boundary due to the mismatch between the congruent deployment and an arbitrarily shaped targeted area. Therefore, we conclude that APOLLO actually leads to a very good approximation of the min-node 2-coverage problem.

¹Deployment density is defined as a ratio of the area of sensing disks to the area of Voronoi polygons generated by sensor nodes [45].

Figure 4.11: Energy consumption in the final deployments of 100 nodes

Table 4.1: The minimum number of sensor nodes to achieve 2-coverage

N	1000	1200	1400	1600
R^* (m)	30.35	27.12	25.23	23.57
$N_{k=2}^*$	836	1047	1210	1386

In [2], Ammari *et al.* propose to decompose a targeted area into adjacent *Reuleaux triangles*, and nodes are deployed in the intersection areas between these triangles (these intersections are termed lens in [2]). According to their derivation, $\frac{6k|\mathcal{A}|}{(4\pi-3\sqrt{3})r^2}$ nodes are required to k -cover $\mathcal{A}$ where $k \geq 3$ and r is the sensing range. Here we compare this feasible deployment with APOLLO. We deploy 180 nodes in a 1 km² area and let all nodes have the same sensing range R_k^* . We also compute the number of nodes that deployed according to the strategy proposed in [2] as $N_k^* = \frac{6k}{(4\pi-3\sqrt{3})R_k^{*2}}$. From the results shown in Table 4.2, it is very clear that APOLLO can use much less nodes to achieve the same level of coverage compared with [2].

4.5.5 Performance in Maximum k -Coverage

In addition to the analysis in Sec. 4.3.3, we evaluate the performance of APOLLO in solving maximum k -coverage problem. Due to space limitation, we only illustrate the results of 4-coverage. We assume sensor nodes have fixed sensing ranges of 135m and

Figure 4.12: Adaptability of APOLLO to arbitrarily shaped areas and obstacles as well.

are deployed in a square targeted region of 1 km^2 . We also vary the network scale from 80 to 110 with a step of 10. The 4-coverage ratio (i.e., the ratio between the 4-covered area and the whole targeted region) in each round is demonstrated in Fig. 4.14. It is shown the area 4-covered by the sensor nodes is increased with the execution of APOLLO, and reaches maximum when APOLLO converges. Since the sensing ranges of the sensor nodes are fixed, a network with small scale may not be able to fully 4-cover the targeted region, e.g., 80, 90 or 100 nodes in our case. By gradually deploying more sensor nodes (e.g., 110 nodes), we can have the whole targeted area 4-covered. Recall that $N = 110$ sensor nodes with a fixed sensing range of $r = 135\text{m}$ can 4-cover (hence $k = 4$) an area of up to $\frac{N(4\pi - \sqrt{3})r^2}{6k} = 0.9\text{km}^2$. Therefore, we believe that, APOLLO provides a good approximation to the maximum k -coverage problem.

4.5.6 Adaptability to Obstacles

Being an autonomous deployment algorithm, APOLLO should be able to adapt to arbitrarily shaped areas (an advantage of autonomous deployments over regular deploy-

Figure 4.13: k -coverage deployments on 3D surfaces, $k = 2, 3, 4$.

Table 4.2: The number of sensor nodes to achieve k -coverage with the strategy proposed in [2] for $k = 3, 4, \dots, 8$

k	3	4	5	6	7	8
R_k^* (m)	87.6	102.1	112.4	123.6	133.9	143.2
N_k^*	318	313	323	320	318	318

ments). We hereby demonstrate the adaptability of APOLLO to arbitrarily shaped targeted area (with obstacles inside) in Fig. 4.12. The “holes” within the network region represent obstacles that mobile sensor nodes cannot move upon. It is clear that APOLLO adapts well to these irregularities and again achieves the even clustering distribution as if the area were regular.

4.5.7 Extension to 3D Surfaces

Finally, we apply the 3D APOLLO extension discussed in Sec. 4.4 for WSN deployments on various 3D terrain surface models. The three terrain models are approximated by 5k, 20k and 130k triangles, respectively. In Fig. 4.13, each row shows one terrain where

Figure 4.14: 4-coverage ratio.

we deploy the sensor nodes. Since a large-scale sensor network are usually air-dropped in the applications of terrain monitoring, we initially deploy 400, 400 and 800 mobile sensor nodes for these three terrains respectively in a random manner. Fig. 4.13 shows the outputs also reflect clustering distributions in the multiple coverage cases with $k = 2, 3, 4$. The similarity between Fig. 4.13 and Fig. 4.6 clearly demonstrates that our deployment algorithm designed for 2D deployments has been successfully extended to handle 3D surface deployments.

In Table 4.3, we use the maximum and minimum (Euclidean) sensing ranges resulted from the autonomous deployments to show the quality (in terms of load balancing) of the coverage. In order to make the numbers comparable to each other, we normalize the three surfaces such that their 2D projections are all on a 1 km^2 area.

Table 4.3: The maximum and minimum sensing ranges for three surface deployments

	Surface 1			Surface 2			Surface 3		
k	2	3	4	2	3	4	2	3	4
Max range	56.77	69.35	81.80	44.94	54.53	55.65	35.56	38.09	47.46
Min range	47.06	62.21	76.97	37.55	46.93	53.32	29.49	31.69	41.45

A direction interpretation, by comparing Table III with Fig. 4.7, the differences between the maximum sensing ranges and the minimum sensing ranges in 3D deployments are a little larger than the ones delivered by 2D deployments. In another word, quality of APOLLOs output in terms of load balancing is worse in 3D than in 2D. This

is expectable as the problem becomes significantly harder to handle on 3D surfaces. However, the results in Table III do not indicate a worse performance of APOLLO in terms of solving the k -coverage optimization problem, because it has never been proven that an optimal k -coverage solution (for $k > 2$) for either 2D planes or 3D surfaces can be or have to be achieved by disks with an identical radius. It is highly possible that, on 3D surfaces, an optimal k -coverage solution indeed accommodates variable radii. Another reason is that, we employ an approximated Chebyshev centers in 3D APOLLO which may lead to compromise in terms of load balancing.

4.6 Summary

In this chapter, we have focused on minimizing the maximum sensing range to achieve load balancing k -coverage through autonomous deployments (i.e., relying on mobile sensors nodes and the wireless communications among them). We have innovated in applying the k -order Voronoi diagram in a localized manner, and we have proposed APOLLO to solve the optimization problem through a distributed and localized procedure. Our approach significantly extends the existing mobility control methods for solving sensor coverage problem, as we are the first to tackle the problem of k -coverage autonomous deployment, for WSNs on both 2D planes and 3D surfaces. We have proven the termination of APOLLO as well as the (local) optimality of its output. We have also explained the close relations between the output of APOLLO and other commonly used optimization objectives, which provides a better understanding of optimal k -coverage deployments whose theoretical characterizations are hard to obtain under general settings. Finally, our simulation results strongly confirm all the theoretical claims we have made, and they also demonstrate the adaptability of APOLLO to the irregularities of the targeted sensing region, as well as validate the effectiveness of its 3D surface extension.

Chapter 5

ART: Adaptive fRequency-Temporal co-existing of ZigBee and WiFi Networks

A well designed network deployment may result in a massive amount of data. Collecting them from the WSNs exerts a heavy burden on the data transmission within networks. Moreover, recent large-scale deployments of wireless sensor networks have posed a high demand on network throughput, forcing all the (orthogonal) ZigBee channels to be fully exploited to enhance transmission parallelism. However, the interference from widely deployed WiFi networks has severely jeopardized the usability of these discrete ZigBee channels, while the existing CSMA-based ZigBee MAC is too conservative to utilize each channel temporally. In this chapter, we propose ART (*Adaptive fRequency-Temporal co-existing*) as a framework to improve the co-existence status between ZigBee and WiFi in both frequency and temporal manner. ART consists of two components: FAVOR (*Frequency Allocation for Versatile Occupancy of spectRum*) and P-CSMA (*Probabilistic CSMA*). On one hand, FAVOR allocates continuous (center) frequencies (thus partially overlapped channels instead of discrete orthogonal channels) to nodes/links in a near-optimal manner, by innovatively converting the problem into a spatial tessellation problem under a unified spatial-frequency space. This can be considered as a coverage problem where each sensor node “covers” not only a certain area but also a piece of frequency band. This allows ART to fully exploit the “frequency white space” left out by WiFi. On the other hand, ART employs P-CSMA to oppor-

tunistically tune the use of CSMA for leveraging the “temporal white space” of WiFi interference: P-CSMA is a randomized algorithm controlled by a real-time assessment of transmission quality. Moreover, ART can be carried out in a fully localized manner without global coordination, which significantly improve the adaptability of the large-scale WSNs to dynamic deployment scenario. We implement ART in MicaZ platforms, and our extensive experiments with different network settings strongly demonstrate the efficacy of ART in enhancing both transmission throughput and quality.

5.1 Introduction

Many recent WSN-based applications have posed increasing demands in network throughput mainly due to their large-scale network deployments [5, 6]; hence, the multi-channel capability of ZigBee radios (e.g., CC2420 of MicaZ Mote [10]) is intensively exploited (e.g., [11, 12]) to enable concurrent transmissions.

Unfortunately, multi-channel WSNs face the contentions from other networks that also operate in the 2.4 GHz ISM band, e.g., the pervasive deployments of WiFi hotspots [14, 15, 93]. On the one hand, as WiFi radios have a much higher power than ZigBee ones while their frequency bands heavily overlap (see Fig. 5.1), the WiFi interference may substantially confine the availability of the orthogonal channels specified in the ZigBee standard [13] and hence degrade transmission throughput and quality (measured by *Packet Reception Ratio*, or PRR). On the other hand, the most popular mechanism to resolve interference, *Carrier Sense Multiple Access* (CSMA), sacrifices throughput for improving PRR: it simply uses conservative temporal spacing to avoid transmission collisions. These observations have motivated us to re-visit the co-existence issue of ZigBee and WiFi, aiming to ensure both transmission throughput and quality for multi-channel WSNs.

To improve the performance of ZigBee networks under WiFi interference, one needs to maximize the utilization of limited spectrum resource. Inspired by the fact that interference attenuates with distance, our experiments demonstrate that continuously allocating center frequencies to nodes/links with respect to their distances would be more efficient, compared with the conventional graph coloring approach [94] where discrete channels are allocated based on the concept of discrete graph distance. For practical implementation, the ability of CC2420 radio in adjusting its channel center

Figure 5.1: We overlap the WiFi signal strength observed through inSSIDer in our research center with the 16 channels of ZigBee. Clearly, apart from channels 25 and 26, all other channels are subject to multiple WiFi interferences.

frequency at a granularity of 1 MHz [10] does offer a good approximation of frequency fine-tuning.

While better spectrum utilization method improves transmission parallelism and avoids interference, a collision avoidance mechanism is still necessary in case of inevitable co-channel interference. Our experiments demonstrate, enabling CSMA offers a high PRR but drastically reducing the throughput regardless of whether interference exists or not, whereas disabling CSMA gives a high throughput but incurring a low PRR in the face of severe interference. Given the temporal variations exhibited by WiFi interference [14], an adaptive MAC that opportunistically tunes the use of CSMA according to interference intensity could achieve both high throughput and high PRR.

Driven by the aforementioned two observations, we propose ART (*Adaptive fRequency-Temporal co-existing*) as a framework to improve the co-existence status between ZigBee and WiFi in a joint frequency-temporal manner. Essentially, ART has FAVOR (*Frequency Allocation for Versatile Occupancy of spectRum*) as a flexible multi-channel access mechanism, and it employs P-CSMA (*Probabilistic CSMA*) to temporally adapt the use of CSMA to the intensity of collision.

The major principle behind FAVOR is a paradigm shift from *channel allocation* to (center) *frequency allocation*. FAVOR transforms the *frequency allocation* problem into a *spatial tessellation* problem by unifying frequency and distance into one continuous space, and adapts CVT (*Centroidal Voronoi Tessellation*) method [50] to seek a near-

optimal solution that the nodes/links that are closer to each other in distance are further away from each other in frequency. Moreover, the density function used in CVT enables us to take as input the intensity of WiFi interference. Though FAVOR can be viewed as a continuous version of the conventional graph coloring approach, it offers greater freedom due to the relaxation from a discrete set to a continuous space.

Although FAVOR's flexible spectrum access enables ZigBee links to fit the "frequency white space" of WiFi spectrum, interference may not be totally avoidable. This is where P-CSMA works to opportunistically leverage the "temporal white space" of the time-varying interference. Based on real-time PRR measurements fed by a receiver to indicate the interference intensity, the randomized control of P-CSMA gives a better chance to disabling CSMA under a good link quality whereas it gets biased towards enabling CSMA under a severe interference. In a nutshell, P-CSMA enables ART to fine-tune the tradeoff between throughput and PRR under a time-varying interference that cannot be totally avoided by FAVOR.

In summary, we make the following main contributions in this chapter as follows.

- We propose to shift the paradigm from traditional discrete channel allocation to continuous frequency allocation.
- We innovate in transforming frequency allocation into spatial tessellation in a frequency-distance space, and propose an algorithm to compute a near-optimal solution in a localized manner.
- We design a probabilistic mechanism to adaptively use CSMA according to real-time interference assessments; it can fine-tune the tradeoff between transmission throughput and quality.
- We perform extensive experiments in order to demonstrate the efficacy of our proposal in tackling the ZigBee-WiFi co-existence issue.

The remaining of this chapter is organized as follows. We first report the experiments that motivate our design of ART in Sec. 5.2. We then present the details of our ART framework in Sec. 5.3. The evaluation of ART is given in Sec. 5.4. We finally conclude this chapter in Sec. 5.5.

5.2 Motivations and Background

In this section, we first report several experiments on MicaZ Motes. Note that the observations delivered by our experiments also exist for other nodes equipped with CC2420 radio, but we will not report results for other platforms due to the space limit. We also explain how the observations acquired from these experiments motivate our design of ART. We finally introduce the mathematical background on which our FAVOR algorithm will be based.

5.2.1 The Impact of CSMA on Throughput

It is documented in [10] that the nominal data rate of CC2420 is 250kbps, but [95] claims that it is only about 50kbps for TelosB mote. Therefore, we design the following experiments to seek the root of this difference. The experiments are done on a single link (consisting of a transmitter and a receiver) which is 3.6m long and operates in a “clean” frequency (e.g., 2480 MHz) to avoid WiFi interference. We let the transmitter send packets to the receiver successively as fast as possible. With CSMA enabled, the transmitter sends only one packet (45 bytes in total including 26 bytes payload and 19 bytes header) every 9ms using MicaZ motes, resulting in a data rate of 40kbps and equivalent to a throughput of 0.16^1 with respect to the nominal data rate. The reason for the difference between the measured data rate and the nominal one is that, the slow MCU (Atmel AVR Atmega 128L) employed by MicaZ mote spends a large amount of time on processing packets within the radio stack, and most prominently, performing CSMA. Hence, the radio interface always stays in an unsaturated state.

In order to obtain the real maximum data rate, we repeat the above experiment but disable CSMA. We then find that this link can carry a packet every 2ms (with an almost 100% PRR), resulting in a data rate of 180kbps and thus a throughput of 0.72. In other words, the data rate in this case is 4.5 times of that with CSMA enabled. This shows that CSMA does pose a heavy restriction on data rate (or throughput). We hereafter say a link achieving *full* data rate (resp. throughput) if it has a comparable data rate (resp. throughput) to a single (CSMA enabled or disabled) link that is free of any interference.

---

¹In this chapter, we measure the throughput as the ratio between *receiving data rate* at receiver and the *nominal data rate* of 250kbps.

To better understand the effect of CSMA coordination, we set up an experiment shown in Fig. 5.2: four MicaZ motes are used to set two parallel ZigBee links l_1 and l_2 . We fix the location of l_1 , and change the location of l_2 such that the distance between l_1 and l_2 varies from 1.2 m to 4.8 m. As we are interested in ZigBee interference for now, we operate the links in 2480 MHz to avoid WiFi interference. If letting the

Figure 5.2: Experiment configuration. The distance from transmitter to receiver is 3.6m.

transmitters of the two link send packets persistently as fast as possible in the same channel, one would get much lower (at least halved) data rate for each link due to their interference. However, Fig. 5.3 shows a very different outcome: both links achieve a

Figure 5.3: The throughputs and PRRs for two co-channel links with different distances.

nearly full throughput of 0.16 and a perfect PRR, even when they are very close to each other. The reason is the “procrastination” of CSMA as we explained earlier.

With a considerable sacrifice in throughput, CSMA can perfectly coordinate co-channel ZigBee link. Testing the performance (in terms of transmission throughput and quality) of a channel allocation mechanism on CSMA-enabled platforms leads to wrong conclusions.

5.2.2 Frequency-Distance Tradeoff without CSMA

In light of the results shown above, we then disable CSMA and test the two links (Fig. 5.2) with different frequency and distance combinations. Here we fix the frequency of l_1 at 2480 MHz, while changing the frequency of l_2 from 2475 to 2480 MHz. The throughputs and PRRs for all the frequency-distance combinations are shown in Fig. 5.4(a) and (b), respectively. The figures clearly show throughput and PRR trade-off surfaces that we may achieve by extending from a 1D discrete frequency space (for conventional channel allocations) to a higher dimensional *frequency-distance space*.

Figure 5.4: The lower throughput and PRR for two links with different frequency-distance combinations. CSMA is disabled for both links.

We also plot two sets of cut views of Fig. 5.4 in Fig. 5.5. In Fig. 5.5 (a) and (b), the distance is fixed at 1.2 m, and in Fig. 5.5 (c) and (d), the frequency of l_2 is fixed at 2479 MHz. As shown in Fig. 5.5 (a) and (b), when two consecutive frequencies (e.g., 2479 and 2480 MHz) are used, the resulting channel overlapping ruins one of the links: one achieves a nearly full throughput (beyond 0.7) and a 100% PRR, while the other one gets much lower throughput (but interestingly, still higher than the full rate of a single link

Figure 5.5: The throughputs and PRRs of both links as functions of frequency (a), (b) and distance (c), (d).

with CSMA enabled) and PRR. Under the same short distance, a frequency separation of 2 MHz (e.g., 2478 and 2480 MHz) is usable for achieving a superior throughput and PRR. Fig. 5.5 (c) and (b) further demonstrate, by increasing the distance between two links (e.g., 3.6m), even a frequency difference of 1 MHz is sufficient to achieve an almost full throughput and a perfect PRR. In other words, slightly increasing distance can resolve the interference resulting from channel overlapping.

Recall that ZigBee standard [13] support only two orthogonal channels in the spectrum 2475-2480 MHz, whereas we almost get six overlapped but highly usable channels by varying center frequency and link distance in a continuous manner.¹ Therefore,

¹If center frequency of CC2420 radio can be tuned at a finer granularity (e.g., of less than 1 MHz) in the future, our mechanism shall offer more available channels.

another important message we get is this:

Continuously allocating frequencies to links with respect to the distances between them allows for a much more flexible and efficient use of limited spectrum, which brings great potential to improve network throughput and to guarantee transmission quality.

One further implication is that, if we implement a multi-channel access mechanism based on the aforementioned continuous frequency tuning for a pure ZigBee network, we may totally disable CSMA and hence allow for a significant improvement on throughput while still preserving PRR. This is indeed what will be achieved by our FAVOR algorithm (Sec. 5.3.2). Now the question is how the performance is affected by co-existing WiFi networks, which we will further investigate in Sec. 5.2.3.

5.2.3 WiFi Interference to ZigBee under Continuous Spectrum

We perform our experiments on a single ZigBee link in our research center where many WiFi networks exist (Fig. 5.1). We continuously tune its center frequency to scan the whole WiFi spectrum (i.e., 2405-2474 MHz) instead of only the 16 standard ZigBee channels. To verify the temporal variation of WiFi interference, we measure throughput and PRR every one hour during the 8 working hours in a day. In order to check the coordination effect of CSMA under WiFi interference, we try both cases with CSMA enabled and disabled.

Obviously, WiFi interference mainly comes from three orthogonal 802.11 channels (i.e., channels 1, 6 and 11), which is exactly shown by the waveform surfaces in Fig. 5.6. As expected, since CSMA provides coordination to deal with interference but restricting data rate, enabling CSMA results in stable transmissions (with a rather high PRR) but a low throughput, as shown by Fig. 5.6 (a) and (b). Whereas disabling CSMA may significantly improve the throughput (even under severe WiFi interference), the transmission quality (in terms of PRR) can go down to 0.5 in the worst case, as shown by Fig. 5.6 (c) and (d). Obviously, there is no need for CSMA within the frequency white space (e.g., around 2425, 2450 and 2474 MHz) where WiFi interference is weak or absent, but CSMA is necessary to preserve a reasonable PRR in the severely interfered frequencies (e.g., around 2412, 2437 and 2474 MHz). In summary, we have the following insight:

Figure 5.6: The impact of CSMA on throughput and PRR in different frequencies under WiFi interference.

Neither enabling nor disabling CSMA adapts well to WiFi interference. The use of CSMA should be tuned in a more flexible fashion according to the transmission quality in a specific frequency band.

5.2.4 Adapting to the Temporal White Space of WiFi Interference

The stark contrast (in performance) between disabling and enabling CSMA implies that sticking solely to one of them would fix the network performance to one extreme (either high loss or low throughput). To this end, we study the possibility of a randomized

control on disabling/enabling CSMA. The basic idea is, for every packet to be sent, a coin is tossed based on a certain probability (namely *CSMA Probability*) to decide whether to enable CSMA or not. In this experiment, we operate a single ZigBee link under two interfered frequencies (2455 and 2460 MHz) and a clean frequency (2480 MHz), respectively.

Fig. 5.7(a) shows that for all the three frequencies, the throughput drops with an increasing CSMA probability, consistent with our observation in Sec. 5.2.1. The

Figure 5.7: Throughputs and PRRs under different CSMA probabilities.

throughputs for 2455 and 2460 MHz are lower than the one for 2480 MHz due to the ambient WiFi interference. An opposite trend can be observed in Fig. 5.7(b), where increasing the CSMA probability may significantly improve PRR for the two interfered frequencies. Note that PRR remains to be 1 for the clean frequency, which corroborates our finding in Sec. 5.2.3 that CSMA should be disabled for clean frequencies. Our results in Fig. 5.7 demonstrate that a randomized control does allow for a smooth fine-tuning of the tradeoff between throughput and PRR. For example, in the 2455 MHz channel, a CSMA probability of 0.3 doubles the throughput with respect to the totally enabled CSMA (i.e., CSMA probability is 1), while achieving a PRR of 0.9 (very close to the totally enabled CSMA). If we afford a PRR of 0.85, we can take a CSMA probability of 0.1 and hence increase the throughput by a factor of three compared with the totally enabled CSMA. So the main insight is:

Adding a randomized control to determine whether to enable CSMA or not on a per packet basis endows us with the leverage to fine-tune the tradeoff between throughput and PRR.

5.2.5 Mathematical Background on Spatial Tessellation and CVT

As a theoretical basis of our later algorithm, we hereby briefly introduce a few key ideas on spatial tessellation. Given a region $\mathcal{A} \subseteq \mathbb{R}^n$, the set $\{A_i\}$ is called a *tessellation* of $\mathcal{A}$, if $A_i \cap A_j = \emptyset$ for $i \neq j$ and $\cup_i A_i = \mathcal{A}$. Let $\|\cdot\|_{\ell^2}$ denote the Euclidean norm on $\mathbb{R}^n$. For a set of points $\{u_i\}$ belonging to $\mathbb{R}^n$, the Voronoi region V_i corresponding to the point u_i is defined by

$$V_i = \{v \in \mathcal{A} \mid \|v - u_i\|_{\ell^2}^2 \leq \|v - u_j\|_{\ell^2}^2, \forall j \neq i\}.$$

The set $\{V_i\}$ is termed *Voronoi tessellation* of $\mathcal{A}$, with points $\{u_i\}$ called *generators* and each V_i referred to as the *Voronoi cell* corresponding to u_i . An energy function is defined upon each generator according to its Voronoi cell. One typical metric of measuring such an energy is $\|v - u_i\|_{\ell^2}$ where $v \in V_i$, hence generator u_i has an energy $\int_{V_i} \|v - u_i\|_{\ell^2}^2 \Phi(v) dv$ where $\Phi(v)$ denotes the density function at v . It is obvious that the optimal location for node n_i to minimize its energy is the centroid of V_i . Moreover, a generator can impact other nearby generators. In particular, if a generator has high energy, i.e., deviating from the centroid and getting close to a boundary edge of its Voronoi cell, it exerts an impact to another generator whose Voronoi cell shares this boundary edge. Such mutual impact can be characterized by the dual graph of the Voronoi tessellation, formed by using edges to connect the pairs of the generators whose associated Voronoi cells are adjacent. The generators connected by shorter edges in the dual graph have larger mutual impact.

The aim of *Centroidal Voronoi Tessellation* (CVT) is to calculate an optimal tessellation by varying the locations of the generators (and thus their Voronoi tessellation) such that their total energy (Equation (5.1)) is minimized .

$$\mathcal{J}(\{V_i\}, \{u_i\}) = \sum_i \int_{V_i} \|v - u_i\|_{\ell^2}^2 \Phi(v) dv. \quad (5.1)$$

As this objective is not convex, achieving a global optimality is hard. Fortunately, the CVT theory [50] indicates a good locally optimal solution that every u_i coincides

with the centroid of its Voronoi cell V_i . Consequently, the generators in CVT are evenly spread in the space if the density function is constant and have almost the same energy. In other words, CVT leads to an energy-balanced tessellation. From the dual point of view, minimizing the total energy of the generators is equivalent to minimizing their mutual impact. Moreover, the distribution of the generators well adapts to a non-constant density function $\Phi(v)$: a larger function value leads to a higher generator density. These unique properties of CVT are related to our need for “spreading” sensor nodes (or links) over the frequency spectrum with their “mutual impacts” (i.e., interference) minimized. In fact, our FAVOR algorithm is a non-trivial extension of CVT to a space involving both frequency and distance (or location).

One classic approach of calculating CVT is Lloyd’s method [96], which iteratively computes the Voronoi tessellation and moves the generators to the centroids of their associated Voronoi cells. However, when the input is in a high-dimensional space (e.g., $\mathbb{R}^n$ for $n > 2$), computing Voronoi tessellation turns out to be highly non-trivial. A very complicated data structure is required to model a $\mathbb{R}^3$ Voronoi tessellation, and the induced computational cost is too high to be affordable for hardware platforms with limited energy and computing capability (e.g., MicaZ motes). For $\mathbb{R}^n$ ($n > 3$), no algorithm for CVT has been implemented even for common CPUs.

5.3 ART: Co-Existing in Frequency and Time

In this section, we first discuss our system model. Then we present in detail our FAVOR algorithm and P-CSMA protocol, respectively. We finally show how ART can be implemented in a practice scenario.

5.3.1 System Model

We consider a WSN consisting of a set of ZigBee nodes $\mathcal{N} = \{n_1, n_2, \dots, n_N\}$ with $|\mathcal{N}| = N$, which are deployed on a 2D plane. Although our proposal can be extended to a 3D space, we confine our scenarios to 2D due to the limitation of our experiment conditions. Let $\{u_i\} \subseteq \mathbb{R}^2$ denote the locations of the nodes. We denote by $\mathcal{N}(n_i)$ the one-hop neighbors of node n_i . Node n_i can utilize range information of its one-hop neighbors to construct a local coordinate system [69], if there is no positioning device equipped.

Given a frequency band $\mathcal{B} = [f_{min}, f_{max}]$ and the channel width f_w , we can assign each nodes n_i a center frequency $f_i \in \mathcal{B}'$, where $\mathcal{B}' = [f_{min} + \frac{f_w}{2}, f_{max} - \frac{f_w}{2}]$. Moreover, we introduce a density function $\Phi(f)$ (with $f \in \mathcal{B}'$) which defines the qualities of the frequencies. In particular, the frequencies less interfered by WiFi have higher density values. This density function can be disseminated by a central station (e.g., sink node) within the network. Considering both location and frequency, the representations of nodes are extended from 2D to 3D space, i.e., each node has a new “coordinate” $(u_i, f_i) \in \mathbb{R}^2 \times \mathcal{B}'$.

Each node n_i maintains a *CSMA probability* $Pr_{csma}(n_i) \in [0, 1]$ indicating the probability of sending a packet with CSMA enabled. We also use *qualification range* in PRR: $\mathcal{P} = [P_{min}, P_{max}]$ to describe the performance required by specific applications. The data transmission is said to be *qualified* if the measured PRR prr falls into the range, i.e., $prr \in \mathcal{P}$.

5.3.2 FAVOR: A Location-Aware Frequency Allocation Scheme

We hereby present our FAVOR algorithm, including its optimization objective and the algorithm to find a local optimal solution. We also discuss a practical implementation issue at the end.

5.3.2.1 FAVOR Objective: Balancing Distance and Frequency

According to our observations in Sec. 5.2.2, a frequency allocation scheme should assign very different frequencies to two nodes that are close to each other but arbitrary frequencies to nodes that are far from each other. A possible location-dependent frequency allocation is illustrated in Fig. 5.8.

We define the following *energy metric* for each sensor node based on Euclidean norm in the frequency-distance space:

$$\|v - u_i\|_{\ell^2} + \beta \|f - f_i\|_{\ell^2},$$

where $v \in \mathbb{R}^2$ and $f \in \mathcal{B}'$, β is a weight to balance frequency and distance. Without loss of generality, we let $\beta = 1$ to simplify our derivation. We divide the location-frequency space $\mathcal{A} = \mathbb{R}^2 \times \mathcal{B}'$ into Voronoi tessellation $\{V_i\}$ with $n_i = (u_i, f_i)$ serving

Figure 5.8: Allocating (center) frequencies to a 2D WSN based on the locations of nodes. The black points on the 2D plane denotes the nodes, while the “poles” of different heights on the points indicate different frequencies.

as generators. Each sensor node n_i has an energy F_i defined as

$$F_i(V_i, f_i) = \int_{V_i} (\|v - u_i\|_{\ell^2}^2 + \beta \|f - f_i\|_{\ell^2}^2) \Phi(f) dz, \quad (5.2)$$

where $z = (u, f) \in V_i$. The interference relationship can be represented by the dual graph of the Voronoi Tessellation: nodes close in both frequency and distance have adjacent Voronoi cells and are thus connected in the dual graph, which implies they may interfere each others.

We aim at an optimal “tessellation” of the subset $\mathcal{A} \in \mathbb{R}^3$ such that the following FAVOR objective is minimized.

$$\mathfrak{F}(\{V_i\}, \{f_i\}) = \sum_{i=1}^N F_i(V_i, f_i). \quad (5.3)$$

As mentioned in Sec. 5.2.5, this is equivalent to minimizing the interference among the sensor nodes according to the dual graph. It is obvious that (5.3) differs from (5.1) mainly in that part of the coordinates are fixed: we do not get the freedom to move nodes, only the frequencies allocated to them are variables. Fortunately, the absence of the freedom does not compromise the ability of CVT in minimizing interference. According to what we discussed in Sec. 5.2.5, an optimal solution tends to spread out the sensor nodes in the frequency-location space. Since the locations in $\mathbb{R}^2$ are

fixed, minimizing FAVOR objective function can be done only by separating the closed nodes in the frequency dimension. In particular, frequencies for two close-by nodes will be pushed further from each other. Moreover, through the density function, we can encourage the nodes to choose the frequencies less interfered by WiFi; thereby reducing the reliance on CSMA to deal with WiFi interference.

5.3.2.2 FAVOR Algorithm: A CVT-based Approach

The challenge we face now is twofold: i) as we need to perform CVT at least in a 3D space (it can be 4D if nodes are distributed in a 3D space), we need an algorithm more efficient than the Lloyd's method [96]; otherwise, nodes with limited computation resource cannot afford it, and ii) part of the coordinates for each $u_i \in \mathcal{A}$ are fixed, whereas CVT requires $\{u_i\}$ to be variables. To tackle these issues, we first propose *Approximate CVT* (A-CVT) to transform the problem into a more tractable and implementable form, then we apply gradient projection method to handle the fixed coordinates.

Given a region $\mathcal{A} = \mathbb{R}^2 \times \mathcal{B}' \subseteq \mathbb{R}^3$ and suppose we apply Voronoi tessellations to partition $\mathcal{A}$, we may re-write the objective (5.3) as the following.

$$\mathfrak{F}(\{V_i\}, \{f_i\}) = \int_{\mathcal{A}} \min_i (\|v - u_i\|_{\ell^2}^2 + \|f - f_i\|_{\ell^2}^2) \Phi(f) dz \quad (5.4)$$

The equivalence between (5.3) and (5.4) is obvious: for each generator (u_i, f_i) , integrating over its own cell V_i implies an integration over all the points in $\mathcal{A}$ that are closer to (u_i, f_i) than to any other generators (by the definition of a Voronoi cell in Sec. 5.2.5). Now we get a global integration over $\mathcal{A}$, eliminating the need for re-computing the Voronoi tessellations in every iteration. Furthermore, we apply an approximation to $\min(\cdot)$ to make the problem tractable, leading to the following A-CVT objective.

$$\mathfrak{F}(\{V_i\}, \{f_i\}) = \int_{\mathcal{A}} \left[\sum_i ((\|v - u_i\|_{\ell^2}^2 + \|f - f_i\|_{\ell^2}^2) \Phi(f))^\lambda \right]^{\frac{1}{\lambda}} dz. \quad (5.5)$$

The proof of (5.5) converging to (5.4) when $\lambda \rightarrow -\infty$ is rather trivial and is thus omitted. In practice, we take $\lambda \in [-40, -20]$.

As minimizing the A-CVT objective (5.5) is a typical non-linear optimization problem, we apply a gradient-descent method with gradient projection to search for a local minimum. The pseudocodes of the algorithm are shown by **Algorithm 6**. Each sensor

node n_i first measures the distances to the nodes belonging to $\mathcal{N}(n_i)$, based on which a certain space embedding technique (e.g. [69]) is applied to construct a local coordinate system (line 1). This step is not necessary if additional positioning mechanisms/devices are available. Roughly, the main part of the algorithm proceeds in rounds and takes

---

Algorithm 6: FAVOR

Input: For each $n_i \in \mathcal{N}$, initial frequency $f_i^0 \in \mathcal{B}'$, stopping tolerance ε_1 and ε_2

Output: f_i^* for each n_i

```

1 Construct a local coordinate system to obtain the locations of  $\mathcal{N}(n_i)$ 
2 For every node  $n_i \in \mathcal{N}$  in each round (every  $\tau$  ms):
3   Compute the gradient  $g(z_i)$  of (5.5)
4   Project  $g(z_i)$  on  $f_i$  to get  $g_f(z_i)$ 
5    $f_i^+ = f_i - \alpha \cdot g_f(z_i)$  /* $\alpha$  is the step size*/
6   if  $|g_f(z_i)| > \varepsilon_1 \vee |f_i^+ - f_i| > \varepsilon_2$  then
7      $f_i \leftarrow f_i^+$ ; BROADCAST( $f_i$ ) to nodes in  $\mathcal{N}(n_i)$ 
8   else
9      $f_i^* \leftarrow f_i^+$ 
10  end
    
```

---

the following three steps in each round:

1. Compute gradient for each generator z_i as (line 3)

$$g(z_i) = \int_{\mathcal{A}} 2 (\Phi(f) \|z - z_i\|_{\ell^2}^2)^{\lambda-1} \Phi(f)(z_i - z) \left(\sum_j (\Phi(f) \|z - z_j\|_{\ell^2}^2)^\lambda \right)^{\frac{1-\lambda}{\lambda}} dz \quad . \quad (5.6)$$

In order to facilitate localized computation and also to reduce the complexity, the summation in the third term can be applied only for $j : n_j \in \mathcal{N}(n_i)$ and the integration can be done only for v in the neighborhood of z_i . This is possible because the terms introduced by those far-away locations contribute only insignificantly to $g(z_i)$, due to the very small value of λ . This is also intuitively correct as the change in z_i for CVT is only affected by z_j whose cell shares boundaries with that of z_i . For a network, we can use the communication neighborhood to approximate the tessellation neighborhood.

2. As u_i is fixed and we can change only f_i , we take $g_f(z_i)$ as the projection of $g(z_i)$ on the frequency axis (line 4).

3. A tentative update is applied to the frequency by $f_i^+ = f_i - \alpha \cdot g_f(z_i)$ where α is a step size. If both $|g_f(z_i)|$ and $|f_i^+ - f_i|$ become sufficiently small, the algorithm is terminated, returning the optimal frequency allocation (line 9); otherwise, the frequency of each n_i is updated by $f_i \leftarrow f_i^+$, the outcome is exchanged among neighboring nodes (line 7), and further computation will be conducted during the next round.

We omit the convergence analysis as it follows directly from the basic theory of gradient-descent methods [97]. We also note that, if a centralized computing is allowed, we can use the Quasi-Newton method to solve (5.5), which often leads to a faster convergence.

We take a one-dimensional network¹ as an example and illustrate the result of FAVOR in Fig. 5.9. In order to show the effect of the density function $\Phi(f)$, we assume a $\Phi(f)$ increasing in f (indicated by the color spectrum, i.e., higher frequency is less interfered by WiFi). All nodes are initially assigned ascending frequencies according to the node IDs with small frequency separations. The outcome of FAVOR shows that the neighboring nodes are assigned frequencies that are significantly further apart (especially for more interfered frequencies), while more nodes choose higher frequencies to avoid being interfered by WiFi.

5.3.2.3 Frequency Allocation for Disjoint Links

As FAVOR relies on a set of (point) locations $\{u_i\}$ to perform allocation, an obvious difficulty it may face is what if no obvious points exist in a networking scenario. In particular, a network usually consists of several point-to-point wireless links. We propose two possible solutions for centralized and distributed computing respectively. If a centralized computing is feasible, we may apply the extended Voronoi diagram where generators are not points but line segments (representing the links) [98]. Whereas this method may result in rather accurate frequency allocation, computing Voronoi tessellation with line segments as generators is quite time consuming. Therefore, we pick one point to represent each link in a distributed computing environment. This point can be either the source or the destination of the link, or it can even be the middle point of the link. If a point is chosen as the source or the destination, the computation

---

¹Our algorithm actually applies to 2D and 3D cases, but we only show the results of a 1D deployment to avoid visual confusion.

Figure 5.9: Frequency allocation based on Voronoi tessellations. A density function $\Phi(f)$ is defined on the frequency-axis. The warm and cold colors indicate the high and low values of $\Phi(f)$, respectively. Given a regular (ascending) initial allocation (a), our algorithm outputs a much better allocation (b). In both cases, the corresponding Voronoi tessellations are also shown using red line segments as cell boundaries.

is performed by that node. If the middle point is chosen, the computation can be done by either of the two nodes. Our experiments reported in Sec. 5.4.3 (which takes the distributed/point approach) show that replacing line segments by points still leads to very good performance.

5.3.3 P-CSMA: Randomizing the Deployment of CSMA

Although FAVOR allows us to better utilize the spectrum resource, exploiting only the “clean” spectrum (2475 to 2480 MHz) may not be sufficient for densely deployed ZigBee networks, and so certain ZigBee links have to step into the WiFi interfered frequency band, e.g., those frequencies in the non-peak areas shown by Fig. 5.6. Therefore, CSMA is still necessary to guarantee transmission quality, but it should be deployed in a more flexible manner. Motivated by the observations presented in Sec. 5.2.3 and 5.2.4, we propose P-CSMA to achieve a balanced performance between throughput and PRR through a randomized control.

In P-CSMA, each transmitting node enables CSMA to send a packet with probability Pr_{csma} . Specifically, Pr_{csma} is tuned up to maintain transmission quality at a minor cost of throughput when experiencing severe WiFi interference; otherwise it is tuned down to improve throughput. It has been reported that WiFi interference exhibits temporal variation [14], but according to our experience, the temporal variation of WiFi interference is highly unpredictable, especially when multiple WiFi networks exist. Therefore, to identify the temporal white space of WiFi interference, we opt for a real-time evaluation method in our P-CSMA protocol, rather than the model-driven predication method proposed in [14], such that a transmitter adaptively tunes Pr_{csma} based on the real-time interference assessment provided by the receiver. In the following, we explain in detail how the real-time assessment is conducted and how Pr_{csma} is tuned accordingly.

5.3.3.1 Real-Time Interference Assessment

As mentioned earlier, we use PRR to indicate the interference level. A PRR is computed by a receiving node and is reported back to the transmitting node periodically. We disable ACK for most of the packets but only enable it for the receiving node to send a PRR report. Data packets delivered by P-CSMA are grouped into *transmission windows* of size W , i.e., each window contains W data packets, and each data packet carries an ID field to indicate its source and the transmission window to which it belongs. We denote by $pkt_{j,k}^i$ the j -th data packets involved in the k -th transmission window of node n_i , where $k = 1, 2, \dots$ and $j = 1, 2, \dots, W$. Data packets not belonging to a window have $k = 0$. An operating example of P-CSMA is shown in Fig. 5.10, where we use n_t and n_r to denote the transmitting and receiving nodes, respectively.

A transmitting node n_t sends the data packets in a transmission window with its current CSMA probability $Pr_{csma}(n_t)$. Upon sending out all packets in a window, n_t expects a PRR report piggyback with an ACK from the receiving node n_r , and then updates $Pr_{csma}(n_t)$ accordingly. When waiting for the PRR report, n_t does not suspend its packet transmissions; it instead keeps transmitting *inter-window data packets* (i.e., the packets between two successive transmission windows) but with CSMA deterministically enabled, so as to ensure the execution of PRR reporting on one hand and to avoid interfering the transmissions of PRR reports in opposite direction on the

Figure 5.10: An example of P-CSMA. In each transmission window, whether CSMA is enabled for individual data packets is determined by the current CSMA probability. If n_r correctly receives the last data packet in a transmission window (a DPRQ), it will feed the PRR of this window back to n_t . In case that this DPRQ or its corresponding PRR report gets loss, n_t keeps sending inter-window data packets (another type of DPRQ) with CSMA deterministically enabled until the expected PRR report arrives. For every DPRQ from n_t , n_r responds with the PRR of the latest window.

other hand. Finally n_t starts the next transmission window with the newly updated $Pr_{csma}(n_t)$.

By checking the ID fields of the data packets, the receiving node n_r can identify the last data packet in the current transmission window, $pkt_{W,k}^t$ ($k \neq 0$). Upon receiving this tail packet, n_r feeds a PRR report back to n_t by attaching it to an ACK. The PRR for each transmission window can be computed by dividing the number of received packets by W . To ensure a successful PRR reporting, n_r reports PRR upon every inter-window packet (if any) that can be identified by checking if $k = 0$, in case that the tail packet in the past window or the previous PRR reports get lost. These data packets (including $pkt_{W,k \neq 0}^t$ and $pkt_{j,0}^t$) that can trigger n_t 's PRR reporting behavior are termed *Data Packets with Report Query* (DPRQ) in Fig. 5.10.

5.3.3.2 Tuning CSMA probability

Being aware of the immediate PRR that reflects the intensity of the current WiFi interference, the transmitting node n_t adapts its CSMA probability $Pr_{csma}(n_t)$ accordingly, and then starts a new transmission window. The objective of $Pr_{csma}(n_t)$ updating is to maintain the PRR of next transmission window in the predefined qualification range $\mathcal{P}$. In our implementation, $Pr_{csma}(n_t)$ is tuned in an additive manner with a step size β in Equation 5.7).

$$Pr_{csma}^+ = \begin{cases} \min\{Pr_{csma} + \beta, 1\} & \text{if } prr < P_{min} \\ \max\{Pr_{csma} - \beta, 0\} & \text{if } prr > P_{max} \\ Pr_{csma} & \text{otherwise} \end{cases} \quad (5.7)$$

Specifically, if the measured PRR is below our predefined qualification range ($prr < P_{min}$), we need to increase $Pr_{csma}(n_t)$ to guarantee the transmission quality in face of severe WiFi interference. Otherwise if the PRR is beyond the qualification range ($prr > P_{max}$), the current $Pr_{csma}(n_t)$ is too conservative. So we can decrease its value to gain a higher throughput. According to our earlier experiment results (shown in Fig. 5.7), both throughput and PRR are very sensitive to $Pr_{csma}(n_t)$ in the range of $[0, 0.1]$. To allow for a finer granularity in adjusting $Pr_{csma}(n_t)$, our implementation takes $\beta = 0.01$ if $Pr_{csma}(n_t) \in [0, 0.1]$; otherwise, $\beta = 0.1$.

5.3.4 Application: Tree-based Data Collection

As data collection is a fundamental operation in ZigBee-based WSNs, we hereby discuss how ART can be applied to this operation. During a data collection process, a set of nodes $\{n_i\}$ send their collected data towards a common sink node n_s , usually over a tree-based routing topology with n_s as the root of the tree (e.g., [99]).

When applying FAVOR as a multi-channel access mechanism, we need to handle the conflicts incurred by multiple links sharing the same incident node, as such conflicts are not the consequence of co-channel interference but of a node having only one radio. Therefore, we need a TDMA-like time schedule to assist ART with resolving the type of conflicts. While a joint frequency allocation and scheduling problem is beyond the scope of this thesis, we simply adapt the minimum latency scheduling mechanism proposed in [100]. We basically arrange the nodes into layers according to their hop-distances to the sink on the collection tree T , then we use a labeling mechanism similar to [100].

The idea of this labeling is twofold: i) to guarantee that the number of labels assigned to an outgoing link of node n_i should be 1 plus the number of descendants of n_i in T , and ii) to assign different labels for links sharing the same node. Whereas the proposal in [100] adopts (orthogonal) channel allocation to enable parallel transmissions and allocates channels using a first-fit distance- $(\rho + 1)$ graph coloring, our FAVOR allocates overlapped channels and thus potentially allows for more parallel transmissions.

Under ART, each node except leaf nodes is assigned with a center frequency by FAVOR for receiving the data packets from its child nodes, whereas the leaf nodes (being receivers to no one) do not need their own receiving frequencies. At a given point in time, for those links that are activated according to the time schedule discussed above, the receivers switch to their own receiving frequencies, while the transmitters tune to the receiving frequencies of the receivers as well. Every pair of nodes forming an active link perform P-CSMA during their data transmissions so as to adapt the use of CSMA to WiFi interference. If a network deployment is sparse, FAVOR will assign only clean frequencies to the nodes, and thus P-CSMA will often disable CSMA by assigning a small value or zero to Pr_{csma} . Otherwise if a network is so dense that the WiFi interfered frequencies have to be used to sufficiently space nodes in frequency spectrum, FAVOR will employ the density function to assign some nodes with the polluted frequencies, and P-CSMA for the affected links will start to deploy CSMA by raising the value of Pr_{csma} .

Our ART framework can be adapted to other types of communication patterns, such as broadcast (dissemination commands from the sink to the network) and aggregation (each relay node may send out less data than it receives). However, we focus only on data collection in this chapter.

5.4 Evaluation

We have implemented ART in our MicaZ platforms and performed extensive experiments on them. We first present the basic configuration of our ART framework for our experiments, then we report the results we get in different network scenarios. Finally, we briefly examine the convergence of our distributed FAVOR algorithm serving as an important component of our ART framework.

5.4.1 Experiment Settings

We adopt MicaZ motes and TinyOS 2.1 as the hardware and software platforms, respectively. We perform our experiments within the frequency spectrum of $\mathcal{B} = [2449, 2459]$ MHz that is partially interfered by the WiFi networks already deployed in our research center. As a ZigBee channel has a width of 2 MHz, the center frequencies we can employ are actually within the ranges of $\mathcal{B}' = [2450, 2458]$ MHz. According to the ZigBee standard [13], there are only two channels specified within this spectrum, i.e., Channels 20 and 21 with center frequencies at 2450 and 2455 MHz, respectively. We term this channel allocation scheme *two-channel*. We also test the proposal in [55], where a frequency separation of 3 MHz is suggested. Given the available spectrum, we may have at most three channels (e.g., those with center frequencies at 2450, 2453 and 2456 MHz), and we term this scheme *three-channel*. We apply a greedy graph coloring approach to allocate channels for both two- and three-channel schemes, while our ART employs FAVOR to allocate all nine potential channels (center frequencies) to nodes. In the following, we use only center frequency to indicate a channel. Considering PRR as a metric to measure frequency quality, we define a density function $\Phi(f)$ for our FAVOR algorithm based on the results shown in Fig. 5.6(d). which ends up with the density function shown in Fig. 5.11.

Figure 5.11: Density function $\Phi(f)$ defined for $[2450 - 2458]$ MHz band.

We deploy our WSNs on the ceiling of our laboratory (see Fig. 5.12), emulating an indoor monitoring application scenario. We perform each experiment for 5 minutes, and use the *Total Received Packets* (TRP) and PRR at each destination node as the

performance measures. Note that, the value of TRP is actually an indicator of throughput, while PRR is the metric for transmission quality. We set the qualification range of PRR as $\mathcal{P} = [0.85, 0.9]$, and initialize the CSMA probabilities to 0.2 for all nodes.

Figure 5.12: A MicaZ-based WSN testbed on the ceiling of our research center.

5.4.2 ART on A Single Link

We first verify the efficacy of ART on a single link. As we do not need channel allocation in this case, only P-CSMA is applied on this link. We let the transmitting node persistently send data packets to the receiver as fast as possible on the channel at 2455 MHz, which suffers from rather severe WiFi interference (as shown in Fig. 5.11). Besides, we vary the transmission window size ω from 100 to 500 to investigate the influence of ω on the performance of ART. We also compare our ART with *BuzzBuzz* [15], a MAC layer protocol for ZigBee to survive WiFi interference, which can be performed by individual links with CSMA enabled. As a benchmark, we also consider the case of deterministically disabling CSMA. We run each of the three configuration for 10 times, and compare them in terms of (average) throughput and PRR.

As demonstrated by Fig. 5.13 (a), ART delivers a much higher throughput than the CSMA-based *BuzzBuzz*. Especially when $\omega = 500$, the throughput is nearly 5 times of the one delivered by *BuzzBuzz*, very close to the maximum achievable throughput at this frequency (indicated by the red dashed lines). In terms of PRR, Fig. 5.13 (b) shows that all transmission window sizes yield a PRR of almost 0.9. This, on one

Figure 5.13: The throughput (a) and PRR (b) of a transmission link at 2455 MHz with different transmission window sizes. The green solid lines indicate the performance of CSMA-based *BuzzBuzz*, while the red dashed lines represent the performance of deterministically disabling CSMA.

hand, satisfies our target qualification range, and on the other hand, goes very close to *BuzzBuzz* (which has a PRR of 0.92), even though *BuzzBuzz* employs sophisticated techniques (i.e., multiple headers and Reed-Solomon coding) to pursue perfect PRR on the basis of CSMA. In a nutshell, ART obtains a considerable improvement in terms of throughput with a negligible compromise in PRR.

Fig. 5.13 also shows that, with an increasing transmission window size, throughput is slightly increased, while PRR experiences a minor degradation. As increasing ω results in less transmission windows within the same period, the reduced inter-window transmissions (all with CSMA enabled) then helps to improve the throughput to some extent. Moreover, less updates of the CSMA probability lead to poorer adaptability to the dynamic WiFi interference, which accounts for the minor degradation of PRR.

5.4.3 A Five-Link Scenario

We test a scenario with five links $\{l_i = (s_i, r_i), i = 1, \dots, 5\}$ as shown in Fig. 5.14. We deliberately deploy the five links in a relatively small area (about 20 m^2), in order to emulate a small part of a densely deployed WSN (which we do not have in full scale

htb

Figure 5.14: A five-link deployment.

at our disposal). The frequency allocations based on different schemes are shown in Table 5.1. It is obvious that, in the three schemes other than ART-FAVOR, the number of available channels is smaller than the number of links; therefore, some links do have to share the same channel, while our FAVOR utilizes the spectrum more efficiently and thus can allocate different channels (center frequencies) to the five links. Furthermore, FAVOR allocates as different frequencies as possible to the links that are closed to each other in distance. Finally, due to the density function (Fig. 5.11) used by FAVOR, three of five links (i.e., l_2 , l_3 and l_5) choose frequencies that are less interfered by WiFi (thus with higher values in $\Phi(f)$). We believe that the superiority of ART-FAVOR can become more evident if more links are involved.

Table 5.1: Frequency allocations for the five-link scenario based on different schemes.

Links	l_1	l_2	l_3	l_4	l_5
Single-channel	2455	2455	2455	2455	2455
Two-channel	2450	2455	2450	2455	2450
Three-channel	2450	2453	2456	2450	2453
ART-FAVOR	2456	2453	2450	2458	2451

We test all seven schemes (with different combinations of channel allocations and CSMA control strategies), and report the results in terms of TRP and PRR in Fig. 5.15 (a) and (b), respectively. For each reported data point, we perform 10 experiments, and plot their statistical quantities such as means, median, and interquartile ranges. As

(a) Statistics on TRPs for all links

(b) Statistics on PRRs for all links

Figure 5.15: Experiment results for the five-link scenario.

shown in Fig. 5.15(a), ART operates l_2 , l_3 and l_5 much better than the other schemes; the three links nearly achieve their full data rates. On one hand, our FAVOR algorithm allocates three less WiFi-interfered center frequencies to the three links; on the other hand, for links closed to each other (e.g., l_2 and l_3), largely separated frequencies are assigned. Although l_1 chooses a center frequency far away from closed l_2 and l_5 , l_1 has a relatively lower throughput due to WiFi interference. Similar with l_1 , the link l_4 has

to entail a compromise in transmission throughput. Fortunately, thanks to the flexible coordination provided by P-CSMA, ART performs fairly well as we expect: the two links, l_1 and l_4 both have a throughput much higher than the ones achieved by the schemes with CSMA deterministically enabled. The superiority of ART in throughput can also be confirmed by its very small interquartile ranges that actually indicate a very stable performance (which normally can be obtained only by enabling CSMA). In terms of PRR, Fig. 5.15(b) shows that ART offers qualified PRRs for all five links, comparable to other schemes where CSMA is enabled (whereas those schemes achieves an excellent PRR performance at the cost of a poor throughput). In summary, as ART combines FAVOR and P-CSMA to make the best use of both frequency and temporal white space left out by WiFi interference, as well as to avoid the ZigBee co-channel interference, it offers the opportunity to greatly improve throughput while maintaining the transmission quality at an excellent level.

5.4.4 Tree-Based Data Collection

In order to demonstrate the benefit of applying ART to a practical application, we test the performance of different schemes in several arbitrary data collection trees deployed in our research center (an area of roughly $600m^2$). As the results obtained are all similar, we only report one case in the following. For this case, the node locations, as well as the network topology, are shown in Fig. 5.16 (a). The frequencies (or channels) are allocated to non-leaf nodes to receive data packets from their child nodes, and as we have discussed in Sec. 5.3.4, we apply the labeling method in [100] to schedule links sharing the same incident node.

According to what is shown in Sec. 5.4.3, the schedule needs also to prevent the links with the same frequency from transmitting simultaneously (for the schemes without CSMA). This leads to the obvious consequence that, the more frequencies we can allocate, the less time slots we need in one round (during which every node gets a chance to transmit). As a result, two-channel scheme needs at least 29 slots, three-channel scheme needs at least 24 slots, while ART-FAVOR needs only 22 slots. For single channel, we enable CSMA and apply CTP [99] to perform data collection. We also enable CSMA for two- and three-channel schemes, but with a different schedule: it needs only to guarantee a sender and a receiver staying at the same frequency when the link between them is active. Consequently, only 9 time slots are needed. Given limited

(a) Data collection tree (frequency allocation marked on links).

1	2	3	4	5	6	7	8	9	10	11	12
$n_2 \rightarrow n_1$						$n_3 \rightarrow n_1$					
$n_{14} \rightarrow n_7$	$n_{15} \rightarrow n_8$	$n_{16} \rightarrow n_8$				$n_{17} \rightarrow n_{10}$					

13	14	15	16	17	18	19	20	21	22
$n_4 \rightarrow n_1$				$n_5 \rightarrow n_2$				$n_6 \rightarrow n_2$	
$n_{11} \rightarrow n_5$	$n_{12} \rightarrow n_5$	$n_{13} \rightarrow n_6$		$n_7 \rightarrow n_3$		$n_9 \rightarrow n_4$	$n_8 \rightarrow n_3$		
							$n_{10} \rightarrow n_4$		

(b) Minimum delay schedule on the tree.

Figure 5.16: Data collection tree with FAVOR frequency allocation (a) and min-delay transmission schedule (b). The node locations in (a) are plotted roughly proportional to their actual locations in our ceiling testbed.

space, we show only the frequency allocation and schedule for FAVOR in Figure 5.16 (b).

We report the results in terms of TRP and PRR in Fig. 5.17 (a) and (b) respectively. Different from Sec. 5.4.3, here both TRP and PRR are defined for each node and measured at the sink. As expected, the results in Fig. 5.17 (a) show that ART surpasses all other schemes, and it achieves a throughput three times of that achieved by the single-channel CTP. Apparently, ART beats two- and three-channel schemes due to

(a) Total Packet Reception (TPR) at the sink.

(b) Packet Reception Ratio (PRR) at the sink.

Figure 5.17: Performance of six different schemes in a data collection tree.

the less time slots in a round offered by FAVOR. Meanwhile, it prevails against all CSMA-enabled schemes thanks to P-CSMA’s adaptive CSMA adjustment. Moreover, ART allows for a more fair sharing of the bandwidth (every node gets roughly the same throughput), which cannot be guaranteed by any other schemes. In terms of PRR, it is shown in Fig. 5.17 (b) that ART achieves a nearly perfect transmission quality, even better than those CSMA-enabled schemes. The reason is that, when multiple co-channel transmissions proceed simultaneously, even CSMA cannot perfectly resolve such conflicts with its procrastination (see Sec. 5.2.1); whereas ART employs FAVOR to fully exploit the spatial-frequency space, providing sufficient room to counteract the interference between concurrent transmissions. This also reduces the dependence on CSMA and thus endows P-CSMA with more opportunities in “relaxing” CSMA to improve throughput. Overall, we have proved that ART is able to guarantee both throughput and PRR even under WiFi interference.

5.4.5 Convergence of FAVOR

We briefly verify the convergence of FAVOR (in terms of communication rounds) in a 30-node WSN. Each node periodically communicates with its one-hop neighbors to calculate its receiving frequency, and also informs its one-hop neighbors of this temporary decision. We unify both frequency and distance into a scale of $[0, 1]$ and employ the density function shown in Fig. 5.11 whose image is also normalized into $[0, 1]$. The A-CVT objective value is thus always small than 1. As illustrated by Fig. 5.18, the convergence under different step sizes (i.e., α) often takes 25-30 rounds. As such message exchanges can piggyback with other transmission activities and frequency (re)-allocation does not happen very often, the overhead of FAVOR (in terms of the entailed communication and computation costs) is affordable, especially considering the substantial improvement it is able to bring.

Figure 5.18: The convergence of FAVOR in a 30-node WSN.

5.5 Summary

In this chapter, we have presented ART as a novel framework for guaranteeing both network throughput and transmission quality (in terms of PRR) for ZigBee-based WSNs in face of WiFi interference. ART consists of two components: FAVOR and P-CSMA.

Shifting the paradigm from discrete channel allocation to continuous frequency allocation, FAVOR innovates in transforming the optimal frequency allocation problem to a spatial tessellation problem in a frequency-spatial space, so as to adapt the CVT

method to search for a near-optimal frequency allocation. By continuously exploiting the spectrum, FAVOR is able to utilize the frequency white space of WiFi interference while largely eliminating the need for CSMA to handle ZigBee interference; thereby significantly improving network throughput. Liberated from dealing with internal ZigBee interference, CSMA can thus be smartly employed to cope with WiFi interference. Aiming to improve throughput while maintaining PRR at a satisfactory level, our P-CSMA is built upon CSMA but with a randomized control and a real-time interference assessment scheme, such that the use of CSMA can be adaptively tuned to exploit the temporal white space of WiFi interference.

In order to demonstrate the efficacy of ART, we have performed intensive experiments on our MicaZ-based WSN testbed under various network settings. All the experiment results have strongly proved the feasibility and superiority of ART in tackling the ZigBee-WiFi co-existence issue.

Chapter 6

Conclusion

The scales of the wireless sensor networks have been increasing with their broadening application scope, while such a developing trend raises many novel challenges in designing algorithms. In this thesis, we exploit the geometric structures of the large-scale WSNs, and design localized geometric algorithms to address various problems in large-scale WSNs, including but not limited to boundary detection, sensor deployment, and co-existence of heterogeneous network deployments. In particular, since the dependency on global coordination is removed, the algorithms can be performed by individual sensor nodes in a fully distributed fashion relying on only local information.

6.1 Research Contributions

In Chapter 3, we have thoroughly studied the boundary detection and regulation problem for 3D large-scale WSNs. We have modeled the 3D WSNs as point clouds sampled from 3D volumes with connectivity structures, based on which, we have proposed LBDP. The first component of LBDP, UNFOLD, reduces the boundary detection to local convexity test through introducing a spherical inversion. Since UNFOLD only relies on local information, it is able to significantly speed up the detection process. We also have presented LBS as another important component of LBDP, which employs a localized parameterization technique to regularize the arbitrarily shaped network boundary surfaces to spheres. We have demonstrated that, this regulation operation can be used to support many other network functionalities, e.g., telling internal boundaries from external ones, and geographic data routing.

Another topic we concern in this thesis is multiple coverage problem. In Chapter 4, we have proposed APOLLO for optimally deploying mobile sensor nodes to k -cover the targeted region. We have formulated the k -coverage problem as classical “space tessellation” problem which has been well studied in geometry community. Inspired by k -order Voronoi tessellation, APOLLO autonomously conducts the movements of the mobile sensor nodes to reach k -coverage with the maximum sensing range among the sensor nodes minimized for load balancing purpose. Besides, we have revealed the relationship between our objective and many other commonly used ones, which delivers a better understanding for our load balancing k -coverage problem. Furthermore, we have generalized 2D k -plane coverage to 3D k -surface coverage under an uniform framework.

In Chapter 5, we have first carried out extensive experiments to reveal the root cause of the low performance of ZigBee-based WSNs in face of WiFi interference, according to which, we proposed ART consisting two components, FAVOR and P-CSMA, to guarantee both ZigBee transmission throughput and quality. In FAVOR, we unify location and frequency into one continuous space where we adapt CVT theory to “tessellate” the sensor nodes in the distance-frequency space; thereby calculating a nearly optimal continuous frequency allocation. Our FAVOR considerably increases the utilization of frequency spectrum (especially the one less polluted by WiFi interference) to enhances the transmission parallelism, while librating ZigBee transmissions from the need for CSMA to cope with internal ZigBee interference. On the other hand, we have presented P-CSMA to probabilistically control the usage of CSMA for dealing with external WiFi interference based on a real-time interference assessment scheme. P-CSMA effectively maintains as high transmission throughput as possible with only minor sacrifice in transmission quality. Overall, the above two components fully exploiting frequency and time resource respectively; thereby significantly improving the network performance.

6.2 Future Directions

The boundary detection technique can provide fundamental service in many functionalities of WSNs. For example, as presented in Chapter 4, our UNFOLD algorithm (see Chapter 3) serves as an infrastructure under our sensor deployment strategy APOLLO.

Besides coverage area, the boundaries of a WSN also imply network topology information. We are on the way of applying our boundary detection algorithm to extract skeletons of WSNs, which can reflect global topology of the WSNs and further be applied in localization, routing and navigation.

As presented in Chapter 4, our current study on sensor deployment problem assumes a omnidirectional sensing range. Specifically, a disk centered at a sensor node is used to characterize the coverage of that node. However, real sensors may have certain directional features (e.g., radar or acoustic sensors). For autonomous deployments of mobile sensor networks, such a realistic sensing model offers two freedoms, i.e., the locations of the sensor nodes, and their orientations. Taking into account both of the freedoms to design autonomous deployment strategy for optimizing network coverage, is highly non-trivial, and demands our further studies.

One of the main observations driving our research in Chapter 5 is that, CVT can spread the generators in the targeted space in a localized manner. Such an idea can be applied to mobile sensing applications where mobile sensor nodes, vehicles equipped with sensing instruments, or crowdsourcing participants serve as generators. Specifically, we need to “spread” these sensing units in the targeted area to avoid data redundancy. One pivotal difference is that, in these application, the sensing procedure is a dynamic process: the sensing units complete their sensing tasks during the movements rather than in a static deployment. We could consider building a dynamic CVT model to favor these applications.

References

- [1] H. EDELSBRUNNER, D. KIRKPATRICK, AND R. SEIDEL. **On the Shape of a Set of Points in the Plane.** *IEEE Trans. on Information Theory*, **29**(4):551–559, 1983. xi, 11
- [2] H.M. AMMARI AND S.K. DAS. **Centralized and Clustered k-Coverage Protocols for Wireless Sensor Networks.** *IEEE Trans. on Computers*, **6**(1):118–133, 2012. xvii, 13, 42, 57, 66, 68, 70
- [3] I.F. AKYILDIZ, W. SU, Y. SANKARASUBRAMANIAM, AND E. CAYIRCI. **A Survey on Sensor Networks.** *IEEE Communication Mag.*, **40**(8):104–112, 2002. 1, 42
- [4] P. CORKE, T. WARK, R. JURDAK, W. HU, P. VALENCIA, AND D. MOORE. **Environmental Wireless Sensor Networks.** *Proceedings of the IEEE*, **98**(11):1903–1917, 2010. 1
- [5] **GreenOrbs: A Long-Term Kilo-Scale Wireless Sensor Network System in the Forest.** 1, 74
- [6] S. DAWSON-HAGGERTY, S. LANZISERA, J. TANEJA, R. BROWN, AND D. CULLER. **@scale: Insights from a Large, Long-Lived Appliance Energy WSN.** In *Proc. of the 11th ACM/IEEE IPSN*, pages 37–48, 2012. 1, 74
- [7] K. DANTU, M. RAHIMI, H. SHAH, S. BABEL, A. DHARIWAL, AND G.S. SUKHATME. **Robomote: Enabling Mobility in Sensor Networks.** In *Proc. of the 4th ACM/IEEE IPSN*, pages 404–409, 2005. 2, 42, 45

- [8] G. WANG, G. CAO, AND T.F. LA PORTA. **Movement-Assisted Sensor Deployment**. *IEEE Trans. on Mobile Computing*, **5**(6):640–652, 2006. 2, 42, 43, 52, 57, 58
- [9] N. BARTOLINI, T. CALAMONERI, T.F. LA PORTA, AND S. SILVESTRI. **Autonomous Deployment of Heterogeneous Mobile Sensors**. *IEEE Trans. on Mobile Computing*, **10**(6):753–766, 2011. 2, 13, 42, 52, 57, 58
- [10] **Chipcon’s CC2420 2.4G IEEE 802.15.4/ZigBee-ready RF Transceiver**. 3, 63, 74, 75, 77
- [11] Y. WU, J. STANKOVIC, T. HE, AND S. LIN. **Realistic and Efficient Multi-Channel Communications in Wireless Sensor Networks**. In *Proc. of the 27th IEEE INFOCOM*, 2008. 3, 14, 74
- [12] L. TANG, Y. SUN, O. GUREWITZ, AND D.B. JOHNSON. **EM-MAC: A Dynamic Multichannel Energy-Efficient MAC Protocol for Wireless Sensor Networks**. In *Proc of the 12th ACM MobiHoc*, pages 23:1–23:11, 2011. 3, 74
- [13] **IEEE 802.15.4 1999. IEEE 802.15.4, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Network (LR-WPANS)**. IEEE Std. 802.15.4, 2003. 3, 74, 80, 96
- [14] J. HUANG, G. XING, G. ZHOU, AND R. ZHOU. **Beyond Co-existence: Exploiting WiFi White Space for ZigBee Performance Assurance**. In *Proc. of the 18th IEEE ICNP*, pages 305–314, 2010. 3, 15, 74, 75, 92
- [15] C.-J.M. LIANG, B. PRIYANTHA, J. LIU, AND A. TERZIS. **Surviving Wi-Fi Interference in Low Power ZigBee Networks**. In *Proc. of the 8th ACM SenSys*, pages 309–322, 2010. 3, 15, 74, 97
- [16] R. XU, G. SHI, J. LUO, Z. ZHAO, AND Y. SHU. **MuZi: Multi-channel ZigBee Networks for Avoiding WiFi Interference**. In *Proc. of the 4th IEEE CPSCoM*, pages 323–329, 2011. 3, 4, 14

-
- [17] N. AMENTA AND M. BERN. **Surface Reconstruction by Voronoi Filtering.** *Springer Discrete & Computational Geometry*, **22**(4):481–504, 1999. 9
- [18] N. AMENTA, S. CHOI, T.K. DEY, AND N. LEEKHA. **A Simple Algorithm for Homeomorphic Surface Reconstruction.** *Internat. J. Comput. Geom. & Applications*, **12**(1-2):125–141, 2002. 9
- [19] R. NOWAK AND U. MITRA. **Boundary Estimation in Sensor Networks.** In *Proc. of the 2nd ACM/IEEE IPSN*, pages 80–95, 2003. 9
- [20] C. ZHANG, Y. ZHANG, AND Y. FANG. **Localized Algorithms for Coverage Boundary Detection in Wireless Sensor Networks.** *Springer Wireless Networks*, **15**(1):3–20, 2009. 9, 10
- [21] M. FAYED AND H. MOUFTAH. **Localised Alpha-Shape Computations for Boundary Recognition in Sensor Networks.** *Elsevier Ad Hoc Networks*, **7**(6):1259–1269, 2009. 9, 11
- [22] H. ZHOU, S. XIA, M. JIN, AND H. WU. **Localized Algorithm for Precise Boundary Detection in 3D Wireless Networks.** In *Proc. of the 30th IEEE ICDCS*, pages 744–753, 2010. 9, 11, 21, 28, 31, 34, 36, 37
- [23] Q. FANG, J. GAO, AND L.J. GUIBAS. **Locating and Bypassing Routing Holes in Sensor Networks.** In *Proc. of 23rd IEEE INFOCOM*, pages 2458–2468, 2004. 9, 10, 18, 32
- [24] R. GHRIST AND A. MUHAMMAD. **Coverage and Hole-Detection in Sensor Networks via Homology.** In *Proc. of the 4th ACM/IEEE IPSN*, pages 254–260, 2005. 9, 10
- [25] S. FUNKE. **Topological Hole Detection in Wireless Sensor Networks and its Applications.** In *Proc. of 3rd ACM DIAL-M-POMC*, pages 44–53, 2005. 9
- [26] A. KRÖLLER, S. FEKETE, D. PFISTERER, AND S. FISCHER. **Deterministic Boundary Recognition and Topology Extraction for Large Sensor Networks.** In *Proc. of the 17th ACM/SIAM SODA*, pages 1000–1009, 2006. 9, 10

- [27] S. FUNKE AND C. KLEIN. **Hole Detection or: "How much Geometry hides in Connectivity?"**. In *Proc. of 22nd ACM SCG*, pages 377–385, 2006. 10
- [28] Y. WANG, J. GAO, AND J. MITCHELL. **Boundary Recognition in Sensor Networks by Topological Methods**. In *Proc. of the 12th ACM MobiCom*, pages 122–133, 2006. 10, 20, 21
- [29] D. DONG, Y. LIU, AND X. LIAO. **Fine-Grained Boundary Recognition in Wireless Ad Hoc and Sensor Networks by Topological Methods**. In *Proc. of the 10th ACM MobiHoc*, pages 135–144, 2009. 10
- [30] H. ZHOU, H. WU, AND M. JIN. **A Robust Boundary Detection Algorithm Based on Connectivity Only for 3D Wireless Sensor Networks**. In *Proc. of the 31st IEEE INFOCOM*, pages 1602–1610, 2012. 10
- [31] G. TAN H. JIANG, S. ZHANG AND C. WANG. **Connectivity-Based Boundary Extraction of Large-Scale 3D Sensor Networks: Algorithms and Applications**. *IEEE Trans. on Parallel and Distributed Systems*, **25**(4):710–721, 2014. 10
- [32] M. DING, D. CHEN, K. XING, AND X. CHENG. **Localized Fault-Tolerant Event Boundary Detection in Sensor Networks**. In *Proc. of the 24th IEEE INFOCOM*, pages 902–913, 2005. 11
- [33] X. ZHU, R. SARKAR, J. GAO, AND J. MITCHELL. **Light-weight Contour Tracking in Wireless Sensor Networks**. In *Proc. of 27rd IEEE INFOCOM*, 2008. 11
- [34] M. CARDEI, M.T. THAI, Y. LI, AND W. WU. **Energy-Efficient Target Coverage in Wireless Sensor Networks**. In *Prof. of the 24th IEEE INFOCOM*, pages 1976–1984, 2005. 12, 42
- [35] Q. ZHAO AND M. GURUSAMY. **Lifetime Maximization for Connected Target Coverage in Wireless Sensor Networks**. *IEEE/ACM Trans. on Networking*, **16**(6):1378–1391, 2008. 12, 42, 58

-
- [36] K. HAN, L. XIANG, J. LUO, AND Y. LIU. **Minimum-Energy Connected Coverage in Wireless Sensor Networks with Omni-Directional and Directional Features.** In *Proc. of the 13th ACM MobiHoc*, pages 85–94, 2012. 12, 18, 42
- [37] S. KUMAR, T.H. LA, AND J. BALOGH. **On k -coverage in a Mostly Sleeping Sensor Network.** In *Proc. of the 10th ACM MobiCom*, pages 144–158, 2004. 12, 42
- [38] Z. ZHOU, S.R. DAS, AND H. GUPTA. **Variable Radii Connected Sensor Cover in Sensor Networks.** *ACM Trans. Sensor Networks*, **5**(1):8:1–8:36, 2009. 12, 42, 45
- [39] L. WU, H. DU, W. WU, D. LI, J. LV, AND W. LEE. **Approximation for Minimum Connected Sensor Cover.** In *Proc. of the 32nd IEEE INFOCOM*, pages 1187–1194, 2013. 12
- [40] Q. YANG, S. HE, J. LI, J. CHEN, AND Y. SUN. **Energy-Efficient Probabilistic Area Coverage in Wireless Sensor Networks.** *IEEE Trans. on Vehicular Technology*, 2014. 12
- [41] G. KASBEKAR, Y. BEJERANO, AND S. SARKAR. **Lifetime and Coverage Guarantees Through Distributed Coordinate-Free Sensor Activity.** *IEEE/ACM Trans. on Networking*, **19**(22):470–483, 2011. 12
- [42] R. KERSHNER. **The Number of Circles Covering A Set.** *American J. Math.*, **61**(3):665–671, 1939. 12
- [43] K. KAR AND S. BANERJEE. **Node Placement for Connected Coverage in Sensor Networks.** In *Proc. of the 1st IEEE/ACM WiOpt*, 2003. 12, 13
- [44] X. BAI, S. KUMAR, D. XUAN, Z. YUN, AND T.H. LAI. **Deploying Wireless Sensors to Achieve Both Coverage and Connectivity.** In *Proc. of the 7th ACM MobiHoc*, pages 131–142, 2006. 12, 13
- [45] X. BAI, Z. YUN, D. XUAN, B. CHEN, AND W. ZHAO. **Optimal Multiple-Coverage of Sensor Networks.** In *Proc. of the 30th IEEE INFOCOM*, pages 2498–2506, 2011. 12, 13, 42, 57, 66, 67

- [46] J.-E. KIM, J. HAN, AND C.-G. LEE. **Optimal 3-Coverage with Minimum Separation Requirements for Ubiquitous Computing Environments.** *Springer Mobile Netw. & Appl.*, **14**(5):556–570, 2008. 13, 58
- [47] Y.-C. WANG AND Y.-C. TSENG. **Distributed Deployment Schemes for Mobile Wireless Sensor Networks to Ensure Multilevel Coverage.** *IEEE Trans. on Parallel and Distributed Systems*, **19**(9):1280–1294, 2008. 13
- [48] M. ZHAO, J. LEI, M. WU, Y. LIU, AND W. SHU. **Surface Coverage in Wireless Sensor Networks.** In *Proc of the 28th IEEE INFOCOM*, pages 109–117, 2009. 13, 43, 57
- [49] M. JIN, G. RONG, H. WU, L. SHUAI, AND X. GUO. **Optimal Surface Deployment Problem in Wireless Sensor Networks.** In *Proc. of the 31st IEEE INFOCOM*, pages 2345–2353, 2012. 13, 43, 57
- [50] Q. DU, V. FABER, AND M. GUNZBURGER. **Centroidal Voronoi Tessellations: Applications and Algorithm.** *SIAM Review*, **41**(4):637–676, 1999. 13, 75, 84
- [51] M. JIN, J. KIM, F. LUO, AND X. GU. **Discrete Surface Ricci Flow.** *IEEE Trans. on Visualization and Computer Graphics*, **15**(5):1030–1043, 2008. 13
- [52] X. WANG, X. WANG, AND J. ZHAO. **Impact of Mobility and Heterogeneity on Coverage and Energy Consumption in Wireless Sensor Networks.** In *Proc. of the 31st IEEE ICDCS*, pages 477–487, 2011. 13
- [53] H.K. LE, D. HENRIKSSON, AND T. ABDELZAHER. **A Control Theory Approach to Throughput Optimization in Multi-Channel Collection Sensor Networks.** In *Proc. of the 6th ACM/IEEE IPSN*, pages 31–40, 2007. 14
- [54] G. XING, M. SHA, J. HUANG, G. ZHOU, X. WANG, AND S. LIU. **Multi-Channel Interference Measurement and Modeling in Low-Power Wireless Networks.** In *Proc. of the 30th IEEE RTSS*, pages 248–257, 2009. 14
- [55] X. XU, J. LUO, AND Q. ZHANG. **Design of Non-orthogonal Multi-channel Sensor Networks.** In *Proc. of the 30th IEEE ICDCS*, pages 358–367, 2010. 14, 96

-
- [56] M. DODDAVENKATAPPA, M. CHAN, AND B. LEONG. **Improving Link Quality by Exploiting Channel Diversity in Wireless Sensor Networks**. In *Proc. of the 33rd IEEE RTSS*, pages 159–169, 2011. 14
- [57] J. LUO, C. ROSENBERG, AND A. GIRARD. **Engineering Wireless Mesh Networks: Joint Scheduling, Routing, Power Control and Rate Adaptation**. *IEEE/ACM Trans. on Networking*, **18**(5):1387–1400, 2010. 15
- [58] K. HAN, Y. LIU, AND J. LUO. **Duty-Cycle-Aware Minimum-Energy Multicasting in Wireless Sensor Networks**. *IEEE/ACM Trans. on Networking*, **21**(3):910–923, 2013. 15
- [59] K. SRINIVASAN AND M. KAZANDJIEVA. **The β -factor: Measuring Wireless Link Burstiness**. In *Proc. of the 6th ACM SenSys*, pages 29–42, 2008. 15
- [60] R. XU, G. SHI, J. LUO, Z. ZHAO, AND Y. SHU. **MuZi: Multi-channel ZigBee Networks for Avoiding WiFi Interference**. In *Proc. of the 4th IEEE CPSCoM*, pages 323–329, 2011. 15
- [61] F. LI, J. LUO, S. XIN, W. WANG, AND Y. HE. **LAACAD: Load bALancing k-Area Coverage through Autonomous Deployment in Wireless Sensor Networks**. In *Proc. of the 32nd IEEE ICDCS*, pages 566–575, 2012. 18
- [62] G. TAN, H. JIANG, S. ZHANG, AND A.-M. KERMARREC. **Connectivity-based and Anchor-Free Localization in Large-Scale 2D/3D Sensor Networks**. In *Proc. of the 11th ACM MobiHoc*, pages 191–200, 2010. 18, 21, 26, 32
- [63] J. LUO, F. LI, AND Y. HE. **3DQS: Distributed Data Access in 3D Wireless Sensor Networks**. In *Proc. of the IEEE ICC*, pages 1–5, 2011. 18
- [64] B. JOE. **Three-Dimensional Triangulations from Local Transformations**. *SIAM J. Sci. Stat. Comput.*, **10**(4):718–741, 1989. 18
- [65] S. DUROCHER, D. KIRKPATRICK, AND L. NARAYANAN. **Guaranteed Delivery in Three-Dimensional Ad Hoc Wireless Networks**. In *Proc. of the 9th ICDCN*, pages 546–557, 2008. 19

- [66] S. KATZ, A. TAL, AND R. BASRI. **Direct Visibility of Point Sets**. In *Proc. of the 34th ACM SIGGRAPH*, 2007. 19, 22, 24, 34
- [67] J.H. HUBBARD AND B.B. HUBBARD. *Vector Calculus, Linear Algebra, and Differential Forms: A Unified Approach*. Matrix Editions, Ithaca, NY, 3 edition, 2007. 20
- [68] J. LUO, H.V. SHUKLA, AND J.-P. HUBAUX. **Non-Interactive Location Surveying for Sensor Networks with Mobility-Differentiated ToA**. In *Proc. of the 25th IEEE INFOCOM*, pages 1241–1252, 2006. 26
- [69] Y. SHANG AND W. RUML. **Improved MDS-Based Localization**. In *Proc. of the 23rd IEEE INFOCOM*, pages 2640–2651, 2004. 26, 29, 52, 85, 89
- [70] D. GOLDENBERG, P. BIHLER, M. CAO, J. FANG, B. ANDERSON, A.S. MORSE, AND Y.R. YANG. **Localization in Sparse Networks using Sweeps**. In *Proc. of the 12th ACM MobiCom*, pages 110–121, 2006. 27
- [71] T.H. CORMEN, C.E. LEISERSON, R.L. RIVEST, AND C. STEIN. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2 edition, 2001. 28, 126
- [72] X. GU, Y. WANG, T.F. CHAN, P.M. THOMPSON, AND S.-T. YAU. **Genus Zero Surface Conformal Mapping and Its Application to Brain Surface Mapping**. *IEEE Trans. on Medical Imaging*, **23**(8):949–958, 2004. 31
- [73] M. DO CARMO. *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976. 31
- [74] J. LUO AND Y. HE. **GeoQuorum: Load Balancing and Energy Efficient Data Access in Wireless Sensor Networks**. In *Proc. of the 30th IEEE INFOCOM (mini-conference)*, pages 616–620, 2011. 33
- [75] C. ZHANG, J. LUO, L. XIANG, F. LI, J. LIN, AND Y. HE. **Harmonic Quorum Systems: Data Management in 2D/3D Wireless Sensor Networks with Holes**. In *Proc. of the 9th IEEE SECON*, pages 1–9, 2012. 33
- [76] D. NICULESCU AND B. NATH. **Trajectory Based Forwarding and its Applications**. In *Proc. of the 9th ACM MobiCom*, pages 260–272, 2003. 33

-
- [77] **TinyOS Home Page:** <http://www.tinyos.net/>. 34
- [78] P. LEVIS, N. LEE, M. WELSH, AND D. CULLER. **TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications.** In *Proc. of the 1st ACM SenSys*, pages 126–137, 2003. 34, 63
- [79] S. TANG, X. LI, X. SHEN, J. ZHANG, G. DAI, AND S.K. DAS. **Cool: On Coverage with Solar-Powered Sensors.** In *Proc. of the 31st IEEE ICDCS*, pages 488–496, 2011. 42
- [80] D. L. HALL AND J. LLINAS. *Handbook of Multisensor Data Fusion*. CRC Press, 2001. 42
- [81] M. HEFEEDA AND M. BAGHERI. **Randomized k-Coverage Algorithms For Dense Sensor Networks.** In *Proc of the 26th IEEE INFOCOM*, pages 2376–2380, 2007. 42
- [82] B.A. BASH AND P.E.J. DESNOYERS. **Exact Distributed Voronoi Cell Computation in Sensor Networks.** In *Proc. of the 6th ACM/IEEE IPSN*, pages 236–243, 2007. 43
- [83] M.P. DO CARMO. *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976. 44, 47, 48
- [84] J. MCLURKIN AND E. DEMAINE. **A Distributed Boundary Detection Algorithm for Multi-robot Systems.** In *Proc. of IEEE/RSJ IROS*, pages 4791–4798, 2009. 45
- [85] M. YOUNIS, S. RAMASUBRAMANIAN, AND M. KRUNZ. **Location-Unaware Sensing Range Assignment in Sensor Networks.** In *Proc of the 6th IFIP Networking*, pages 120–131, 2007. 45
- [86] L. XIANG, J. LUO, AND A. VASILAKOS. **Compressed Data Aggregation for Energy Efficient Wireless Sensor Networks.** In *Proc. of the 8th IEEE SECON*, pages 46–54, 2011. 45
- [87] M.I. SHAMOS AND D. HOEY. **Closest-Point Problems.** In *Proc. of the 16th IEEE FOCS*, pages 151–162, 1975. 46

- [88] S. XIN AND G. WANG. **Improving Chen & Han's Algorithm on the Discrete Geodesic Problem.** *ACM Trans. on Graphics.*, **28**(4):1–8, 2009. 48, 49
- [89] E. WELZL. **Smallest Enclosing Disks (Balls and Ellipsoids).** *Results and New Trends in Computer Science (LNCS 555)*, pages 359–370, 1991. 52
- [90] D.T. LEE. **On k -Nearest Neighbor Voronoi Diagrams in a Plane.** *IEEE Trans. on Computer*, **31**(6):478–487, 1982. 52
- [91] K. HILDEBRANDT, K. POLTHIER, AND M. WARDETZKY. **On the convergence of metric and geometric properties of polyhedral surfaces.** *Geometriae Dedicata*, **123**(1):89–112, 2006. 59
- [92] H. KARCHER. **Riemannian Center of Mass and Mollifier Smoothing.** *Communications on Pure and Applied Mathematics.*, **30**(5):509–541, 1977. 61
- [93] J.-H. HAUER, V. HANDZISKI, AND A. WOLISZ. **Experimental Study of the Impact of WLAN Interference on IEEE 802.15.4 Body Area Networks.** In *Proc. of the 6th EWSN*, pages 17–32, 2009. 74
- [94] A. MISHRA, S. BANERJEE, AND W. ARBAUGH. **Weighted Coloring Based Channel Assignment for WLANs.** *ACM Mobile Computing and Communication Review*, **9**(3):19–31, 2005. 74
- [95] P. TRENKAMP, M. BECKER, AND C. GOERG. **Wireless Sensor Network Platforms – Datasheets versus Measurements.** In *Proc. of the 6th IEEE SenseApp*, pages 966–973, 2011. 77
- [96] S. LLOYD. **Least Square Quantization in PCM.** *IEEE Trans. on Information Theory*, **28**:129–137, 1982. 85, 88
- [97] D. BERTSEKAS. *Nonlinear Programming.* Athena Scientific, Belmont, Massachusetts, 2 edition, 1999. 90
- [98] C. BURNIKEL, K. MEHLHORN, AND S. SCHIRRA. **How to Compute the Voronoi Diagram of Line Segments: Theoretical and Experimental Results.** In *Proc. of the 2nd ESA (LNCS 855)*, pages 227–239, 1994. 90

- [99] OMPRAKASH GNAWALI, RODRIGO FONSECA, KYLE JAMIESON, DAVID MOSS, AND PHILIP LEVIS. **Collection Tree Protocol**. In *Proc. of the 7th ACM SenSys*, pages 1–14, 2009. 94, 101
- [100] P.-J. WAN, Z. WANG, Z. WAN S.C. HUANG, AND H. LIU. **Minimum-Latency Scheduling for Group Communication in Multi-channel Multihop Wireless Networks**. In *Proc. of the 4th WASA (LNCS 5682)*, pages 469–478, 2009. 94, 95, 101

REFERENCES

Appendix A

Proofs for the propositions of LBDP

Proof of Proposition 1: Let p_i be on the boundary of $\mathcal{P}$ according to either *Definition 1* or *Definition 2*, and let v be the concerned viewpoint. Without loss of generality, we may expand $\mathcal{N}_i$ to a nontrivial $\mathcal{P}' \subseteq \mathcal{P}$ such that $\mathcal{N}_i \subset \mathcal{P}'$, simply by adding one more point, i.e., $\mathcal{P}' = \mathcal{N}_i \cup \{p'\}$. There are two possible cases: p' lies either **on** or **not on** the line segment between v and p_i . As $p' \notin \mathbf{conv}(\mathcal{N}_i)$ (otherwise we would have $p' \in \mathcal{N}_i$, given the convexity of the transmission volume), we can avoid the former case by choosing v very close to the boundary of $\mathbf{conv}(\mathcal{N}_i)$. For the latter case, we simply let $v' = v$. According to the “blowing-up” feature of the transformation $f(\cdot)$ (which will be quantified by *Proposition 4*), there always exists an $f'(\cdot)$ such that $f'(p_i)$ lies on the convex hull of $f'(\mathcal{P}') \cup \{v'\}$, as far as there is no point in $\mathcal{P}'$ that lies on the line segment between v' and p_i .

In particular, if p_i is on the external boundary of $\mathcal{P}$, it is “visible” from some viewpoint v' . Therefore, we can directly expand $\mathcal{N}_i$ to $\mathcal{P}$ and be sure that no $p' \in \mathcal{P}$ is on the line segment between v' and p_i . Consequently, there exists an $f'(\cdot)$ such that $f'(p_i)$ lies on the convex hull of $f'(\mathcal{P}) \cup \{v'\}$. Conversely, the existence of an $f'(\cdot)$ such that $f'(p_i)$ lies on the convex hull of $f'(\mathcal{P}) \cup \{v'\}$ suggests that no $p' \in \mathcal{P}$ is on the line segment between v' and p_i . Since $v' \notin \mathbf{conv}(\mathcal{P})$, p_i has to be on the external boundary of $\mathcal{P}$. Q.E.D.

Proofs of Propositions 2 and 3: It is straightforward to verify that the transformation $f(\cdot)$ preserves surface convexity. In other words, if part of the surface is described

by a convex function, this convexity is preserved in the image of $f(\cdot)$. The fact that p_i is on the convex hull of $\mathcal{P}' \subseteq \mathcal{P}$ suggest that the surface through p_i is locally convex. Therefore, simply choosing a viewpoint v that faces that surface and applying a proper $f(\cdot)$ will lead to the claim made in *Proposition 2*.

It is also trivial to show that, if v, p_i, p'_i are collinear, then $v, p_i, p'_i, f(p_i), f(p'_i)$ are also collinear. Therefore, at most one of p_i and p'_i can be recognized as a boundary node, hence the claim made in *Proposition 3* follows. Q.E.D.

Proofs of Propositions 4 and 5: We only sketch the proof of *Proposition 4* by omitting the tedious trigonometric derivations. To derive the upper bound for d , we consider the worst case where the images of the three points become collinear, as shown in Fig. A.1(a), because further increasing d would compromise the convexity of $\check{p}_i$. Using basic trigonometry, we obtain the relation between d and $(r_i, r_j, r_k, \alpha_1, \alpha_2)$ as

$$d \leq \frac{r_i(r_j \sin \alpha_1 + r_k \sin \alpha_2) - r_j r_k \sin(\alpha_1 + \alpha_2)}{\sqrt{r_j^2 + r_k^2 - 2r_j r_k \cos(\alpha_1 + \alpha_2)}}.$$

Given $(r_j, r_k, \alpha_1, \alpha_2)$, d is increasing in r_i . Therefore, we can also bound d from above by bounding r_i . The key to subsequent derivations is: $\beta_1 + \beta_2 = \pi \Rightarrow \tan \beta_1 + \tan \beta_2 = 0$ in the worst case shown in Fig. A.1(a). Using $(R, r_j, r_k, \alpha_1, \alpha_2)$ to represent $(\tan \beta_1, \tan \beta_2)$, we have

$$r_i \leq 2R - \frac{(2R - r_j)(2R - r_k) \sin(\alpha_1 + \alpha_2)}{(2R - r_j) \sin \alpha_1 + (2R - r_k) \sin \alpha_2}.$$

The case where $\alpha_1 = \alpha_2$ and $r_j = r_k$ follows trivially.

According to differential geometry, at a point p of a differentiable surface S , two *principal curvatures* k_1 and k_2 determine the convexity of the surface at p . However, as the *principle directions* associated with these two curvatures are unknown in our case (we do not have the boundary surface), we need to sample three non-identical sectional planes to determine three sectional curvatures, by which we can uniquely determine the principal curvatures/directions and hence the surface convexity. Therefore, for boundary detection in 3D WSNs, we choose three non-identical planes containing v and p_i , with one arbitrarily chosen but the other two perpendicular to each other, and the aforementioned 2D conditions need to be satisfied in each of these three planes.

The statement that four transformations are sufficient to recognize a boundary point p_i follows from the sufficiency of “seeing” p_i from one of the four viewpoints. Let

Figure A.1: Illustrations for proofs. (a) Depth of Notch (DoN) in 2D – the extreme case, and (b) p_i is “visible” from at least one of the three vertices of an equilateral triangle enclosing $\text{conv}(\mathcal{N}_i)$.

us still use a 2D scenario to illustrate the idea in Fig. A.1(b). It is straightforward to see that, as far as the line-of-sight from v to p_i passes through the line segment (p_j, p_k) , p_i can be detected as a boundary node with a properly chosen transformation. However, for an arbitrary $\mathcal{N}_i$, as the orientation of (p_j, p_k) is unknown, the best choice is to put three view points at the vertices of an equilateral triangle enclosing $\text{conv}(\mathcal{N}_i)$. Consequently, as far as the angle $\angle p_j p_i p_k$ is larger than $2\pi/3$, at least one view point will have its line-of-sight towards p_i passing through (p_j, p_k) . It can be shown that $\frac{d}{\rho} < \frac{1}{2}$ is a sufficient condition of $\angle p_j p_i p_k > 2\pi/3$. Extending this idea to 3D, the view points should be put on the vertices of a regular tetrahedron enclosing $\text{conv}(\mathcal{N}_i)$, which leads to the same requirement on $\frac{d}{\rho}$ claimed in *Proposition 5* Q.E.D.

Proof of Proposition 6: For the k -th outer iteration, let $\vec{n}^k$ and $\vec{n}^k \cdot \vec{p}_i$ denote the plane used for the test, i.e., $\vec{n}^k \cdot \vec{m} = \vec{n}^k \cdot \vec{p}_i$, $\forall m \in \mathbb{R}^3$, and let $\check{\mathcal{N}}_i^k$ be the subset of $f(\mathcal{N}_i)$ that belongs to the convex hull of $\{v, m^1, \dots, m^k, \check{p}_i\}$, where m^k refers to the point chosen for the k -th iteration. We first show that $\{\check{\mathcal{N}}_i^k\}$ is a strictly increasing sequence.

Lemma 1. $\check{\mathcal{N}}_i^k \subset \check{\mathcal{N}}_i^{k+1}$.

Proof. Given the construction, it is clear that $\check{\mathcal{N}}_i^k \subseteq \check{\mathcal{N}}_i^{k+1}$. Assume, by contradiction, that $\check{\mathcal{N}}_i^k = \check{\mathcal{N}}_i^{k+1}$. This implies that $m^{k+1} \in \check{\mathcal{N}}_i^k$, and hence contradicts the fact that $\vec{m}^{k+1}$ is the largest among $f(\mathcal{N}_i)$ in terms of its dot product with $\vec{n}^k$. □

As $f(\mathcal{N}_i)$ is a finite set, LEMMA 1 shows that CVX-TEST-DC terminates in finite time regardless of the status of $\check{p}_i$. If $\check{p}_i$ is on the convex hull of $f(\mathcal{N}_i) \cup \{v\}$, the correctness of the algorithm is obvious. Otherwise the correctness is confirmed by the following result.

Lemma 2. *If neither m^{k-1} nor m^k can be replaced at the k -th iteration and the current plane is not supporting, $\check{p}_i$ is not on the convex hull of $f(\mathcal{N}_i) \cup \{v\}$.*

Proof. Let us consider the simplex induced by the set $\{v, m^{k-1}, m^k, m^{k+1}\}$ in Fig. A.2(b), where m^{k+1} is a candidate to be added for the next iteration. The assumption that m^{k+1} cannot replace either m^{k-1} or m^k implies that a plane determined by $\check{p}_i$ and any two points in $\{m^{k-1}, m^k, m^{k+1}\}$ separates the third from v . It is straightforward to show that this is possible only if $\check{p}_i$ is inside the simplex, hence proving $\check{p}_i$ is not on the convex hull of $f(\mathcal{N}_i) \cup \{v\}$. $\square$

Figure A.2: Illustration for proof of Lemma 2.

It is straightforward to see that CVX-TEST-DC is similar to *quicksort*, as the new point chosen in each iteration acts in a similar way as the *pivot* in quicksort. Moreover, CVX-TEST-DC only needs to “conquer” points in $\mathcal{N}_i \setminus \check{\mathcal{N}}_i^k$ after dividing. Therefore, we have the following recurrence for the time complexity $T(\eta)$, given totally η points in $f(\mathcal{N}_i)$:

$$T(\eta) = T(c\eta) + \mathcal{O}(\eta), \quad 0 < c < 1,$$

According to *master theorem* [71], the average-case complexity of CVX-TEST-DC is $\mathcal{O}(\eta \log \eta)$. Q.E.D.

It can be shown that the average-case of complexity of CVX-TEST-DC in 2D is only $\mathcal{O}(\log \eta)$, demonstrating again the non-trivialness of extending from 2D to 3D. Q.E.D.

Publications

JOURNAL ARTICLES

1. **F. Li**, J. Luo, W. Wang, and Y. He, “Autonomous Deployment for Load Balancing k -surface Coverage in Sensor Networks”, *IEEE Transactions on Wireless Communications* (to appear), 2014.
2. M. Zhang, **F. Li**, Y. He, J. Lin, X. Gu, and J. Luo “GRIP: Greedy Routing through dIstributed Parameterization for Guaranteed Delivery in 2D/3D Sensor Networks”, *Springer Wireless Networks* (to appear), 2014.
3. **F. Li**, C. Zhang, J. Luo, S. Xin, and Y. He, “LBDP: Localized Boundary Detection and Parameterization for 3D Sensor Networks,” *IEEE/ACM Trans. Networking*, Vol. 22, No. 2, pp 567-579, 2014.
4. **F. Li**, T. Xiang, J. Luo, G. Shi, and Y. He, “ART: Adaptive fRequency-Temporal co-existing of ZigBee and WiFi Networks”, under submission.

CONFERENCE PAPERS

5. C. Zhang, **F. Li**, J. Luo, Y. He. “iLocScan: Harnessing Multipath for Simultaneous Indoor Source Localization and Space Scanning”, in *Proc. of the 12th ACM SenSys*, 2014.
6. T. Xiang, Z. Chi, **F. Li**, J. Luo, L. Tang, L. Zhao, Y. Yang. “Powering Indoor Sensing with Airflows: A Trinity of Energy Harvesting, Synchronous Duty-Cycling, and Sensing”, in *Proc. of the 11th ACM SenSys*, 2013.

7. **F. Li**, T. Xiang, Z. Chi, J. Luo, L. Tang, L. Zhao, Y. Yang. “Demo Abstract: Powering Indoor Sensing with Airflows - A Trinity of Energy Harvesting, Synchronous Duty-Cycling, and Sensing”, in *Proc. of the 11th ACM SenSys*, 2013.
8. **F. Li**, J. Luo, G. Shi, Y. He. “FAVOR: Frequency Allocation for Versatile Occupancy of spectrum in Wireless Sensor Networks”, in *Proc. of the 14th ACM MobiHoc*, 2013
9. C. Zhang, J. Luo, L. Xiang, **F. Li**, J. Lin, and Y. He, “Harmonic Quorum Systems: Data Management in 2D/3D Wireless Sensor Networks with Holes,” in *Proc. of the 9th IEEE SECON*, 2012.
10. **F. Li**, J. Luo, S. Xin, W. Wang, Y. He. “LAACAD: Load bAlancing k-Area Coverage through Autonomous Deployment in Wireless Sensor Networks”, in *Proc. of the 32nd IEEE ICDCS*, 2012.
11. **F. Li**, J. Luo, C. Zhang, S. Xin, Y. He. “UNFOLD: UNiform Fast On-Line Boundary Detection for Dynamic 3D Wireless Sensor Networks”, in *Proc. of the 12th ACM MobiHoc*, 2011.
12. J. Luo, **F. Li**, and Y. He. “3DQS: Distributed Data Access in 3D Wireless Sensor Networks”, in *Proc. of the 12th IEEE ICC*, 2011.