COVERAGE AND CONNECTIVITY MANAGEMENT IN WIRELESS SENSOR NETWORKS

ZHANG MINGZE
B.Eng. (Hons.), NUS

A THESIS SUBMITTED

FOR THE DEGREE OF PH.D. IN COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2009

The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.

– Mark Weiser

Acknowledgement

I want to express my deeply-felt thanks to my Ph.D. supervisor, Dr. Mun Choon Chan, for his inspiring ideas, valuable suggestions and constant encouragement during the whole course of the work. Without him the work would not haven been possible. I am grateful to my co-supervisor, A/P Akkihebbal L. Ananda, for his thoughtful and important advice throughout this work.

I wish to express my special thanks to Dr. Vikram Srinivasan, Dr. Mehul Motani and Dr. Chen Khong Tham, for the wonderful course in sensor networks and their guidance on the course project.

I would also like to express my gratitude to all present and former members of Communication and Internet Research Lab, as well as my friends and classmates who helped me at different periods of my work. In particular, I would like to thank Mr. Binbin Chen and Shuai Hao, for the countless hours spent in setting up the sensor testbeds, as well as the interesting discussions on asymmetric links. I would like to thank Dr. Wei Wang and Mr. Kok Kiong Yap, from whom I learned a lot on research methodology. I would like to thank Mr. Xiuchao Wu for his patient helps in locating and using lab resources. I would also express special thanks to Dr. Sridhar K.N. Rao, Mr. Tao Shao, Mr. Feng Xiao, and Mr. Zhiguo Ge for their helps in many aspects of my work and my life.

My special thanks goes to my dear parents, who always support me and encourage me in my entire life. I would also like to thank all my friends who supported me in completion of my studies.

Lastly I would like to express my heartful thanks to my wife, Dr. Yuwen Pan. She helped me concentrate on completing this dissertation and encouraged and supported me during the whole course of this work.

Contents

Abstract

Both coverage and connectivity are the fundamental performance measures of the service provided by wireless sensor networks. Coverage represents how well the sensing goal of the network is accomplished, and connectivity represents how well the information can be delivered among the sensor nodes or to the central controller. Managing network coverage and connectivity is thus important in sensor networks. This thesis focuses on the coverage and connectivity management problem in wireless sensor networks. The coverage and connectivity management functions are classified into microscale management and macroscale management according to the geographical scale within which the sensor nodes collaborate.

This thesis first investigates several important coverage and connectivity management problems according to this categorization. In particular, for the microscale coverage and connectivity control problem, a Configurable Coverage Protocol (CCP) is proposed to control the "on" and "off" of the sensor nodes and meanwhile maintaining network coverage and connectivity. CCP is an efficient and lightweight protocol, in which each node makes decision based only on the collaboration between its local neighbors. Unlike existing protocols, CCP targets coverage of only α portion of the network, where α can be freely configured by the network administrators.

For the problem of microscale connectivity monitoring, a hashing based protocol (H^2CM) is proposed for efficient neighbor table collection. Collecting neighbor tables from individual sensor nodes are generally hard due to the high communication cost. By utilizing connectivity-based constraints and several hashing techniques, H^2CM allows the central controller to collect the neighbor tables from interested sensor nodes with very high probability, but with much lower communication cost.

Lastly, for macroscale topological hole detection and monitoring, a simple but powerful algorithm based on the connectivity changes of the sensor nodes is proposed. The algorithm first distributively elects the set of indicator nodes, and only the indicator nodes are required to send their information to the central controller. The location and size of the hole can be fairly accurately estimated using the information from only a few indicator nodes.

The thesis then integrates these individual management protocols and functions into

a unified coverage and connectivity management system, which allows the network administrators to monitor and control the network coverage and connectivity, from both microscale and macroscale level. The dependencies of these individual components are analyzed and system initialization and operation sequences are explained.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Wireless Sensor Networks

The technologies of semiconductors, wireless communications and computing have enjoyed rapid development in the twentieth century. Microprocessors, wireless radio transceivers and batteries have been greatly improved in terms of performance, size and price. This progress, together with the marked advances in the area of microsensors, has allowed the integration of automatic sensing, embedded computing and wireless networking, at low cost, to quickly become a reality.

Low-power and tiny sensor nodes, each empowered with the ability of sensing, computation and wireless communication, enable a broad range of applications. They are normally deployed on large scale over the geographic region of interest, and cooperate among themselves distributively for various sensing, tracking, and actuation tasks. The potential applications of these networked sensors are enormous: e.g., habitat monitoring, environmental monitoring, smart home and office, inventory tracking, precision agriculture, transportation, military, health care, and many more.

Wireless sensor networks (WSNs), consisting of hundreds and thousands of such smart sensor nodes, have received a lot of attention recently. During the past decade, many testbeds and commercial products have been built - bird habitat observations [66],

ocean water monitoring [2], avalanche rescue [70], and army weapon tracking [6], just to name a few. It is not hard to foresee that with further advances in technologies, networked tiny sensors will soon be integrating into people's everyday activities and transforming the way people understand and manage the environment. In fact, wireless sensor networks are considered to be one of the most important technologies that may revolutionize the world [34, 33, 83, 22].

The advantages of wireless sensors over traditional wired ones lie in their ability to perform wireless communication and distributed local processing. These sensor nodes can be easily deployed in many hard-to-reach or hazard locations that are inaccessible to wired sensors. The large-scale deployment of wireless sensor networks allows the sensor nodes to be placed closer to the phenomena being monitored and thus resulting in larger signal-to-noise ratio and higher possibility of line-of-sight sensing. On the other hand, distributed local processing among low-cost and densely-deployed sensors is not only a cheaper solution compared to expensive and sparsely-deployed wired sensors but also provides more accuracy and robustness.

However, despite the many benefits of wireless sensor networks, most sensor network applications encounter one or more of the following challenges.

- Sensor nodes are untethered and hence energy consumption is of critical importance. The limited bandwidth of wireless communications also creates additional barriers.

- Sensor nodes are deployed in an ad hoc manner and most of the protocols and algorithms used are distributed in nature.

- Sensor nodes often operate in a dynamic environment. They may fail at any time and the wireless links are time-varying.

- Computation, storage and memory efficiencies need to be carefully considered in many cases due to the size and cost requirements of sensor network applications.

- Different sensor network applications impose different requirements and constraints on the system design and it is not possible to have one unified structure that works for all.

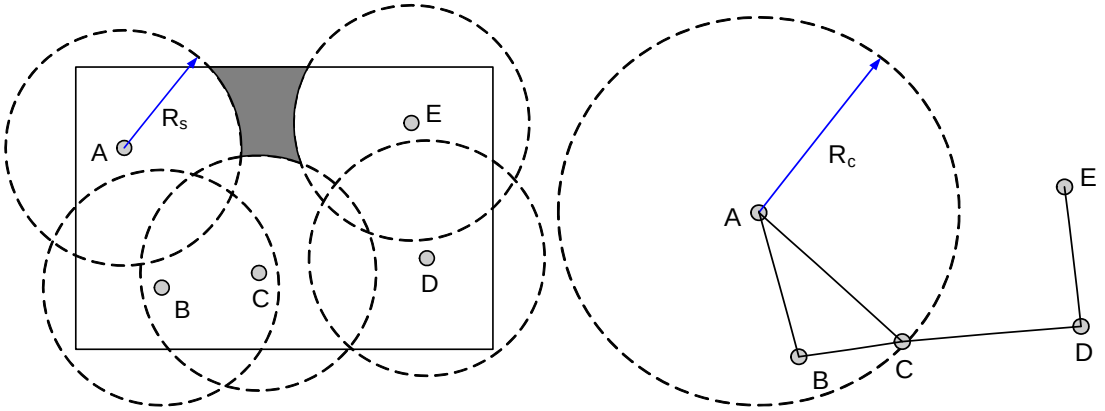
On one hand, wireless sensor networks have a bright future; on the other hand, there are a large number of technical challenges awaiting to be tackled. This has spurred tremendous research interest in sensor networks since the mid-1990s: ranging from physical layer to application layer, and from low level signal processing to high level security issues. This thesis focuses on two of the most important and fundamental research areas in wireless sensor networks, namely coverage and connectivity.

1.2 Coverage and Connectivity in WSNs

Coverage is a measure of the quality of service provided by a sensor network. Due to the attenuation of energy propagation, each sensor node has a sensing gradient, in which the accuracy and probability of sensing and detection attenuate as the distance to the node increases. The total coverage of the whole network can therefore be defined as the union (including possible cooperative signal processing) of all nodes' sensing gradients. It represents how well each point in the sensing field is covered. A coverage hole refers to a continuous area (or volume in 3-dimensional space) in the sensing field that is not covered by any sensor node, i.e., the events that occurred within a coverage hole cannot be sensed nor detected.

Figure 1.1(a) shows a coverage example where the sensing gradient of a sensor node is modeled as a binary disk. Every point within the sensing radius R_s of a sensor node is considered to be covered by the node. The union of all the disks forms the total coverage of the network. The region of interest is enclosed by a rectangle in the Figure. The shadowed region is not covered by any sensor node and thus considered to be a coverage hole.

Similarly, connectivity represents how well the sensor nodes in the network are "con-



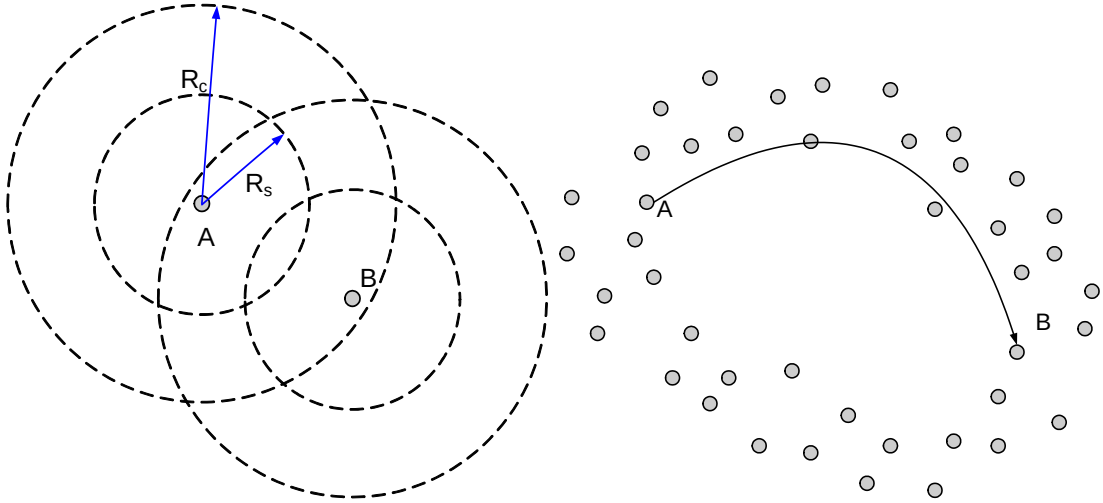
(a) Coverage and coverage hole (b) Connectivity graph

Figure 1.1: Illustrations of coverage and connectivity.

nected" to each other. It is a fundamental property of a wireless sensor network, for many upper-layer protocols and applications, such as distributed signal processing, data gathering and remote control, require the network to be connected. Since the sensor nodes communicate via wireless medium, a node can only directly talk to those that are in close proximity to itself (within its communication range). If a sensor network is modeled as a graph with sensor nodes as vertices and direct communication links between any two nodes as edges, by a connected network we mean the graph is connected.

Figure 1.1(b) shows the connectivity graph of the same set of nodes as in Figure 1.1(a). The communication model in this example is also a binary disk model where if the distance between two nodes is greater than the communication range R_c, they cannot talk to each other directly. Every node in Figure 1.1(b) can communicate with every other node, either directly or indirectly via some intermediate nodes. The network is thus connected.

Although coverage and connectivity have many differences, they are not unrelated. In fact, a covered network and a connected network are closely related due to their common requirement on the geographical placement of sensor nodes. A completely covered network requires that each point in the sensing region to be covered by at least one sensor node. This implies that the distance between a node and its closest neighbor cannot be



(a) Covered region and connected nodes (b) Coverage hole and topological hole

Figure 1.2: Relationship between coverage and connectivity.

larger than some threshold to avoid coverage holes. A similar implication can be drawn from a connected network.

Coverage is generally a stronger constraint on sensor node placement because it requires *every* point in the region to be covered by at least one node. If a region is "well" covered by a set of sensor nodes, these nodes are likely to be "well" connected if the communication radius is large enough. It is proven [99, 107] that with the binary disk sensing and communication models, if $R_c \geq 2R_s$, a completely covered network implies a connected network. On the contrary, connectivity does not imply coverage regardless the relationship between R_c and R_s. However, if the set of sensor nodes are "well" connected, the region where these connected nodes are deployed is also likely to be "well" covered by intuition.

The intuition behind this result can be explained using a simple example shown in Figure 1.2(a). A point that is just outside the sensing range of node A has to be covered by another node (node B in the example). This implies that the distance between A and B must be less than $2R_s$. The two nodes are then connected to each other if $R_c \geq 2R_s$. On the other hand, when node A and node B are connected, the region between A and B is likely to be well covered if the sensing range R_s is not too small.

The relationship between coverage and connectivity can also be understood in terms of coverage and topological holes. As defined previously, a coverage hole is a geographical region where events cannot be detected by any sensor node. On the other hand, a topological hole or a routing hole is a kind of connectivity anomaly which causes the routing path between two nodes to be unnecessarily long relative to their physical locations. Because both types of holes are created due to the lack of sensor nodes in the hole region [1], a coverage hole generally implies a topological hole in the same region, and vice versa (excluding boundary conditions). This is especially true when the size of hole is much larger than both the sensing and communication ranges.

An example is shown in Figure 1.2(b), where a topological hole is created in the area of interest. The messages from node *A* have to be routed along the boundary of the topological hole to reach node *B*. If the sensing range R_s is small compared to the size of the hole, the topological hole naturally implies a coverage hole in the same region. Similarly, a coverage hole implies a topological hole too.

1.3 Coverage and Connectivity Management

Due to the large variety of application requirements and physical parameters of sensor nodes, the problems involving coverage and connectivity are highly diverse. Taking coverage as an example, according to the different application objectives, coverage can be classified into point coverage, barrier coverage, and area coverage [15, 45]. Each of the classification can be further subclassified. Furthermore, each of the problem can be tackled from different angles according to assumptions like whether a centralized or distributed algorithm is required, the sensing and communication model used, and the availability and accuracy of localization.

It is generally difficult, if not impossible, to construct a single framework that solves all problems. This thesis focuses on the problem of area coverage and connectivity man-

[1] With the exception that the topological holes or routing holes can also be created due to obstacles.

agement, which is defined as the activities, methods and procedures to monitor and control the network sensing coverage (area coverage) and connectivity. It involves the functions of coverage and connectivity planning, monitoring and maintenance according to user needs.

Network management is by itself a broad topic. The network management functions are traditionally categorized into the well-known FCAPS (fault, configuration, accounting, performance and security) in ISO Telecommunications Management Network model. However, this categorization is defined for broad-sense network management and does not directly apply when the focus is narrowed down to coverage and connectivity management. In this work, the coverage and connectivity management functions are categorized into *microscale* management and *macroscale* management according to the geographical scale upon which the collaboration between sensor nodes takes place.

Microscale management controls network coverage and connectivity by monitoring and controlling each node's local coverage and connectivity. It only requires collaboration among the sensor nodes in close proximity (e.g., the 1-hop neighbors). Management tasks like local coverage and connectivity monitoring [27, 29], coverage control [107, 99], and topology control [88, 14] belong to this category. As opposed to microscale management, management in macroscale level involves collaboration of sensor nodes that are far away geographically. Management tasks like topological hole boundary detection and coverage hole boundary detection [100, 21] fall under this category.

The categorization of microscale and macroscale management is justified by the fact that coverage and connectivity problems can be investigated at both microscale level, where the focus is on the coverage and connectivity of individual components, and macroscale level, where the focus is on the coverage and connectivity over a large geographical scale. For example, collecting each sensor node's connectivity (neighbor table) information at the central controller belongs to the problem of microscale connectivity monitoring. While monitoring a large-scale topological hole belongs to the problem of macroscale connectivity monitoring.

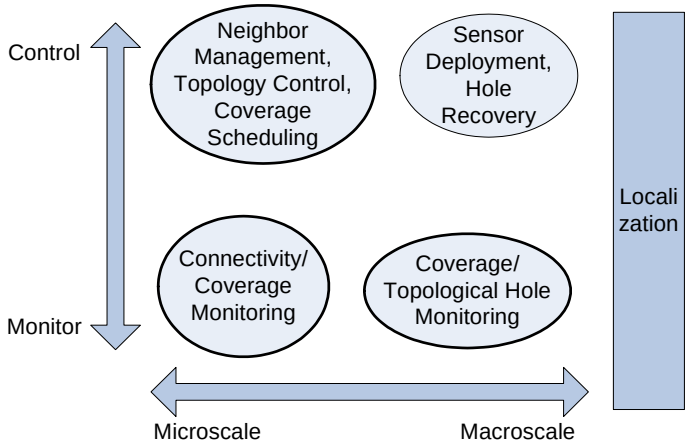


Figure 1.3: Coverage and connectivity management system.

The microscale management and macroscale management can be more precisely defined using the concept of OSI network model. Microscale coverage and connectivity management resides in data link layer and provides coverage and connectivity support for network layer protocols. On the other hand, macroscale coverage and connectivity management resides in application layer and provides coverage and connectivity support for other application layer protocols.

Figure 1.3 shows the general coverage and connectivity management architecture in sensor networks. It categorizes the coverage and connectivity management functions into four categories: microscale monitoring, microscale controlling, macroscale monitoring and macroscale controlling. The thesis mainly works on the problems in the first three categories, which are enclosed in bolded lines in the figure. Localization is an important property for coverage and connectivity management, for most problems involving coverage and connectivity require some form of localization support. This is also shown in Figure 1.3.

1.4 Problem Formulation and Thesis Contribution

This thesis addresses the following questions related to the coverage and connectivity monitoring and controlling at both microscale and macroscale levels.

1. How to control the sensor nodes' behavior such that the coverage and connectivity requirements are satisfied? Sensor nodes are normally over deployed in the sensing region to enhance system reliability. To save energy, only a partial collection of nodes need to be active at any particular time while maintaining the coverage and connectivity requirements. This problem belongs to the category of microscale coverage (area coverage) and connectivity control.

2. How to collect each sensor node's local connectivity information at the central controller? Collecting each sensor node's local connectivity (neighbor table) generally encounters very high communication cost. This is because each node's neighborhood information is normally very large and it has to be routed to the central controller via multiple hops periodically (for continuous connectivity monitoring). Thus, microscale connectivity monitoring at low communication cost is not a trivial problem and requires careful study.

3. How to detect and monitor the large-scale coverage or topological holes in sensor network? Large-scale coverage and topological holes can be naturally derived from microscale coverage and connectivity information collected at the central controller. However, if only macroscale information is required, solving it at the microscale level is generally not efficient. More efficient algorithms on large-scale hole detection and monitoring are needed. This problem belongs to the category of macroscale coverage and connectivity monitoring.

Note that simply solving these problems is not difficult, the challenges lie in the fact that the proposed solutions have to be efficient and scalable. Efficiency in sensor networks requires low communication overhead and low energy cost. This is an important measure due to the fact that the sensor nodes are untethered and powered by batteries. Scalability is also an important measure because of possibility of very large-scale deployments. In addition, distributed solutions are preferred in most scenarios rather than centralized ones to ensure resiliency.

This thesis systematically investigates these coverage and connectivity management problems. In particular, this thesis proposes:

1. A distributed node scheduling algorithm for microscale coverage and connectivity control. The proposed protocol relies on the distance estimates of the neighboring sensor nodes and does not require network localization. Unlike most existing research that works on complete coverage, the protocol works on partial coverage and the coverage objective can be configured by the network administrators.

2. An efficient way for partial or complete microscale connectivity collection. The problem is tackled by three components (vector distance, Bloom filters and signature hashing). By smart combination of these components, the network connectivity can be collected at different level of details with low communication cost. The proposed protocol is supported by the theoretical analysis on Bloom filters.

3. An efficient algorithm for large-scale hole detection, monitoring and estimation by observing the network connectivity changes. Based on the theoretical analysis on the geometric properties of holes, the holes can be detected and estimated using only a few indicator nodes, which requires very low communication cost.

All these proposed protocols are simple, lightweight and easy to implement, and they achieve the coverage and connectivity management objectives with much lower communication cost compared to existing protocols.

The thesis then integrates these proposed solutions into a unified coverage and connectivity management system, which allows the network administrators to monitor and control the network coverage and connectivity, at both microscale and macroscale levels [2]. The dependencies of these individual components are analyzed and system initialization and operation sequences are explained.

[2] This thesis only provides the conceptual design of the management architecture. The implementation of the management system is left for future work.

1.5 Thesis Organization

Chapter 2 briefly introduces various localization techniques with the main focus on two localization techniques: connectivity-based localization and sequentially distance-based localization, for the proposed unified coverage and connectivity management framework relies on these two techniques. The related work in coverage and connectivity monitoring and controlling, both in microscale and macroscale, is also given.

Chapter 3 presents the design of Configurable Coverage Protocol (CCP) – a node scheduling protocol for microscale coverage control. The goal of CCP is to schedule the on and off of the sensor nodes for energy saving while maintaining the network coverage and connectivity. CCP allows partial network coverage (with the configurable coverage parameter α) thus using a smaller number of active nodes compare to protocols that provide full coverage.

Chapter 4 presents H^2CM – a microscale connectivity monitoring protocol. H^2CM is an efficient way to encode the neighborhood information of each sensor nodes, such that the communication cost of microscale connectivity collection can be much reduced. H^2CM utilizes several methods under different situations for the optimal information collection.

Chapter 5 presents an efficient large-scale topological hole detection and monitoring protocol. The protocol relies on the information of maximum connectivity change in the network due to the formation of the hole to detect the hole and estimate its size. Note that although the protocol is targeted at topological holes, the results obtained can be regarded as coverage holes too if the hole sizes detected are large.

Chapter 6 presents a unified coverage and connectivity management framework, by integrating the previously proposed solutions. Conclusions and possible future work are shown in Chapter 7.

Chapter 2

Related Work

As mentioned in previous chapter, localization is an important property for coverage and connectivity management. In this chapter, various localization techniques will be briefly introduced first, with the main focus on two localization techniques: connectivity-based localization and sequentially distance-based localization. The unified coverage and connectivity management framework proposed in this thesis relies on these two localization techniques. The related work in coverage and connectivity monitoring and controlling, both at microscale and macroscale levels, will then be given.

2.1 Localization Techniques

Localization is the process of discovering the two-dimensional or three-dimensional positions of sensor nodes. It is an important property for coverage and connectivity management. Most problems involving coverage and connectivity, from microscale coverage control, to macroscale hole monitoring (e.g., knowing the hole location and size), require some form of localization. This section introduces a general background on the existing localization approaches, with the focus on two types of localization techniques: connectivity-based and sequential distance-based localization.

2.1.1 A Brief Summary on Localization Techniques

Various localization schemes can be classified into two categories in literature: range-based approaches and range-free approaches. Range-based approaches assume that the range information among the sensor nodes (e.g., distance and relative directions) is available, while range-free approaches do not require any range information.

Several hardware technologies provide the capability to measure the distance and relative directions between two sensor nodes. These technologies include Time of Arrival (TOA), Time Difference of Arrival (TDOA), Received Signal Strength (RSS) and Angle of Arrival (AOA). All these techniques estimate the distance or angle information among the sensor nodes with some hardware support. Localization algorithms based on TOA or TDOA, such as Global Positioning System [49] and the cricket system [80], normally have high accuracy. However, they all require expensive and energy-consuming devices and their accuracy also rely on the line-of-sight signal propagation. On the other hand, RSS and AOA [73] based techniques have relatively low accuracy, because they normally suffer from signal fading and Doppler effect. Recently, researchers have found that the techniques such as TOA, TDOA and AOA can achieve better accuracy in an ultra-wideband system over a normal wireless system [44].

Range-based localization methods assume that the sensor nodes are equipped with one or several of the ranging techniques introduced above. They can be mainly classified into two categories: the global localization algorithms and the sequential localization algorithms. The global localization algorithms localize all the sensor nodes simultaneously, either by relating the ranging information to some anchor nodes [1] [49, 80], or by some centralized computation using the collected ranging information among the sensor nodes [8, 55, 89, 64]. On the other hand, the sequential localization algorithms localizes the sensor nodes sequentially (and mostly distributively) using local ranging information [32, 7, 72, 73]

Rage-free localization methods are generally more cost-effective and lightweight than

[1] Anchor nodes are a small set of selected nodes whose locations are known.

range-based localization, due to the fact that they do not require any special hardware devices. The Centroid method [11] requires that the anchors have very large transmission ranges such that each node can hear from multiple anchors. The sensor nodes estimate their locations by calculating the center of all the anchors it can hear. APIT [46] lets each node estimate whether it resides inside or outside the triangles bounded by the anchors it can hear, and locations can be estimated by overlapping the triangle regions that a sensor node could possibly reside in. Embedding approaches [30, 52, 90] rely on various optimization techniques to centrally project the nodes to their geographical locations using only connectivity information. Connectivity-based methods [74, 62] utilize the hop count information to several anchors for sensor node localization.

Each localization algorithm has its own advantages and defects. Throughout the rest of this thesis, we only utilize the connectivity-based and distance-based localization methods.

2.1.2 Connectivity-based Localization

Connectivity-based localization algorithms only utilize connectivity information (e.g. hop count). They are lightweight and do not require extra hardware devices. Although they may have large localization errors, these errors do not cause significant impact on some applications such as connectivity monitoring (Chapter 4) and macroscale hole detection and monitoring (Chapter 5).

DV-hop [74] is probably the simplest connectivity-based localization method. The system contains some *anchor* nodes whose locations are known. Each node measures its hop counts to the anchors. DV-hop relies on the heuristic of proportionality between the distance and hop count in *isotropic* networks. The system firstly estimates the average distance-per-hop from anchor locations and the hop counts among the anchors. Each node then estimates its own distance to the anchors using the hop count information. The final location of each sensor node can be decided by trilateration [54]. The localization error of DV-hop can be in the scale of the sensor communication range R_c. However, such an

error is tolerable for applications such as monitoring a very large hole whose size is much larger than R_c.

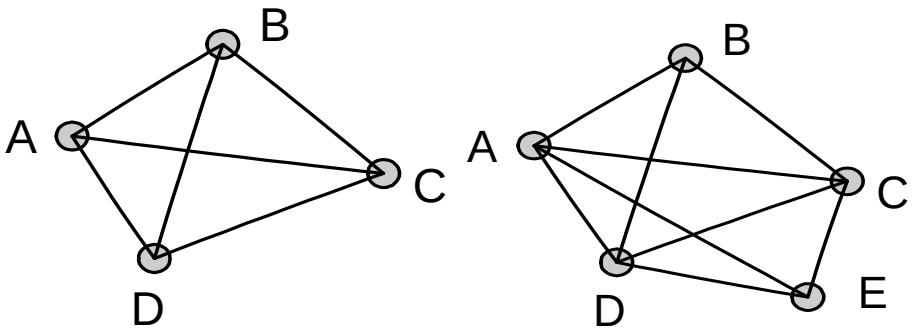
Rendered Path (REP) [62] is another connectivity-based localization algorithm. Unlike DV-hop, it mainly targets on the scenario of *anisotropic* networks where there is possibility of holes. In the presence of holes, the Euclidean distance between two sensor nodes may not be estimated using hop count because the shortest path between them can be curved by the intermediate holes and the proportionality assumption in DV-hop does not hold. REP solves this problem by constructing some virtual holes and rendering another path which routes around these virtual holes. By comparing the shortest path and rendered path between two nodes, the distance can be accurately estimated. The localization error of REP is only slightly higher than DV-hop algorithm.

2.1.3 Sequential Distance-based Localization

Connectivity-based approaches cannot support some applications which require a small localization error. For example, for the application of microscale coverage control (Chapter 3), the localization error shall be at least smaller than the sensing range R_s. Connectivity-based localization schemes have localization errors up to the range of R_c, which is normally larger than R_s. Under these circumstances, more accurate distance-based localization can be utilized.

While various distance estimation methods have been introduced in the previous section, this section focuses on sequential distance-based localization – how to distributively construct the location information of each sensor nodes from the (estimated) distance information among the neighbors.

In [72], Moore et al. propose a complete solution for such sequential localization when the distance estimation among the direct neighbors can have errors. The work is based on the notion of *robust quadrilateral*. Quadrilaterals are the smallest unit that can be unambiguously localized in isolation. Figure 2.1(a) shows a fully connected quadrilateral in which all the 6 pairwise distances between the four nodes are known. Such a



(a) A globally rigid quadrilateral. (b) Two quads sharing three vertices.

Figure 2.1: Globally rigid structures.

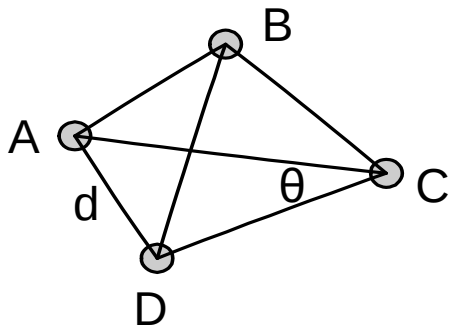


Figure 2.2: Robust quadrilateral.

quadrilateral is *globally rigid* [32], i.e., the relative positions of the four nodes are unique up to a global rotation, translation, and reflection. Two globally rigid quadrilaterals sharing three common vertices which forms a five-vertex graph is also globally rigid. This is shown in Figure 2.1(b), where two quads *ABCD* and *ACED* share the same vertices *A, C* and *D*.

However, the global rigidity does not guarantee a unique realization of graph when there are errors in distance estimation. It is proven in [72] that the graph realization is free of flip errors when

$$d \sin\theta > d_{\min}, \tag{2.1}$$

where d is the minimum distance out of the six known distances in a globally rigid quadrilateral, θ is the minimum angle explained below, and $d_{\min}$ is the threshold defined from

distance estimation errors. As shown in Figure 2.1.3, θ is the minimum internal angle for all the four triangles $\triangle ABC$, $\triangle ABD$, $\triangle ACD$ and $\triangle BCD$, and d is the minimum edge. Therefore, when a globally rigid quadrilateral also satisfies Equation 2.1, the probability of graph realization with no flip error is bounded. Such a quadrilateral is called *robust quadrilateral*. Based on the concept of robust quadrilateral, the neighboring nodes can sequentially estimate their relative locations using trilateration.

2.2 Related Work in Coverage and Connectivity

Both network coverage and connectivity are the fundamental performance measures of the service provided by wireless sensor networks. Coverage represents how well the sensing goal of the network is accomplished, and connectivity represents how well the information can be delivered among the sensor nodes or to the central controller. In this section, the state of the art in research related to coverage and connectivity is introduced. As illustrated in Chapter 1, the management of coverage and connectivity is mainly about monitoring and controlling, in both microscale level and macroscale level. The related work presented in this section is also categorized in this way.

2.2.1 Coverage and Connectivity Preserving Node Scheduling

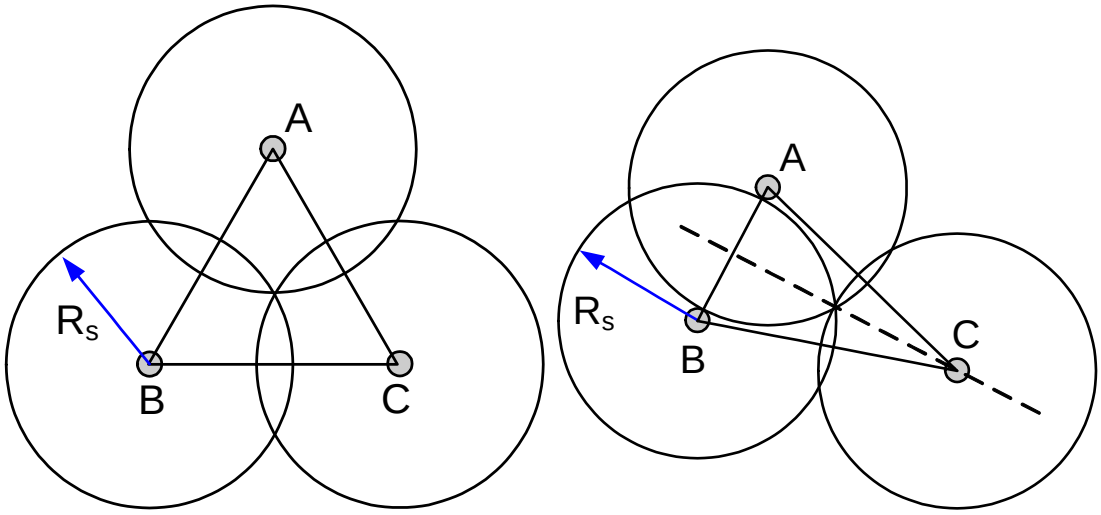
The aim of node scheduling is to select a minimum number of on-duty nodes that are active at any time, so that requirements on coverage, or connectivity, or both are still fulfilled. By doing so, the network energy cost can be minimized, and the network lifetime can be prolonged. These node scheduling problems are also sometimes regarded as density control problems. They control the “on” and ”off” of each sensor node (i.e., control the connectivity or topology of the network) to save energy, while maintaining the network microscale coverage or connectivity (or both). They are therefore categorized into the microscale coverage and connectivity control problems in this thesis.

GAF [101] divides a region into rectangular grids using location information, and

ensures that the maximum distance between any pair of nodes in adjacent grids is within the transmission range of each other. Only the leader in each grid stays awake. The leader election scheme in each grid takes the battery usage into account. The leaders form a dynamic routing backbone for packet forwarding. SPAN [19] adaptively decides whether a node should be working or sleeping based on connectivity among its neighbors. Only the selected coordinators are active to conserve energy. Some MAC layer protocols [95, 79, 104, 105] for wireless sensor networks also aim to maintain node sleep schedule. The nodes are dynamically woken up by the MAC protocols to create energy efficient network topologies.

In [94], Tian and Georganas proposed an algorithm that ensures the complete coverage using the concept of *sponsored area*. Whenever a sensor node receives a packet from one of its working neighbors, it calculates its sponsored area (defined as the maximal sector of the node's sensing circle covered by its neighbor's sensing circle). If the union of all the sponsored areas of a sensor node cover the sensing circle of the node, the node turns itself off. The sponsored area is only defined when the nodes are within sensing range of each other. The neighbors lying outside the sensing range are not considered although they can contribute to the node coverage. An improved version of [94] is proposed in [57]. The authors introduced the concept of effective neighbor nodes for calculating the node coverage accurately. Results in [57] show that the proposed protocol is able to outperform the protocol in [94] by about 30% in terms of reducing the actual number of nodes required for maintaining the original coverage.

Zhang and Hou [107] proposed the Optimal Geographic Density Control (OGDC) protocol based on certain optimality conditions of coverage and connectivity for large-scale sensor networks. The authors first proved that when communication range is at least two times the sensing range ($R_c \geq 2R_s$), a completely covered network guarantees connectivity. Thus, one can work on the optimal coverage problems without considering network connectivity. In OGDC, the sensor nodes decide whether they should turn on or off themselves distributively by observing whether they are at or close to the optimal



(a) Nodes *A*, *B* and *C* are free to move (b) Nodes *A* and *B* are fixed

Figure 2.3: Illustrations of the optimal node positions for minimum overlap in coverage.

positions for coverage. It defines the notion of the *crossing points* as the intersection points of the sensing circles of two nodes. To cover one crossing point of two nodes with minimum overlap, only one node should be used and the centers of the three nodes should form an equilateral triangle with side-length $\sqrt{3}R_s$, as illustrated in Figure 2.3(a). Furthermore, to cover one crossing point of two nodes whose positions are fixed, the third node has to be on the perpendicular bisector of the segment connecting the other two nodes, which is shown in Figure 2.3(b).

In [99], the authors introduced the close relationship between coverage and connectivity with the following theorems,

Theorem 2.1 *For a set of sensors that at least 1-cover a convex region A, the communication graph is connected if* $R_c \geq 2R_s$.

Theorem 2.2 *A set of nodes that k-cover a convex region A forms a k-connected communication graph if* $R_c \geq 2R_s$.

Theorem 2.3 *For a set of sensors that k-cover a convex region A, the interior connectivity is* $2k$ *if* $R_c \geq 2R_s$.

They then proposed the Coverage and Configuration Protocol (CCP) that configures the network for different degrees of coverage. For the case of $R_c < 2R_s$, the combination of CCP and SPAN [19] can provide both the network coverage and connectivity.

[51] describes a method to determine if an area is *k*-covered by checking the the perimeter of a sensing circle. An area is *k*-covered if and only if each sensor node in the network is *k*-perimeter-covered. The paper provides both algorithm for the binary disk sensing model (*k*-UC) and algorithm for non-disk sensing model (*k*-NC). The proposed method is extended to an algorithm that finds the set of nodes who provide *k*-coverage. *k*-UC and *k*-NC are centralized protocols.

Yan et al. [103] proposed a distributed density control algorithm capable of providing differentiated coverage based on different requirements in different areas of the network. Each node decides its own on-duty time by observing its neighbors' advertisement.

In [43], the authors analyzed the number of random sensing neighbors (nodes within sensing range) required for some confidence of redundancy of the current node, as well as the probability of complete redundancy based on the number of random sensing neighbors. This approach is based purely on random point processes (Poisson Point Process), it is also based on sponsored area (as in [94]) which may produce inefficient results.

In [53], the authors proposed a way to totally eliminate the communication cost of coverage calculation. This is a grid-based approach whereby only one node will be awake in each grid, and by doing so, nodes do not need to know the neighboring node information.

2.2.2 Other Coverage and Topology Control Protocols

Existing literature in node scheduling (or density control) algorithms for coverage and connectivity maintenance are summarized in the previous section. However, not all coverage and connectivity control protocols are based on density control. In this section, several other coverage and topology control problems are introduced.

In contrast to the static sensor networks, nodes in mobile sensor networks are capable

of moving in the sensing filed. Such networks are able to self-deploy themselves starting from an initial location configuration. The nodes would move around the area of interest such that coverage in the sensing field is maximized while the network connectivity is also maintained (and the moving distance shall also be minimized).

Wang et al. [98] proposed three distributed protocols for mobile sensors using Voronoi diagram: vector-based algorithm (VEC), Voronoi-based algorithm (VOR) and min-max algorithm (Minmax). VEC pushes the sensors from densely deployed areas to the sparsely deployed areas. Two sensors exert a repulsive force when they are close. VOR pulls the sensor nodes towards their local maximum coverage point. Each sensor node locally moves towards the farthest Voronoi vertex. The Minmax algorithm is similar to VOR. It moves each sensor node inside its Voronoi polygon to a point such that the distance from its farthest Voronoi vertex is minimized.

Potential field algorithms [50, 78] move the mobile nodes using the concept of potential field. Each node is subjected to two kinds of forces: F_{cover}, which causes the nodes to rebel from each other to increase the coverage and F_{degree}, which causes the nodes to attract each other to remain the necessary connectivity degree. Virtual force algorithms [109, 110] operate in a similar way. Each node is subjected to three kinds of forces: obstacles exert repulsive forces, areas of preferential coverage exert attractive forces, and other sensor nodes exert attractive or repulsive forces depending on the distance and orientation. The virtual force algorithm is a centralized one, where the computation is performed in a cluster head. In [48], the authors proposed the concept of electric force that depends on the internode distance and local current density μ_{curr}.

Bidding-based algorithm [97] is mainly targeted on the scenario where only partial of the sensor nodes are mobile. Each static node calculates its bid based on the distance to the farthest Voronoi vertex. It then finds the closest mobile node whose base price is lower than this bid. The mobile node considers all bids and service the highest bid among its neighboring static nodes.

The power-based topology control algorithms are to dynamically change the node

transmission power in order to maintain some property of the communication graph (mainly connectivity) and meanwhile the energy consumption for packet delivery is to be minimized. There are a lot of work in this area and only a few are listed here. [88, 14, 82, 65] try to optimize the transmission power levels so that the resulting topology is well connected. Under the total power minimization objective, topology control problems for many graph properties (e.g., connectedness, bounded diameter) are known to be NP-hard and approximation algorithms for many such problems have been developed [56, 14, 59].

There are a different set of coverage problems that work on the path exposure in the network. [68] defines a sensor coverage metric called surveillance that can be used as a measurement of quality of service provided by a particular sensor network. Centralized optimum algorithms that take polynomial time are proposed to evaluate paths that are best and least monitored in the sensor network. [67] further investigates the problem of how well a target can be monitored over a time period while it moves along an arbitrary path with an arbitrary velocity in a sensor network. Localized exposure-based coverage and location discovery algorithms are proposed in [69]. [96] investigates both minimal and maximal exposure path problems. It proves that maximal exposure path is NP-hard because it is equivalent to finding the longest path in an undirected weighted graph. It proposes several heuristics on this problem: random path heuristic, shortest-path heuristic, best-point heuristic and adjusted best-point heuristic.

2.2.3 Connectivity Monitoring

Connectivity monitoring is another important management tasks in sensor networks. The network connectivity information provides important support for various management functions such as debugging and root-cause analysis. In [81], Ramanathan et al. proposed a sensor network debugging system called Sympathy which requires connectivity information from the sensor nodes for root-cause cause analysis. The authors simply assume that each sensor node periodically sends its neighbor table to central controller.

Since the testbed on which they experiment is small, this is not a serious issue. In [85], the authors proposed a protocol that each node locally monitors its 1-hop neighbors and the neighborhood information aggregates along the path to the central controller. This approach utilizes the bitmap structure and is only applicable to a relatively small network.

In [28], the authors proposed TopDisc algorithm for sensor networks with its applications to network management. The idea of the algorithm is to find a set of distinguished nodes (minimum dominating set), using their neighborhood information to construct the approximate topology of the network. In graph theory, a *dominating set* for a graph $G = (V,E)$ is a subset D of V such that every vertex not in D is joined to at least one member of D by some edge. The problem of finding the *minimum dominating set* (MDS) is NP-complete. TopDisc is a heuristic algorithm for distributive MDS election based on the idea of node coloring. Only those nodes in MDS will reply back to the topology discovery probes, thereby reducing the communication overhead of the process.

STREAM [27, 29] is a multi-resolution topology retrieval protocol which makes a tradeoff between topology details and resources expended. The algorithm makes use of Minimal Virtual Dominating Set (MVDS) to define the distinguished nodes that will response the topology probes. The construction of MVDS relies on the concept of virtual radius, who defines a set of virtual neighbors that are within the virtual radius of each node. By adjusting the virtual radius, the MVDS of different resolution can be constructed, and the multi-resolution topology retrieval can be achieved.

In [18], the authors propose a mesh based topology retrieval algorithm with slow moving nodes in wireless ad hoc networks. Each node has multiple parents to which the local communicable neighbor information will be sent, and thus the algorithm is more error resilient.

The topology discovery algorithms mentioned above try to select a small percentage of the nodes who will respond to the topology discovery probes, and each of these nodes may only send partial neighborhood information to the central controller. Therefore, the total communication cost can be tradeoff for the accuracy of network topology informa-

tion. In [108], the problem of complete topology discovery is discussed, the work is based on the assumption that location information is available. The neighborhood pattern is also assumed to have strong correlation with the distance between a pair of the sensor nodes. By making use of these information, the cost of topology retrieval can be much reduced.

2.2.4 Macroscale Hole Recognition

Existing research in hole detection can be classified into four categories: sampling-based methods, statistical methods, geometric methods and topological methods.

Examples of sampling-based methods can be found in [42] and [91]. [42] presents an algorithm that continuously monitors a subset of the sensor nodes (samples) to detect large-scale event. When an event occurs, the sample nodes who detect the event will report to the central controller. The task is to estimate the event area by knowing which sample nodes detect the event. The detection algorithm requires the knowledge of the event geometry (e.g. circle or rectangle) for estimation of the event size and shape. This work also assumes that the location information of all the sensor nodes is known to the central controller and the set of samples has to be pre-computed in a centralized way to ensure best performance.

[91] presents a sampling method to detect and estimate straight line cuts in the network using sample nodes. By knowing which sample nodes have failed to send information to the central controller, the line that cuts the network can be estimated using some geometric properties. The location information is also assumed in this algorithm.

In [100, 39, 40], the problems of boundary detection using topological methods are investigated. In [100], Yue Wang et al. proposed an algorithm to detect the inner and outer boundary of holes by topological method. The boundary detection algorithm is motivated by an observation that holes in a sensor field create irregularities in hop count distances. Simply, the shortest path tree rooted at one node naturally “split” around the hole. The work identifies the “cut”: the set of nodes where shortest paths of distinct homotopy types terminate and touch each other, trapping the holes between them. The nodes in a cut can

be identified based on the fact that their common ancestor in the shortest path tree is fairly far away, at the other side of the hole. by removing different branches of the cut, multiple holes are virtually merged into one hole. The algorithm then refines the "coarse" boundary to recognize both inner and outer boundaries of the multiple holes.

In [39, 40], Stefan Funke et al. proposed to detect a boundary using the concept of isolevel. It observes that the end nodes of each isolevel in terms of hop counts to a root node are either on the inner boundary of the hole, or on the outer boundary. The protocol firstly builds the isolevel by grouping neighbors with same hop counts, and then distributively builds the shortest-path tree to a randomly selected node within each isolevel. The end points of the shortest-path trees are on the boundary. Although these algorithms [100, 40] are able to recognize the sensor nodes on boundary and they do not require any impractical information (e.g., location information or binary disk assumption of communication range), they generally involve a number of message flooding, thus generating a large amount of message exchanges. Moreover, these protocols have to be run periodically for dynamic hole detection.

Geometric methods are presented in [36, 58, 24]. In [36], Fang et al. identified the properties of *weak stuck node* and *strong stuck node*. All the stuck nodes must be on the boundary of the hole. These stuck nodes can be identified locally using only neighborhood information. This work assumes that accurate location information is known, and the communication model of the sensor nodes is the binary disk model. [58] assumes that connectivity information is available and the communication model is the quasi-binary disk model. By recognizing the structures of a "flower", a distributed algorithm on boundary node detection is proposed. [24] presents a hole-finding algorithm based on the fact that the shortest-path distance (in hop count) is larger than the distance between two nodes when the direct path between the two nodes is "cut" by the hole. By observing how much "longer" the shortest-path is compared to the distance, the hole information can be estimated. In these works, the boundary of holes can be detected distributively and locally. Although these algorithms can efficiently detect nodes on the boundary, accurate loca-

tion information is required and the communication model is normally considered to be binary-disk or quasi-binary-disk model. These requirements are practical.

Notice that for the boundary recognition algorithms [100, 40, 36, 58], even after the nodes on the boundary are locally identified, from management point of view, the central controller or network administrator is still unable to obtain "global" information of the hole (e.g., size or shape) until all or a subset of these nodes on boundary send information to the sink.

All the approaches described so far are proactive. Therefore, they may be executed even if no hole has been formed. Statistical based hole boundary detection methods [24, 38, 37] are based on the changes in node density or number of neighbors and can be considered as reactive. However, these protocols need to make assumptions on the node density and node placement distribution. In addition, all neighbors need to be maintained in the nodes' neighbor table. This is usually not the case, as the sensor nodes often only keep track of a small set of "good" neighbors in order to reduce communication and energy cost. All nodes on the boundary will also report changes detected, generating unnecessary overhead.

Finally, work in event boundary detection can also be found in [21, 75, 102, 92, 63]. These works try to detect and construct the boundary of sensing event (not the boundary of topological hole). Detecting event boundary is generally simpler than hole boundary because the sensor nodes on the edge of the event boundary can know that they are on the boundary based on their sensor readings. In [41, 47, 12, 93], efficient compression and representation of the event boundary are also studied.

Chapter 3

Coverage-Preserving Node Scheduling

3.1 Introduction

Low-cost sensor devices are failure-prone. In typical sensor networks, these devices are deployed in higher than necessary densities to meet various design specifications. In order to conserve energy and prolong network lifetime, at any time instant, only a portion of these sensors are required to be active while others operate in "sleep" or inactive mode. However, if too many nodes are turned off, coverage holes can be formed and the network can be disconnected. In this chapter, the problem of node scheduling for energy saving and meanwhile the network coverage is still preserved is investigated. A Configurable Coverage Protocol (CCP) is proposed.

CCP makes uses of the distance between two nodes rather than their actual locations. Distance information among nodes is easier to obtain than accurate global location information. In addition, CCP allows the trade-off between coverage and node usage (i.e., the number of active nodes). It can be configured to cover at least α portion of the area with high probability. For complete coverage ($\alpha = 1$), CCP is comparable to OGDC [107] in terms of coverage and number of active nodes required. Simulation shows that for 90% coverage, 22% node savings can be achieved. E.g., for the node density of 10, about 400 active nodes can support 90% coverage while about 530 active nodes are required to

support full coverage.

Setting the value of α allows the network administrator to flexibly control the number of active nodes in the network and the coverage level. For example, for a security monitoring scenario, the value of α can be set to 100% during night time and set to 80% or even smaller during day time.

The main aim of CCP is to schedule the "on" and "off" of the sensor nodes and preserve the microscale network coverage. The overall network coverage requirement can also be achieved if local coverage is preserved. CCP also implicitly maintains the network connectivity. Therefore, the work presented in this chapter serves the purpose of microscale coverage and (implicit) connectivity control. At last, one shall notice that the vacancy estimation scheme proposed in CCP also provides a way to compute the microscale vacancy of the given network, and thus can also serve as a management function for microscale coverage monitoring.

3.2 System Model

The sensor nodes are assumed to be deployed in high density over the whole area of interest, such that the network is completely connected and the area is fully covered. The sensing model is the binary disk model, i.e., each node has a sensing radius R_s and all points located within R_s of a sensor node are considered to be covered by the node.

Each node maintains the distance information to its direct neighbors. It can be built on top of the distance estimation scheme proposed in Section 3.7, nevertheless, it will also work with any other distance estimation schemes or absolute co-ordinate localization schemes as long as the error is constrained to be within a small potion of the sensing radius R_s. We do not assume any communication pattern in the chapter. However, note that the distance estimation methods in Section 3.7 assumes multi-energy-level binary disk communication model.

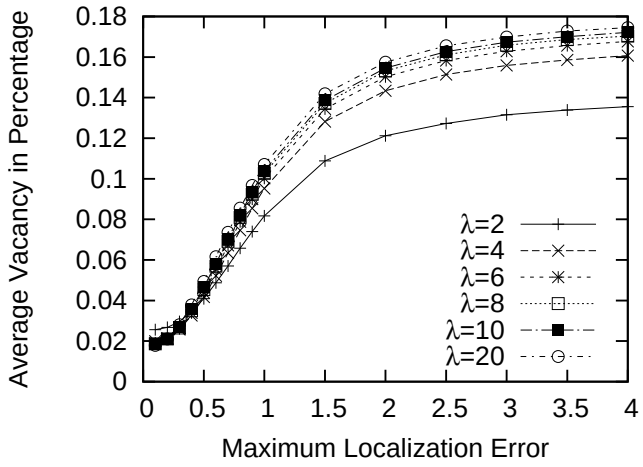


Figure 3.1: Average vacancy in percentage v.s. maximum localization error, with R_s normalized to 1

3.3 Effects of Localization Errors on Coverage

In most WSN coverage protocols, knowing the exact location of each sensor node is essential to determine how well the whole network is covered. However, accurate and low cost localization is still a big research challenge as discussed in Section 2.1. In fact, the accuracy of the localization scheme used is often determined by application requirements. Accurate location information normally requires extra computation, storage, communication and even hardware cost. In this section, we study the impact of localization errors on optimal coverage protocols, taking OGDC [107] as an example.

The model of localization error may vary depending on different localization algorithms and sensor operating environments. To keep the study simple, we define a simple circular uncertainty model: the location obtained by a localization algorithm is uniformly distributed in a circular region centered around the actual location of the node. The radius of the circular region is $R_{\max}$, which is also the maximum possible localization error.

Most coverage algorithms try to build a coverage set distributively such that minimum number of sensors are used to cover the entire region of interest. In this section, OGDC protocol is used to study the effect of location errors. Connectivity is not considered for simplicity.

Simulation results of coverage vacancy with respect to maximum localization error $R_{\max}$ for OGDC is shown in Figure 3.1. The simulated area is 30×30 unit length and R_s is normalized to 1. It can be seen that mean vacancy increases with localization error. When the maximum localization error $R_{\max} = R_s$, the vacancy is around 10% of total region to be monitored. When the maximum localization error reaches $2R_s$, the vacancy increases to around 15%. As the localization error increases further, the number of active nodes selected by the algorithm approaches that of a random selection.

Another interesting observation is that when there are localization errors, larger node density produces larger vacancy. This is due to the property of OGDC that uses the minimum number of sensors to cover the region. With the assumption of no location error, a larger node density means that nodes closer to optimal locations can be found. As a result, the algorithm will generate a sparser active node topology, and is therefore less tolerant to localization errors.

3.4 Overview of Configurable Coverage Protocol (CCP)

In this section, we present the configurable coverage protocol (CCP), which only makes use of the distance information among the neighboring nodes. CCP allows the users to specify the coverage objective α, in which at least α portion of the network is covered. In order to ensure that the coverage objective will be met, a way to compute or estimate the vacancy of the network in a distributed manner (with only distance information) is needed.

The approach used in CCP is shown in Figure 3.2. Given a set of active nodes, the area is divided into non-overlapping triangles (without considering boundary effects), and the vertices of these triangles are the active nodes. The vacancy is estimated within each of the triangle. For a large WSN, by ensuring that coverage objective is met locally (in each triangle), the global coverage which is computed as $1 - \frac{\sum V_j}{\sum T_j}$ will be satisfied too, where V_j is the vacancy in triangle j and T_j is the area of the triangle.

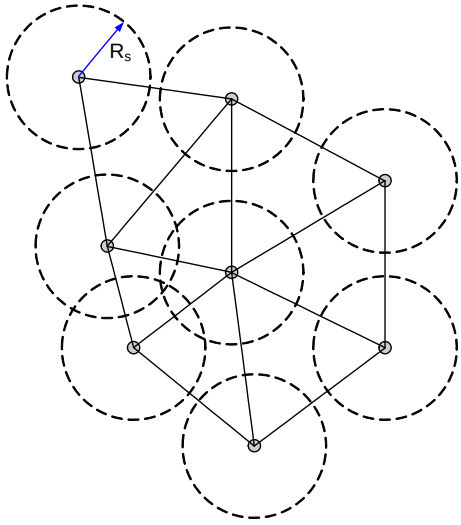


Figure 3.2: Illustration of coverage and vacancy estimation.

The basic idea of CCP is to sequentially select an additional node to be active such that the ratio of the size of the vacancy (V_j) inside the newly formed triangle to the area of the triangle (T_j) should be less than or equal to $1-\alpha$. In CCP, each node distributively elects itself based on the existing edges/triangles that have already been formed and the vacancy values of possible new triangles if it is active. Each node will start a timer based on the vacancy value of the new triangle formed by itself and existing edges, and once a node decides to be active, it will broadcast power on information first and other nodes will implicitly cancel their timers.

Note that in order to ensure the correctness of CCP, it is necessary that active nodes are added one at a time and this is built into the protocol design. By adding only one active node at a time, a unique sequence of active nodes is obtained. Such a sequence will generate a unique set of triangles formed by adding a new active node to two existing active nodes. This set of unique triangles covers the entire area of interest (excluding boundary effects) and the triangles do not overlap, ensuring that there is no double counting of vacant and covered area. The time complexity of the protocol will be linear to the number of nodes in the network. Because sensor nodes transmit packets in milliseconds, for a tens of thousands active node network, the total time of the whole process can be as

short as tens of seconds. For an even larger network, the process can start from multiple regions so that triangles are built up at different regions simultaneously.

3.4.1 Vacancy Inside Triangle

While the vacancy may be easily identified graphically or visually, computing the exact values of V_j using only distance information among nodes is more complicated. Before we formally describe CCP, it is essential to have a look at how the vacancies inside the triangles can be calculated.

Given the distances between each pair of the sensor nodes are d_1, d_2 and d_3, the area of the triangle is,

$$T(d_1, d_2, d_3) = \sqrt{s(s-d_1)(s-d_2)(s-d_3)}, \tag{3.1}$$

where $s = \frac{1}{2}(d_1 + d_2 + d_3)$. The common coverage between any pair of the nodes with distance d, where $d < 2R_s$, is given by

$$f(d) = 2R_s^2 \arccos(\frac{d}{2R_s}) - \frac{d}{2}\sqrt{4R_s^2 - d^2}. \tag{3.2}$$

The vacancy V of the several cases shown in Figure 3.3 can be calculated easily. The circle in the figures represent the sensing radius. The percentage of vacancy inside the triangle can then be evaluated by V/T.

However, for some other cases as will be listed in the next subsection, the vacancy is not in a simple form as shown in Figure 3.3. We call these exceptional cases. CCP tries to avoid such cases during selection of active nodes.

3.4.2 Exceptional Cases of Vacancy Calculation

Note that for the cases shown in Figure 3.3, the sensing nodes are in “good” positions where the angles of the triangle are “balanced”. These cases can be easily identified using the distance information and the vacancy inside the triangle calculated in a very simple

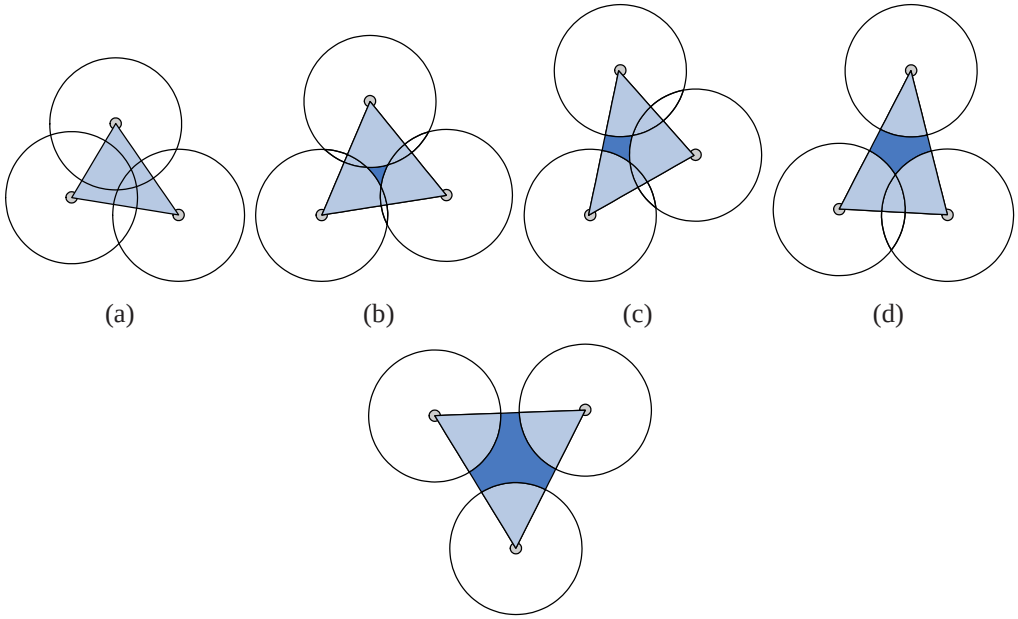


Figure 3.3: Triangle vacancy calculation. (a) $V = 0$ (b) $V = T - \frac{1}{2}\pi R_s^2 + \frac{1}{2}(f(d_1) + f(d_2) + f(d_3))$ (c) $V = T - \frac{1}{2}\pi R_s^2 + \frac{1}{2}(f(d_1) + f(d_2))$ (d) $V = T - \frac{1}{2}\pi R_s^2 + \frac{1}{2}f(d_1)$ (e) $V = T - \frac{1}{2}\pi R_s^2$

and standard way. Figure 3.4 shows the exceptional cases where the simple formula does not apply.

For exceptional cases (a), (b) and (c) shown in Figure 3.4, the problem comes from the fact that one edge of the triangle crosses all three circles. In addition, it can also be observed that in these cases, the angles inside the triangles are highly imbalanced. In cases (a) and (b), one of the angles is very large while in case (c), one of the angles is very

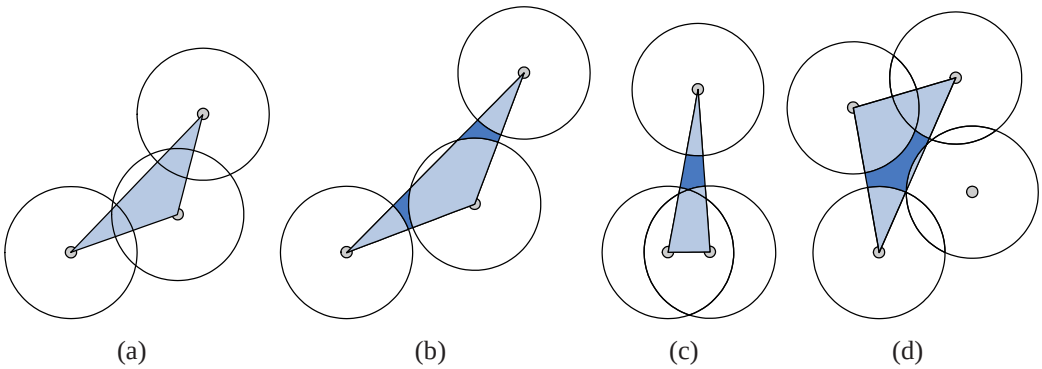


Figure 3.4: Exceptional cases of triangle vacancy calculation.

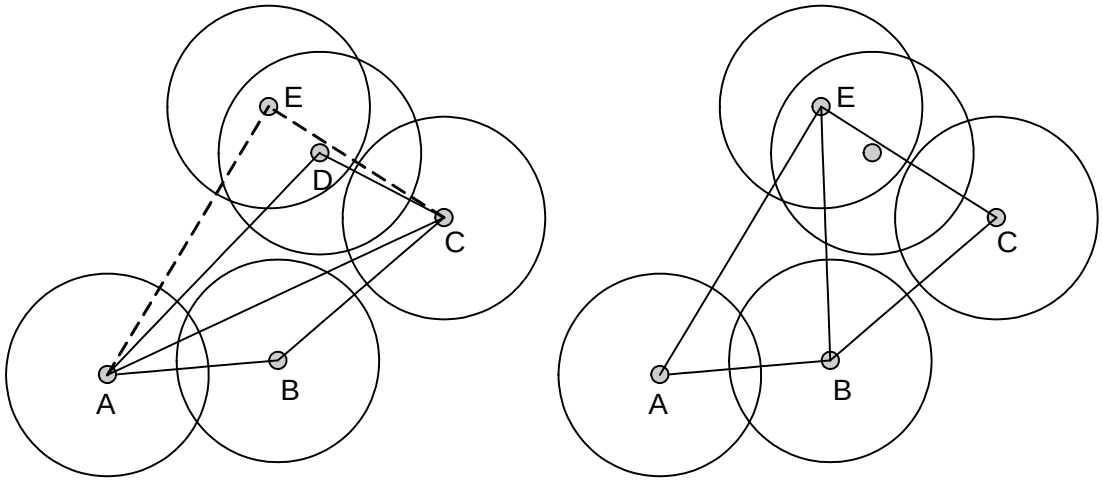
small.

In exceptional case (d) shown in Figure 3.4(d), the vacancy in the left triangle (shaded) is actually affected by one of the nodes in the other triangle on the right. The vacancy in the left triangle is smaller than the vacancy compute using the calculation stated in the previous section. In this situation, the vacancy is over estimated and the global objective α can still be satisfied. It can also be observed that case (d) is always linked to cases (a) and (b).

These exceptional cases cover all the exceptional possibilities. For example, by observing Figure 3.3(c), it can be seen that Figure 3.4(a) is the only possible exceptional case when there are two intersections among the three circles. These exceptional cases are not desirable. In particular, in cases (a) to (c), the vacancy is difficult to compute. In fact, we would further argue that these cases should also be avoided because they potentially increase the number of active nodes that are needed for the same coverage objective. The inefficiency of cases (a) and (b) can be explained using an example shown in Figure 3.5(a). Nodes *A* and *B* are known active nodes, if node *C* decides to be active because the vacancy in triangle *ABC* is smaller than the predefined value, then node *E* will not be selected based on the edge *AC* because there is a very large vacancy in triangle *EAC*. A node that is closer to edge *AC* has to be elected, which is node *D* in this example. On the other hand, as shown in Figure 3.5(b), if node *E* decides to be active based on edge *AB*, the final results will be triangle *ABE* and *BCE*, which is better than the example in Figure 3.5(a) because the former example tends to have more active nodes than the later one, even though in both cases, the average objective is met. Thus, when both *E* and *C* hears information about edge *AB*, *E* should elect itself first, as *C* is undesirable.

Case (c) only happens when node *A* and *B* are too close to each other. For a sufficiently high node density, case (c) is not likely to happen. It is also undesirable because the amount of redundancy is high.

In conclusion, in order to design an efficient distributed algorithm for configurable coverage, the exceptional cases should be avoided because they provide less efficient



(a) (b)

Figure 3.5: Illustration of inefficiency caused by exceptional cases a and b

solutions and the vacancy for these cases are hard to calculate. However, depending on the actual placement, it may not be possible to avoid these cases completely. Nevertheless, for most node densities of interest where complete coverage is possible, these cases are rare (from the simulations in Section 3.6). Hence, even when these cases are included and no vacancy is assumed (instead of computing the actual vacancy), the error is small.

3.4.3 Node Selection Constraint

As previously mentioned, in CCP, active nodes are added one at a time. In the new node selection process, the set of active sensor nodes must be connected at all times (connectivity constraint) and the exceptional cases analyzed in previous section shall be avoided as much as possible (angle constraint). Thus, during the selection process, nodes that satisfy both connectivity and angle constraints are considered first. If both constraints cannot be met at the same time, then connectivity constraint naturally takes priority over the angle constraint.

Connectivity Constraints

CCP tries to elect a subset of sensor nodes that cover α portion of the environment, it does not consider the connectivity of the network formed by the active sensor nodes. To maintain network connectivity in CCP, a node should only volunteer itself if it is able to communicate with both end vertices of the edge. Thus, each edge of the triangles is connected, and the whole network is then connected.

Angle Constraints

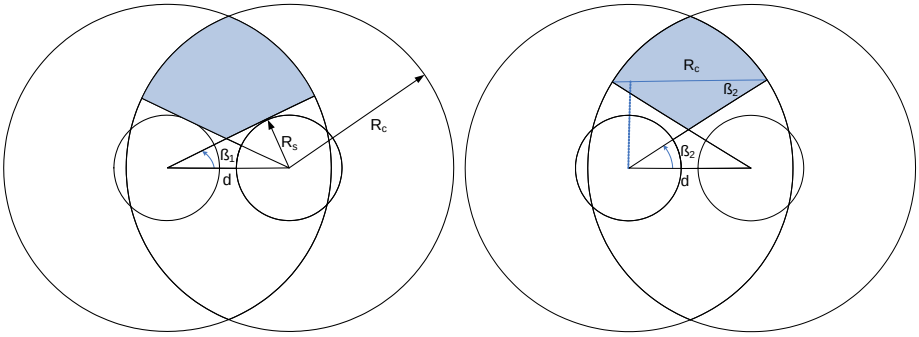
From observation, the exceptional cases in Figure 3.4 occur only when there are small (or large) angles inside the triangle. These small or large angles will cause imbalance in the length of edges, and thus may cause the imbalance in vacancies in adjacent triangles. In order to avoid the exceptional cases, small (or large) angles in the triangles should be avoided. Therefore, CCP selects the node not only based on the vacancy values inside the triangle, it also tries to select the triangle that maximizes the minimum angle. Note that this is different from the concept of Delaunay triangulation.

As discussed, the exceptional cases in Figure 3.4(a) and 3.4(b) are undesirable. For a dense network, it is better to eliminate all such possibilities to form a triangle of such cases, i.e., the nodes that will form exceptional triangles will not perform any action. As shown in Figure 3.6(a), considering the connectivity constraint and avoiding the exceptional cases, only the nodes in the shadowed area should compete for the active nodes. The minimum angles formed by the competing nodes and the edge should be β_1. Any node that has an angle smaller than β_1 will just ignore the new triangle and edge message.

The value of β_1 can be calculated by,

$$\beta_1 = \arcsin(\frac{R_s}{d}). \tag{3.3}$$

Note that the value of β_1 can be up to $\frac{\pi}{2}$ when d is close to 0. Thus, even when network is dense, such constraint shall not be performed when d is small.



(a) (b)

Figure 3.6: Angle constraints.

Another constraint in angle is shown in Figure 3.6(b). When a node decides to become active and form a new triangle, it will broadcast the power on message. All other nodes that are within the communication range of this node will hear this broadcast and try to cancel their timers. It is thus essential for every node that is trying to compete for the new vertex to hear this message. These nodes should be constrained in the shadowed area in Figure 3.6(b), in which every node is able to directly communicate with other nodes. The shadowed area is limited by angle β_2, which is calculated by,

$$\beta_2 = \arccos(\frac{R_c + d}{2R_c}). \tag{3.4}$$

Nodes that can form an angle larger than $max(\beta_1, \beta_2)$ meet the angle constraints and are preferred. For a sufficiently dense network, one or more nodes will be able to meet this angle constraints in most cases.

Rigidity Constraints

It is possible that given only distance information, the relative position of a node to some of its neighbors cannot be determined (i.e., the local distance graph is not rigid, the node can possibly be on either side of an edge), especially when there are errors in distance estimation. In CCP, any node who cannot form robust quadrilateral to existing triangles will not elect itself as an active node.

It should be mentioned that, the angle constraints in previous section also help in dealing with distance errors because maximizing the minimum angle is able to help the protocol tolerate more distance errors without affecting the robustness of the local distance graph [72].

3.5 CCP Details

In this section, we present the details of the CCP algorithm.

3.5.1 Selection of Starting Node

At the initial selection phase, all nodes are in the "UNDECIDED" state. A node should volunteer to be the starting node with probability p. The value of p should be a small value such that it is not likely to have many volunteer starting nodes in each round of selection.

When a node decides to be a starting node, it first waits for a random time t_s uniformly distributed within $[0, t_{smax}]$. t_{smax} can be any reasonably large values, for example, 20 times the maximum transmission time. This waiting time is used to reduce the probability of having multiple starting nodes but is not crucial for the correctness of CCP. If the node does not hear any messages from neighboring nodes within t_s, it will change its state to "ON" and broadcast the power on message. If it receives any power on messages from neighbor nodes, it will simply cancel the timer.

3.5.2 First Edge and First Triangle Formation

After the first starting node broadcasts the power on message, all neighbors around the starting node will set a timer t_1. If the timer fires, the node will change its state to "ON". The value of t_1 is based on the distance to the starting node d.

When a node turns "ON", it broadcasts power on message together with the edge information. The edge information includes the local unique id of the two end nodes as

well as the length of the edge.

Upon receiving the edge information, the neighboring nodes will set a timer t_2. If the timer fires, the node turns ”ON” and form the first triangle. The value of t_2 depends on the vacancy as well as the angles inside the triangle it forms.

The node will broadcast the power on message together with the triangle information. The information includes the id of the three vertices and the length of the three edges. This message also has information about the new edges generated by this triangle (there are normally two new edges). All nodes will save the triangles formed associated with itself (i.e. if a node is a vertex of the triangle, it will save this triangle information). All nodes that hear the triangle information and locate at the same side with the broadcasting node will cancel their timers.

3.5.3 Node Selection Process

Upon receiving the triangle and new edge message, only those nodes that are located at different sides of the new edge with the triangle will perform actions. Each node will first examine whether it has any triangle associated with itself and share a common vertex with the new edge. If there is, it will then look at the edge connecting itself and the common vertex, to see whether the edge has two triangles associated with it. The node will take no action if there are already two triangles associated with this edge. If there is only one triangle associated with the edge, and it satisfies the vacancy requirement, it will announce the creation of a new triangle with only one new edge immediately. This approach always tries to close the region around the common node first.

Otherwise, all other nodes set timer t_2 based on the vacancy and angles to the new edges. The node that fires first turns itself ”ON” and announce the existence of a new triangle with two new edges. All nodes that hear the new triangle information will cancel their timer t_2. Based on the triangle information broadcast by its neighbors, when a node notices that it is within one of the triangles formed, it turns itself ”OFF”. The protocol terminates when all nodes are either in the ”ON” or ”OFF” states.

3.5.4 Discussions

Starting Node Probability p

The value of p should be small enough so that in the ideal case, only one node in the whole network becomes starting node. This can be a value of say $\frac{1}{N}$.

Timer t_1

The timer t_1 should be based on the distance to the initial startup node. Based on the heuristics used in CCP, the optimal distance should be the edge length of the equilateral triangle which exactly has vacancy of $1-\alpha$.

The value of t_1 is then calculated by $t_1 = a(d_o - d)$ if $d < d_o$, and $t_1 = a(d - d_o) + c$ if $d > d_o$, where a, c are constants and c is used to degrade the distances that are larger than optimal (which may cause more vacancy).

Timer t_2

The value of t_2 can be calculated by the vacancy, as well as the minimum angle. The value of t_2 is computed as $a|\frac{V}{A} - \alpha| + \frac{b}{\min(a_1,a_2,a_3)} + c$, where a_1, a_2, a_3 are the angles of the triangle, a, b and c are constants. c is the penalty for the nodes that have vacancy larger than predefined value. It is 0 for the nodes that have vacancy smaller than predefined value.

Joint of Different Sets of Sensor Nodes

The above protocol description only considers the situation that there is only one starting node. Once there are more than one starting nodes, if there are no special considerations, most probably there will be multiple sets of active sensors at the end of the algorithm.

When a node hears a broadcast of triangle message from another set of sensor nodes (differentiated by the ID defined by the starting node), it will consider the joining of the

new edges associated with itself and the new edges associated with the triangle if any of the possible triangles satisfy the vacancy requirements.

3.6 Performance Evaluation

3.6.1 Simulation Setup

In all sets of simulations, we normalize the sensing radius R_s to be 1. The communication range R_c is set to be 3. The world size is a 30×30 square. The communication range is set to 3 times larger than sensing range so that the CCP is able to select the nodes that leave some vacancy. The values of R_c and R_s vary significantly for different sensor nodes and different applications, however, in a typical network, wireless communication ranges is generally several times larger than sensing range. We set $a = b = 0.5$ for CCP as the weights of vacancy and angle constraint respectively in all simulations.

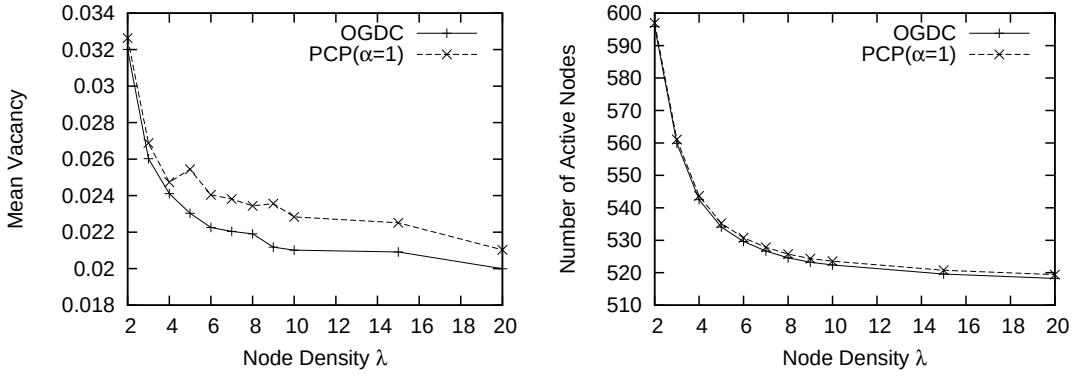
The relative localization scheme in the simulation assumes that the nodes are able to dynamically change the transmission power levels. Two power levels are used to estimate the distances, one is with $R_c = 2$ and one is $R_c = 1$. Note that the value of $R_c = 3$ is used for CCP packet transmission, it is not used in distance estimation.

The performance matrices are the average vacancy and the number of active nodes to monitor the environments after applying CCP.

3.6.2 Performance of CCP and OGDC

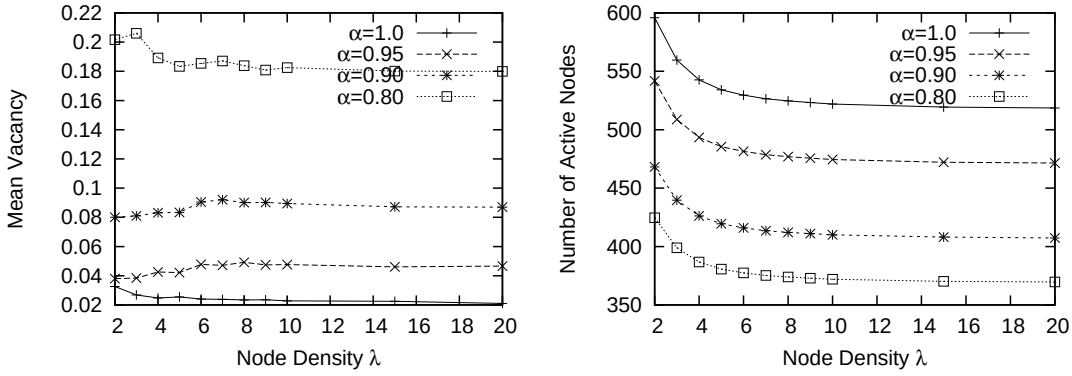
In the first set of experiments, we compare the performance of CCP and OGDC with both algorithms using the same distance estimate obtained using the scheme described in Section 3.7. To make CCP comparable to OGDC, we set the coverage objective α to 1. In addition, we modify OGDC protocol to use distance information rather than position.

The simulation results are shown in Figure 3.7. It can be observed that CCP with $\alpha = 1$ has very similar performance to OGDC. Overall, OGDC has a slightly better per-



(a) (b)

Figure 3.7: Comparison between OGDC and CCP



(a) (b)

Figure 3.8: CCP with Coverage Objective $\alpha = 1, 0.95, 0.9, 0.8$

formance because CCP does not try to minimize node redundancy but simply tries to select the nodes that leave no vacancy and satisfy the angle constraints. However, the performance degradation is small. Using the same distance estimates, the vacancy achieved by OGDC is less than 0.2% lower and the number of nodes needed is reduced by less than 1%.

It is interesting to note that for both OGDC and CCP ($\alpha = 1$), there is always some amount of uncovered area (about $2\% - 3\%$) in the network. The vacancy is a result of the distance estimation error. In addition, when node density is low, the amount of vacancy increases due to insufficient coverage. Therefore, in the presence of location or distance estimation error, it may not be meaningful to demand complete coverage even when net-

work density is high. In our simulation setup, only coverage objective of 0.98 or below can be achieved for both OGDC and CCP.

3.6.3 Performance of CCP with $\alpha < 1$

In the second set of experiments, we evaluate the performance of CCP if the objective α is set to a value less than 1. The results for $\alpha = 1, 0.95, 0.9$ and 0.8 are shown in Figure 3.8. We have observed that for values of α between 0.98 and 1.0, there is little difference in terms of average vacancy and number of active nodes needed. As a result, they are not shown in Figure 3.8.

From Figure 3.8(a), we can see that CCP is able to meet the coverage objectives most of the time. There are two reasons why the objective may not be met. First, the network density is too low and there are insufficient nodes. Second, due to distance estimation errors. Nevertheless, it can be observed that even when $\lambda = 2$ and the distance error is about $0.1R_c$, the mean vacancy is still very close to the objective.

In Figure 3.8(b), when α is decreased from 1.0 to 0.95, the number of active nodes required is about 91% of the total nodes required when $\alpha = 1$. The decrease in nodes required for α values of 0.9 and 0.8 are 22% and 29% respectively. The results can be explained as follow. When α is decreased to 95% the savings (9%) is limited by the number of nodes that contribute less than 5% of additional normalized coverage. The biggest savings (12%) comes from moving from 95% to 90% coverage when many more redundant nodes can be found. However, when coverage objective is further decreased to 80%, the amount of redundancy is already low and further savings is only 7%. Further reduction in coverage objective will not be an effective way to reduce the number of nodes required.

3.7 Neighbor Node Distance Estimation

As shown in the previous section, distributed distance-based localization relies on the distance estimates among the neighboring sensor nodes. In this section, we propose a simple distance estimation algorithm which can provide enough accuracy to support microscale coverage and connectivity control that will be presented in Chapter 3. In fact, for microscale coverage and connectivity control, global localization is not required. Distance estimations among the neighbors are sufficient because they provide local rigidity such that the relative locations among neighboring nodes can be determined. In this section, we will present an algorithm to perform distance estimation, using only connectivity information.

3.7.1 Assumptions and Notations

We assume that the sensor nodes are randomly distributed in a large 2-dimensional region with density λ. Thus, node distribution can be estimated as a Poisson point process. The uniform binary disk communication model is assumed. All the sensor nodes have the same communication range R_c.

The binary disk communication model is generally not true in real world. Therefore, algorithms proposed based on such an assumption may not work in practice. However, this problem is not too severe for the application of distance estimation. We will show via simulation in the later section that the binary disk model still works for distance estimation when the communication range of a sensor node is not a perfect circle.

We use capital letter such as A to represent a region of interest, and N_A is the random variable for the number of nodes inside region A. n_A represents the actual number of nodes inside regions A. When the context is clear, A will also be used to represent the area of a region.

3.7.2 Basic Idea and Problem Formulation

For binary disk communication model, the common area covered by the communication circles of two neighbor nodes has a one-to-one correspondence to the distance between the two nodes. Therefore, in order to estimate the distance, it is equivalent to estimating the common area covered by the communication circles of the two neighbor nodes.

Intuitively, given the fixed node density λ, the larger a region is, the more likely nodes will be located inside the region. Conversely, given a region containing n nodes, the larger the value of n is, the larger the area is likely to be. The above statements are supported by the following two facts.

Fact 3.1 *For a Poisson point process with node density λ, the probability that N_A nodes locate inside an region A (with area A) follows Poisson distribution.*

$$p(N_A = n) = \frac{e^{-\lambda A}(\lambda A)^n}{n!} \tag{3.5}$$

and $E(N_A) = \lambda A$, $V(N_A) = \lambda A$.

Fact 3.2 *For a Poisson point process with node density λ, the area of the region A_n that exactly contains n nodes follows Gamma distribution.*

$$f(A_n = A) = \frac{\lambda e^{-\lambda A}(\lambda A)^{n-1}}{(n-1)!} \tag{3.6}$$

and $E(A_n) = n/\lambda$, $V(A_n) = n/\lambda^2$.

Thus, for each sensor node, it can make use of the local communication graph (number of common communicable neighbors between two neighboring nodes) to estimate the common communication areas and then the distances between itself and its neighboring nodes.

The basic idea can be explained as follow. In Figure 3.9, the distance d between two nodes A and B is to be estimated. Let A and B be the region of communication circles of

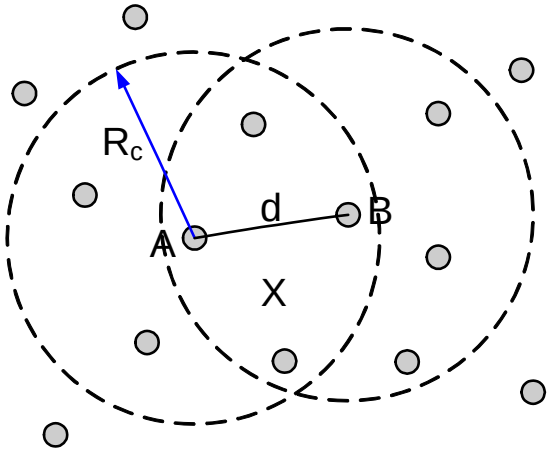


Figure 3.9: The number of common neighbors of two nodes can be used to estimate the distance between the two nodes.

node A and node B respectively. The common region of region A and B is X. Also, let there be n_a nodes in A, n_b nodes in B, and n_x nodes in X. Intuitively, when d is small, n_x is large and n_a and n_b are small. Conversely, when d is large, n_x is small, while n_a and n_b are large. Hence, by taking into account the values of n_a, n_b and n_x, d can be estimated.

As n_a, n_b and n_x are correlated, the problem can be redefined as follow. Given $(n_a - n_x)$ nodes in $A - X$, $(n_b - n_x)$ nodes in $B - X$, and n_x nodes in X, what is the estimated distance d between node A and node B? In the following analysis, we let m, n, and c denote $(n_a - n_x)$, $(n_b - n_x)$, and n_x respectively to simplify the expressions.

3.7.3 Maximum Likelihood Distance Estimation

Maximum likelihood estimation is used to estimate the size of X and thus the distance d. The probability of having a certain number of nodes inside an area given the value of the area is given in Equation 3.5.

We need to find the value of X which maximizes (let $a = \lambda A = \lambda B$ and $t = \lambda X$),

$$
\begin{aligned}
& M = p(c|X)p(m|A-X)p(n|B-X) \\
= \quad & \frac{e^{-\lambda X}(\lambda X)^c}{c!}\frac{e^{-\lambda(A-X)}(\lambda(A-X))^m}{m!}\frac{e^{-\lambda(B-X)}(\lambda(B-X))^n}{n!} \\
= \quad & \frac{e^{-2a}e^t t^c (a-t)^{m+n}}{c!m!n!}
\end{aligned} \tag{3.7}
$$

Maximizing Equation 3.7 is same as maximizing the value of $\ln M$,

$$
\ln M = -2a + t + c\ln t + (m+n)\ln(a-t) - \ln(m!n!c!) \tag{3.8}
$$

Let $\frac{d\ln M}{dt} = 0$ we have,

$$
\frac{d\ln M}{dt} = 1 + \frac{c}{t} - \frac{m+n}{a-t} = 0 \tag{3.9}
$$

Solving the above equation we get,

$$
X = \frac{-(m+n+c-a) + \sqrt{(m+n+c-a)^2 + 4ac}}{2\lambda} \tag{3.10}
$$

If $X < X_{min}$, we can set $X = X_{min}$, and if $X > X_{max}$, we can set $X = X_{max}$. Where $X_{min} = (\frac{2\pi}{3} - \frac{\sqrt{3}}{2})R_c^2$, and $X_{max} = \pi R_c^2$.

Results obtained using Equation 3.10 turn out to be fairly inaccurate when node density is low. This is because the number of nodes within communication range is too small to provide good accuracy, though the accuracy is much better for high node density. The approach taken to improve the estimation accuracy is to increase the number of samples through the use of multiple transmission power levels. By varying the transmission power, the sensor nodes can communicate with different sets of neighbors. This additional information helps to improve the estimation accuracy.

Take an example of two power level sensor nodes, as shown in Figure 3.10. The two sensor nodes have 2 communication radii R_{c1} and R_{c2} ($R_{c1} \leq R_{c2}$), and communication

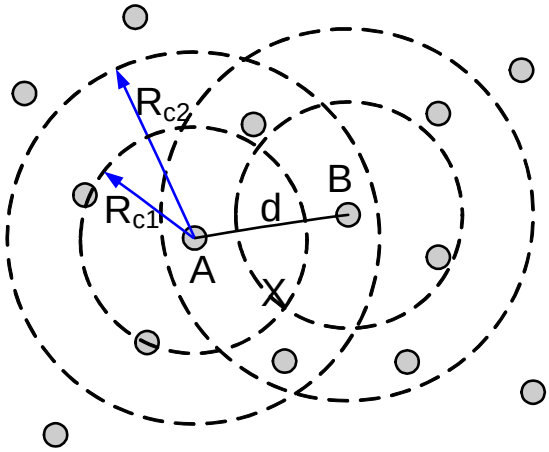


Figure 3.10: Distance estimation based on 2 transmission power levels

covered areas by the two power levels are $A_1 = B_1 = \pi R_{c1}^2$, and $A_2 = B_2 = \pi R_{c2}^2$. By adjusting the power levels, there are 4 combination of estimations, A_1 with B_1, A_2 with B_2, A_1 with B_2, and lastly A_2 with B_1. For the case of A_1, B_1 and A_2, B_2, the maximum likelihood estimation proposed above still works.

For the two cases with different communication radii, again, the maximum likelihood estimation method can be used.

$$\begin{aligned} & M = p(c|X)p(m|(A_1 - X)p(n|A_2 - X) \\ = & \frac{e^{-\lambda X}(\lambda X)^c}{c!}\frac{e^{-\lambda(A_1-X)}(\lambda(A_1-X))^m}{m!}\frac{e^{-\lambda(A_2-X)}(\lambda(A_2-X))^n}{n!} \\ = & \frac{e^{-\lambda(A_1+A_2)}e^{\lambda X}(\lambda X)^c(\lambda A_1 - \lambda X)^m(\lambda A_2 - \lambda X)^n}{c!m!n!} \end{aligned} \tag{3.11}$$

To find the maximum value of M ($t = \lambda X$, $a = \lambda A_1$, and $b = \lambda A_2$), let

$$\ln M = -(a+b) + t + c\ln t + m\ln(a-t) + n\ln(b-t) - \ln(c!m!n!) \tag{3.12}$$

Let $\frac{d\ln M}{dt} = 1 + \frac{c}{t} + \frac{m}{t-a} + \frac{n}{t-b} = 0$, we get

$$t^3 + (m+n+c-a-b)t^2 + (ab - ac - bc - an - bm)t + abc = 0 \tag{3.13}$$

The equation can be solved by any approximation algorithms or cubic formula. For a cubic equation

$$x^3 + \alpha x^2 + \beta x + \gamma = 0 \tag{3.14}$$

let $Q = \frac{\alpha^2 - 3\beta}{9}$, $R = \frac{2\alpha^3 - 9\alpha\beta + 27\gamma}{54}$, and $\theta = \arccos \frac{R}{\sqrt{Q^3}}$. The solution for the cubic equation is then,

$$\begin{aligned} x_1 &= -2\sqrt{Q}\cos\frac{\theta}{3} - \frac{\alpha}{3} \\ x_2 &= -2\sqrt{Q}\cos\frac{\theta + 2\pi}{3} - \frac{\alpha}{3} \\ x_3 &= -2\sqrt{Q}\cos\frac{\theta - 2\pi}{3} - \frac{\alpha}{3} \end{aligned} \tag{3.15}$$

Cubic equation generally has three solutions (if real solutions exist). We have the following theorem regarding the three real solutions.

Theorem 3.1 *Given the problem defined above, there are three solutions t_1, t_2 and t_3 for the cubic equation, and assume $t_1 < t_2 < t_3$, then t_2 is the point where M reaches the global maxima.*

Proof: The solution t must be less than a and b, because X must be less than A_1 and A_2.

Observing $f(t) = t^3 + (m + n + c - a - b)t^2 + (ab - ac - bc - an - bm)t + abc$, it is clear that $f(0) > 0$, $f(a) < 0$ and $f(b) > 0$. Thus $t_1 < 0$, $0 < t_2 < a$, and $a < t_3 < b$. t_2 is the only feasible solution. □

The final estimates can be calculated as the average of the estimates on four possible combinations.

3.7.4 Evaluation

A set of simulations are run to evaluate the performance of distance estimation schemes. We compare the performance of using one and two transmission power levels. The communication range R_c of the single power level is normalized to 1 and the communication

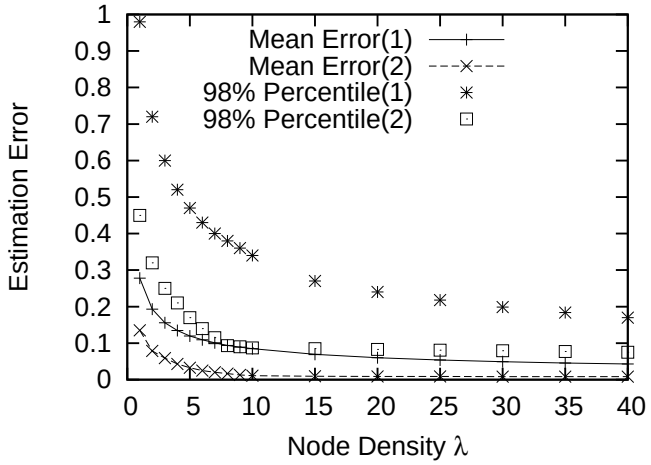


Figure 3.11: Distance estimation error (98% percentile and mean) v.s. node density. Single and dual power levels are indicated as (1) and (2) respectively.

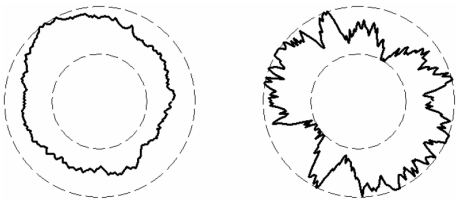


Figure 3.12: Radio pattern examples with DOI=0.05 and 0.2 respectively. [46]

ranges of two power levels are $R_{c1} = 0.5$ and $R_{c2} = 1$ respectively. The results are show in Figure 3.11. It can be clearly seen that with low node density, the estimation based on multiple transmission power gives significant improvements on estimation accuracy. The performance of the estimation improves with the increasing of node density.

The mean distance estimation error can be reduced to 2% of R_{c2} for node density larger than 5 using two transmission power levels. At such node density, the 98% percentile values is less than 10% of R_{c2}.

Next, we relax the assumption on the perfect binary disk communication model. Instead, we adopt the model suggested in [46]. In this model, there is a lower bound and upper bound on signal propagation. Beyond the upper bound, all nodes are out of communication range; and within the lower bound, every node is guaranteed to be within communication range. In between lower and upper bound, degree of irregularity (DOI) is

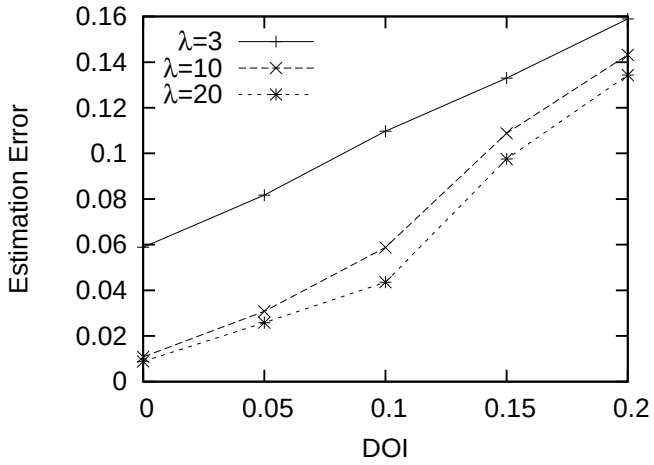


Figure 3.13: Mean distance estimation error v.s. DOI

used to denote the irregularity of the radio pattern. DOI is defined as the maximum radio range variation per unit degree change in the direction of radio propagation. Figure 3.12 shows an example of radio irregularity with the value of DOI 0.05 and 0.2 respectively (DOI of value 0 is the same as the perfect binary disk model).

Figure 3.13 shows how the estimation (two power levels) error varies with DOI (assume upper bound is 1 and lower bound is 0.5 for the first power level, and upper bound is 0.5 and lower bound is 0.25 for the second power level). It can be observed that estimation error increases almost linearly with DOI. With a relatively high irregularity (DOI=0.2) in communication range, and with two power level of estimation, the average error can still be confined in about 15% of R_{c2}.

Finally, in order to execute the estimation algorithm, there is still the need to estimate the node density. Our simulation result shows that for a randomly deployed sensor field with sufficiently high node density ($\lambda \geq 10$), the local node density can be approximated with an error less than 10% if the densities of all 1-hop neighbors are averaged. Hence, it is possible to estimate the node density locally even if this information is not available.

3.8 Summary

In this chapter a configurable coverage protocol (CCP) which uses only distances among the neighboring nodes is proposed. CCP is able to estimate the vacancies distributively and the global coverage objective α can be maintained. Using simulation, the effects of distance estimation error on coverage density control protocols (OGDC) are investigated. CCP performs very similar to OGDC for complete coverage. By relaxing the constraints of complete coverage, CCP is able to generate a subset of sensor nodes which is smaller than the number of nodes required for a complete coverage. At last, a simple distance estimation algorithm which can be utilized by CCP is proposed and evaluated.

Chapter 4

Microscale Connectivity Monitoring

4.1 Introduction

In traditional centralized network management, network topology is one of the key parameters that needs to be known in order to perform operations like performance management, fault detection and isolation, and capacity planning. Large displays showing the network topology are common sights in Network Operation Centers (NOC). In the Internet, despite the fact that the control and ownership are highly distributed, researchers have also attempted to gain understanding of the Internet topology. Examples include [3], [1], [106] and [31]. Knowledge of Internet topology allows researchers to better understand important issues such as Internet growth, routing behaviors, and DDoS attacks.

In wireless sensor networks, in addition to tackling traditional network issues such as fault monitoring/debugging and root-cause analysis [81], connectivity information also helps in ways that are unique to how sensor networks operate. For example, it is observed in [17] that connectivity statistics can be used to compute mean topological density, study the impact of link asymmetry, evaluate geographical routing algorithms, and assess behaviors of algorithms that depend on spatial correlation.

The complete network connectivity graph is formed by aggregating the microscale connectivity information (neighbor tables) of all the sensor nodes in the region of inter-

est. Microscale connectivity monitoring is thus an important management task in sensor networks.

However, obtaining the local connectivity information efficiently in wireless sensor networks is generally a hard problem. First, the neighborhood information at each sensor node is large. This is especially true for a dense network (tens of neighbors per node). Second, connectivity is highly unpredictable due to low power transmission, limited energy resource, ad-hoc deployment and factors such as obstacles and movement in the environment. As connectivity of wireless links can vary over time, nodes need to send information to the central controller periodically or on-demand, via multiple hops. The cost can be significant due to the limited energy and bandwidth resources available on the sensor nodes.

As stated in Chapter 2, previous protocols either reduce the number of nodes that will send their neighborhood information to a central controller [28, 27, 29], or let each sensor node send a subset of its neighbors to the central controller [27, 29]. Both approaches result in significant loss of accuracy.

In this chapter, we propose a Hop-count and Hashing-based Connectivity Monitoring (H^2CM) algorithm, a flexible and efficient algorithm to obtain connectivity information of the nodes located in the area of interest (monitored nodes). H^2CM is based on a *divide-and-conquer* approach, in which several techniques are combined to deal with various network and neighbor set sizes. These techniques are (1) hop count filtering, (2) Bloom filter and (3) use of a single hash value as checksum. By varying the amount of information exchanged, H^2CM is able to provide different levels of connectivity information accuracy.

H^2CM is flexible in that each node can be individually configured to provide the desired accuracy. As a result, nodes deemed more important can be configured to provide more accurate connectivity information. H^2CM is efficient in reducing communication cost, even when complete connectivity information of the nodes is required.

At last, a simple application of connectivity monitoring – node failure detection will

be studied. By combining H^2CM with the concept of dominating set, the communication cost can be drastically reduced compared to traditional data collection methods.

4.2 System Model

There are T nodes in the network and each node has a unique global ID. This ID can be its own MAC address or assigned by any ID assignment protocol that ensures the globally unique property [76]. Through pre-planning or a one-time initialization process, the central controller is assumed to be aware of the identities of the deployed nodes.

The size of node ID t (in number of bits) is at least $\lceil \log(T) \rceil$ bits, i.e., $t \geq \lceil \log(T) \rceil$. In this chapter, we use log to represent logarithm of base 2 unless otherwise mentioned. Let $X_i = \{x_1, \ldots, x_{m_i}\}$ be the set of neighbors of a node i and m_i be the size of X_i. When the context is clear, we also use m to represent m_i and X to represent X_i.

Connectivity information is uni-directional (links can be asymmetric, which is common in wireless networks [17, 4]). Based on existing link management process using periodic beacons, each node determines the set of connected neighbor nodes with incoming links. The definition of a connected neighbor depends on the application domain. For example, a node A can be considered to be connected to node B if at least one of the last several beacon packets transmitted by A can be received by B.

The connectivity monitoring process is performed by the central controller and the individual nodes to be monitored. Monitoring can be performed for a single node, the whole network, or any subset of nodes. Each monitored node sends its own neighborhood information to the central controller via possibly multiple hops.

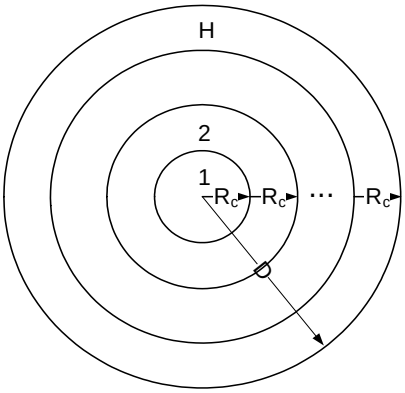


Figure 4.1: An illustration of the ring model.

4.3 Cost Analysis

4.3.1 Cost of Microscale Connectivity Monitoring

The communication cost is affected by the following factors: number of hops in the routing path, amount of data sent by the source node, data aggregated in the intermediate relay nodes and wireless transmission overhead (e.g. retransmission due to noise or interference). In this work, we only consider the first two factors. We do not consider retransmissions of packets nor aggregation of data in the intermediate nodes.

Assume a node is h hops away from the central controller and the amount of neighborhood data to be sent at this node is I, the communication cost for the central controller to know the neighborhood information of this node is simply hI.

The analysis on the communication cost of a complete neighborhood discovery can be based on a simple model called *ring model* as shown in Figure 4.1. In [20], the authors show that the number of hops (h) and geometric distance (d) that a packet travels, in high-density networks and a broadcast percolation scenario, can be well approximated by the following relation: $h = \lceil \frac{d}{R_c} \rceil$, where R_c is the average communication radius of the wireless nodes. Based on this, ring model can be used to analyze the communication cost of connectivity discovery [27, 29]. Though this estimation does not work well in low-density networks, the accuracy of h does not affect the intuition behind. For simplicity of analysis, we do not consider boundary conditions, and only the nodes inside the circle of

radius $D = HR_c$ are counted, where D is the maximum radius of the area of interest and H is the maximum number of hops.

As shown in Figure 4.1, the nodes that are h hops away from the central controller lie between the circles of radii $(h-1)R_c$ and hR_c, and we call this region ring h, where $h \in [1, H]$. The area of ring h is given by,

$$\pi(hR_c)^2 - \pi((h-1)R_c)^2 = \pi(2h-1)R_c^2 \tag{4.1}$$

Thus, there are on average $\lambda\pi(2h-1)R_c^2$ nodes located in ring h, where λ is the average network node density.

Assume I is the average amount of neighborhood data to be sent at each node, the total average cost to retrieve the complete connectivity of the whole network is given by,

$$\begin{aligned} & \sum_{h=1}^{H} \lambda\pi R_c^2 I(2h-1)h \\ = \; & \frac{1}{6}\lambda\pi R_c^2 I(4H^3 + 3H^2 - H) \end{aligned} \tag{4.2}$$

For a complete connectivity discovery, the term I is highly related to the communication radius R_c and network node density λ. If we assume each node simply sends all of its neighbor IDs, in a unit disk graph model, $I = O(\lambda R_c^2)$. The total communication cost shown in Equation 4.2 is then $O(\lambda^2 R_c^4 H^3)$, which increases rapidly with network size and node density.

A node in ring h sends its own neighborhood data, as well as relayed data for nodes in rings $h+1, \ldots$, and H. The average amount of data to be sent at a node in ring h is

$$\begin{aligned} & \frac{\sum_{i=h}^{H} \lambda\pi(2i-1)r_c^2}{\lambda\pi(2h-1)r_c^2} I \\ = \; & \frac{H^2 - h^2 + 2h - 1}{2h-1} I \end{aligned} \tag{4.3}$$

Note that each node in ring 1 has to send a total of H^2I amount of data where I is $O(\lambda r_c^2)$.

From Equations 4.2 and 4.3, it can be observed that there are two ways to reduce the communication cost. One way is to only let a small portion of nodes transmit its own neighborhood information, such as MDS and MVDS in [28, 27, 29]. By doing so, the value of λ in Equation 4.2 and 4.3 can be reduced. However, such an approach can significantly reduce the accuracy of the connectivity information. Another way is to let each node send less data, such that I is reduced. In [27, 29], each node also sends only a subset of its neighbor IDs. Again, these approaches also result in significant loss of neighborhood information.

In this paper, our approach to cost reduction is through reducing I. Unlike previous approaches, our approach reduces I with no or little loss of neighborhood information. It is worthy noting that reducing I has no conflicts with reducing the number of nodes who transmit their neighborhood information. One can still choose to use MDS-based approach [28] orthogonally with our work. More details are shown in Section 4.10.

4.3.2 Related Encoding Techniques

This section summarizes other possible techniques in encoding (and possibly reducing) the amount of data I sent by each node and their corresponding limitations. These techniques include direct transmission, bitmap, and hashing.

Direct Transmission: In the most direct form, a node transmits its neighbor IDs directly to the central controller. Without considering the packet overheads, the size of data to be sent is mt bits, where m is number of neighbors of the node and t is the size in bit of the node's ID. When mt is small, e.g. in a sparsely deployed network, direct transmission may be the most appropriate mechanism.

Bitmap Representation: With bitmap, each node transmits a bit string of size T to the central controller. The central controller decides whether node k is a neighbor of node i by looking at the k^{th} bit in the bit string node i transmits. The size of data transmitted is

at least T bits if the bitmap representation is compact. Note that bitmap will only be more efficient than direct transmission when $m > \frac{T}{t}$ or $t > \frac{T}{m}$.

For large network where $T \gg m$ and $m < \frac{T}{t}$, direct transmission is more efficient than bitmap. For mid-size network with relatively high density, m may be larger than $\frac{T}{t}$ and bitmap is more efficient.

If we consider the case where the bitmap can be efficiently compressed, the data size can be even smaller, especially for large sensor networks. However, in any case, the maximal compression is lower bounded by $\log \binom{T}{m} \geq m(\log(T) - \log(m))$ [77].

Since the physical address of a sensor node (e.g., MAC address) can be 16 bits or even 32 bits and more, the identities of the sensor nodes need to be mapped to position in the bitmap for efficient representation. Such a mapping needs to be performed in advance and nodes have to be informed if there are changes to the mapping. For a sensor network where self organization is important, use of pre-configured and static information is a serious drawback.

Exact Membership Testing Using Hashing: Hashing is a common solution to compress the data for membership information. The space required to hash the neighbor IDs such that they can be decoded *exactly* is also lower bounded by $\log \binom{T}{m}$ [16].

In summary, considering both bitmap compression and hashing, the maximum savings in theory over direct transmission is $m \log m$. The reduction is $\frac{\log m}{t} < \frac{\log m}{\log T}$. As m is number of neighbors of a node, the reduction is small when the network size is large with respect to communication range.

4.4 Overview of H^{2}CM

In this section, we describe H^{2}CM, a Hop vector distance and Hashing-based Connectivity Monitoring scheme.

The central controller is assumed to maintain three sets of nodes for each monitored node i. The three sets are:

- V_i, the set of confirmed neighbors of node i,
- U_i, the set of nodes whose relationship with node i cannot be determined,
- W_i, the set of confirmed non-neighbors of node i.

Let $v_i = |V_i|$, $u_i = |U_i|$ and $w_i = |W_i|$.

Note that W_i is introduced only for convenience of the description and does not have to be maintained in practice. Initially, V_i and W_i are empty. U_i contains all other nodes in the network except node i. At all times, the union of i, V_i, W_i and U_i forms the set of all nodes in the network T.

Each sensor node i transmits its own connectivity information to central controller. Intuitively, H^{2}CM tries to reduce u_i and increase v_i at central controller using the combination of several techniques so that the most appropriate technique can be applied in different situations. The objective is to achieve the desired accuracy with the minimum communication cost, where the accuracy is defined by $\frac{v_i}{m_i}$. Recall that m_i is the number of connected neighbor nodes with incoming links. Hence, $\frac{v_i}{m_i} \leq 1$.

The first technique of the algorithm applies when the values of $\frac{u_i}{m_i}$, m_i and u_i are large. The technique utilizes hop count to compute hop vector distance between nodes to identify possible set of neighbors. The value of u_i and thus $\frac{u_i}{m_i}$ maintained at central controller can be effectively reduced. This is presented in Section 4.5.

The second technique involves the use of Bloom filters for approximate membership testing. This technique is most appropriate when $\frac{u_i}{m_i}$ is less than some bounded value (see Section 4.6.3). Our use of Bloom filter is unique in two ways. First, traditional Bloom filter can only remove non-members (move elements from U_i to W_i). In our approach, Bloom filter can also confirm nodes as members (move elements from U_i to V_i). Second, we use a combination of normal and counting Bloom filters depending on the values of m_i, u_i and v_i. The details are presented in 4.6.2. Analysis on the behaviors of Bloom filters is provided in Section 4.6.3.

In the third technique, if $m_i - v_i$ and u_i are small enough, a node can use hashing to generate fingerprint of all m_i nodes to help the central controller identify the complete set of its neighbors. The discussion is presented in Section 4.7.

Data needed for the techniques applied can be combined into a single packet resulting in only one transmission from each monitored node i. In Section 4.8, we describe how all these techniques are put together.

4.5 Hop Vector Distance-based Neighborhood Constraints

In order to more "accurately" decide if two nodes can be neighbors, location information is the most natural neighborhood constraint. Only nodes that are within the maximum communication distance can be neighbors. However, localization itself is a challenging research issue and often incurs substantial overhead. In this work, we propose the use of hop vector distance computed from connectivity based localization [74] to remove a large amount of non-neighbors for each node when T is large.

We assume at the end of the localization process, the central controller knows the locations of all nodes. While the hop count localization process and collecting of location information from all nodes incur substantial cost, this process will only need to be performed once. It can be reused for later cycles of connectivity collection or update as long as there is no substantial change of this initialized hop counts to the relative locations of the neighbors. For a large network, taking into account the gain in reducing the candidate set for all nodes and the cost amortized over many monitoring cycles, the benefit can easily outweigh the cost.

Assume that there are S anchors in the network and each node maintains its own hop count to the S anchors in a hop count vector $(h_{i1}, \ldots, h_{iS})$. The hop vector distance d between two nodes i and j is calculated using 3-norm distance between two hop count

vectors, i.e.,

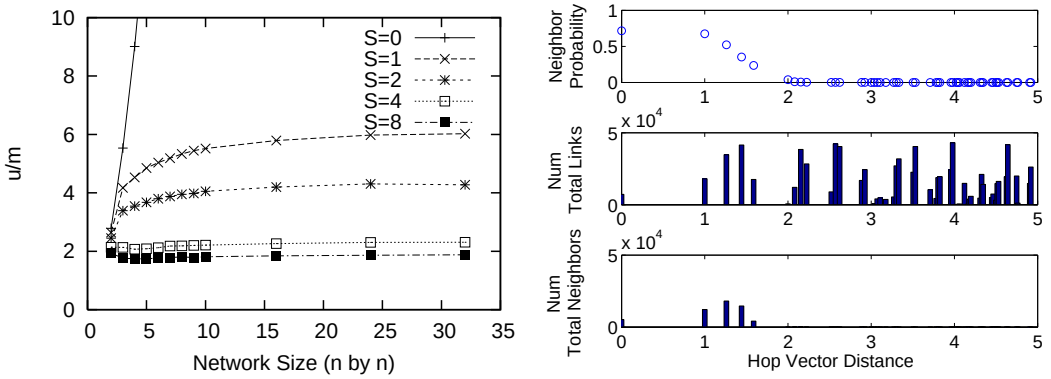
$$d = \sqrt[3]{\left(\sum_{s=1}^{S} |h_{is} - h_{js}|^3\right)} \tag{4.4}$$

A lower norm like 2-norm is not desirable because it is not able to differentiate the two cases where two nodes have absolute hop count difference vectors (2,0,0,0,0,...,0) and (1,1,1,1,0,...,0). 3-norm provides enough accuracy to differentiate most cases for a small value of S. One the other hand, 3-norm is just enough and larger norm is harder to calculate in computational limited sensor nodes.

Each node sends to the central controller the ID of the neighbor node with the largest hop vector distance. With this information, the central controller can move nodes from U_i to W_i (reduce u_i). All nodes with larger hop vector distance cannot be a neighbor of node i. The utility of the hop vector distance based technique is in terms of the size of potential neighbors in U_i relative to the actual neighbor size m_i after this phase.

We evaluate the utility of the hop vector distance using simulation. The network area is varied from 2×2 to 32×32 and maximum transmission range is normalized to 1. The number of anchors S is set from 1 to 8 and the node density λ simulated is from 5 to 30. In order to generate graphs with different characteristics, we also define two parameters *link connectivity (LC)* and *link asymmetricity (LA)*. Link connectivity is defined as the ratio of node pairs that are able to communicate (at least in one direction) to nodes that are within maximum communication range. Link asymmetricity is defined as the percentage of asymmetric links over the total number of uni-directional and bi-directional links.

The result of one specific case when $\lambda = 10$, $LC = 0.8$ and $LA = 0.2$ is shown in Figure 4.2. The trends are similar for other cases. Figure 4.2(a) shows the ratio of average u and m versus the network size as well as the number of anchors S. Without the hop vector distance based filtering technique (number of anchors $S = 0$), the value of $\frac{\overline{u}}{\overline{m}}$ increases fast with the network size, where $\overline{u}$ and $\overline{m}$ are expected values of u and m. It can be observed that, with 4 anchors, the ratio of average u and m is less than 2.3 even for very large network. A larger number of anchors only performs marginally better.

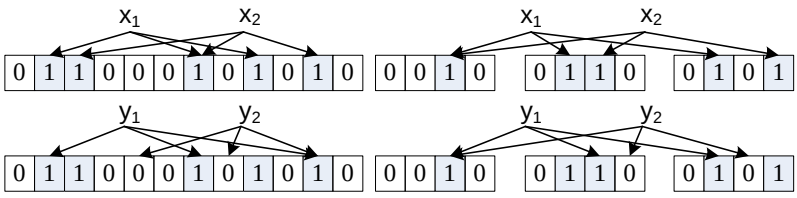


(a) Hop vector distance utility evaluation (b) Neighbor probability versus hop vector distance

Figure 4.2: Effects of hop vector distance based technique.

Figure 4.2(b) shows the distribution of neighbors versus the hop vector distance d when $S = 4$. The top graph shows the probability that a node with a specific hop vector distance away from node i is actually a neighbor of i. The middle graph shows hop vector distance for all node pairs (2560x2559) with respect to the hop vector distance d. The distribution of actual number of neighbors of a node with respect to d is shown in the bottom figure. We can see that when the hop vector distance of two nodes is greater than 2, the probability that they are neighbors drops to almost 0. It in turn shows the effectiveness of how hop vector distance can be used to reduce the value of u_i.

From the results, we can see that, filtering based on hop vector distance is very useful for reducing the size of U_i, especially in large sensor networks. However, utilizing hop vector distance based approach alone is apparently not enough because although it effectively removes elements from U_i at the central controller, it does not help in determining which of the remaining elements in U_i are actually neighbors. In the next two sections, we present Bloom filter-based approach that can effectively move elements from U_i to V_i.



(a) A standard Bloom filter (b) A Bloom filter variant

Figure 4.3: Examples of Bloom filters.

4.6 Bloom Filter-based Connectivity Monitoring

4.6.1 Bloom Filter Preliminaries

Before we go into the details of the basic idea of our Bloom filter approach, some fundamental knowledge on Bloom filters is introduced.

A Bloom filter [9, 10] is a simple and space-efficient probabilistic data structure that belongs to the class of approximate membership testers as given in [16]. It is used to represent a set with much less space requirement than directly representing the entire whole set. Membership testing over Bloom filters is simple and fast though a small probability of false positives may present. Recently Bloom filters have been widely applied in networking areas such as distributed caching [35, 86], p2p and overlay networks [13], measurement [60], and many others.

The standard form of Bloom filter represents a set $X = \{x_1, \ldots, x_m\}$ using a bit array of length b bits. There must also be k independent hash functions $h_1, \ldots, h_k$ defined, and each of the function hashes any value in the universal to a value of range $[1, b]$ uniformly.

To construct the Bloom filter, the bit $h_i(x)$ of the bit array is set to 1 for each $i \in [1, k]$ and for each element $x \in X$. To check whether an element y is in X, we simply check whether the bit positions $h_i(y)$ for all $i \in [1, k]$ are 1. It is clearly seen that if y is indeed a member of X, it will never be considered not. However, if y is not a member of X, there is a possibility that it can be considered as a member of X (false positive). This is illustrated in Figure 4.3(a), where y_1 is considered to be in set X and y_2 is not.

The false positive probability can be approximated as [10],

$$(1-(1-\frac{1}{b})^{km})^k \approx (1-e^{-km/b})^k \tag{4.5}$$

Given fixed values of m and b, the value of k that minimizes the false positive probability is,

$$k = \ln 2 \frac{b}{m} \tag{4.6}$$

In another word, given fixed values of m and k, increasing b (using more space) always reduces the false positive probability but the most efficient size of b is,

$$b = (\log e)km \tag{4.7}$$

There is another formulation of Bloom filter which takes a slightly different form. The bit array of size b is divided into k disjoint bit arrays of size $\frac{b}{k}$ each. Each of the hash functions h_1 to h_k has an output range of $[1, \frac{b}{k}]$. To construct the Bloom filter, set the bit position $h_i(x)$ of bit array i to be 1 for each $x \in X$. The membership testing is similar to the standard form. This process is shown as an example in Figure 4.3(b), where y_1 is considered to be in set X and y_2 is not.

Again, there is probability of false positives, the false positive probability can be approximated as,

$$((1-(1-\frac{k}{b})^{m})^k \approx (1-e^{-km/b})^k \tag{4.8}$$

which is asymptotically close to the false positive rate of standard Bloom filter.

Given fixed values of m and b, the value of k that minimizes the false positive probability is given by,

$$k = \ln 2 \frac{b}{m} \tag{4.9}$$

And given fixed values of m and k, the most efficient bit array size is,

$$\frac{b}{k} = (\log e)m \tag{4.10}$$

Note that the false positive probability of the second form is asymptotically larger than first of standard form (although the difference is small), because the following inequality always holds,

$$(1-\frac{k}{b})^m \leq (1-\frac{1}{b})^{km} \tag{4.11}$$

A more general form of Bloom filter is to expand each bit in the bit array into a c bit counter. This is also known as the counting Bloom filter [35]. Whenever an element in x is hashed into an entry, we increase the counter associated with that entry by 1 if there is no overflow (greater than $2^c - 1$). Hence, a counting Bloom filter provides an exact count of the number of items that match that entry if there is no overflow. Note that when $c = 1$, it becomes a normal Bloom filter described above.

We choose to use the second form of Bloom filter in our design of connectivity discovery protocol. This is because the second form allows incremental update through sending smaller pieces of data, each using a different hash function, and it also allows the easier combination of results from several bit arrays using different c values. In the rest of the paper, when we mention Bloom filter, we refer this second formulation of Bloom filter.

4.6.2 Basic Idea

To apply Bloom filter in the context of connectivity monitoring, we assume each node i sends the central controller k_i rounds of counting Bloom filters with number of bits per entry from c_1 to c_{k_i} for round 1 to round k_i respectively. For each round of Bloom filter, the bit positions are set according to the hash values of all the elements in neighbor set X_i using the corresponding hash function.

Upon receiving the Bloom filters from a node i, the central controller is then able to remove some nodes from U_i to W_i (reduce u_i), as well as from U_i to V_i (reduce u_i and increase v_i) according to the properties described below.

Non-member Removal Property: The first property of a Bloom filter is the same as the traditional usage of Bloom filters explained in Section 4.6.1. We call it *Non-member Removal* property. Upon receiving the bit array from a node, the central controller tests each element in U_i and moves those that are not neighbors from U_i to W_i. After enough rounds of non-member removal from U_i to W_i, the set U_i will be V_i and the central controller can confirm the membership of set V_i (e.g., by checking the length of U_i is equal to the total number of neighbors of a node).

Membership Confirmation Property: The second property of Bloom filter applies when $X_i \subseteq (U_i \cup V_i)$, which means the initial guess of a node's neighborhood information at the central controller (U_i) by hop vector distance based scheme contains exactly all the neighbors of that node. We then have the following theorem.

Theorem 4.1 *Hash each element in X_i into a counting Bloom filter with number of bits per entry c ($c \geq 1$). Assuming the value of the counting Bloom filter at entry j is $s(j)$, then if there are only $s(j)$ elements in $U_i \cup V_i$ that hash into entry j, then the $s(j)$ elements in $U_i \cup V_i$ that hash into entry j must all be in X_i.*

Proof: This can be proved by contradiction. Consider one element is in U_i that hashes into entry j but is not in X_i. Since $X_i \subseteq (U_i \cup V_i)$, the number of elements in X_i that hashes into entry j cannot exceed $s(j)-1$, which is a contradiction because there are at least $s(j)$ elements in X hashed into entry j. □

Upon receiving Bloom filter data from a node, the central controller can confirm that some elements in U_i are in X_i. We call this *Membership Confirmation* property. This is an interesting property because unlike the traditional usage of Bloom filters, which probabilistically tests whether an element is in the set, now we are able to confirm some portion of the elements are in the set. These elements are moved from U_i to V_i. Note that

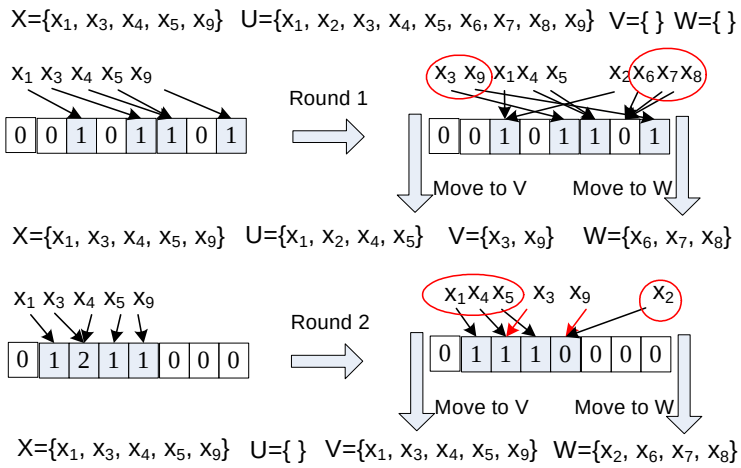


Figure 4.4: Bloom filter properties.

this property is not utilized in traditional Bloom filter applications because X_i is generally not a subset of $U_i \cup V_i$ whereas in our case X_i is always a subset of $U_i \cup V_i$.

Counting Removal Property: This property only applies for a counting Bloom filter with bits per entry c greater than 1. One property of the counting Bloom filter is that it supports deletion of an element when there is no overflow. Based on this property, the central controller can remove some elements of confirmed set V_i from the counting Bloom filter if overflow does not occur. We call this property as *Counting Removal* property. Give a Bloom filter bit array, this property should be applied if possible before the previous two properties to be applied.

An example showing how these properties can be applied is shown in Figure 4.4. In the example, initially $U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9\}$, V and W are empty. Two rounds of Bloom filters are applied with first round a normal Bloom filter with $c_1 = 1$ and second round a counting Bloom filter with $c_2 = 2$. After first round, x_6, x_7 and x_8 are moved into W because they all hash into a bit in the bit array with value of 0. x_3 and x_9 are moved to V according to membership confirmation property because they are the only nodes that hash into the bit array with bit value of 1. In round 2, firstly counting removal property is applied (x_3 and x_9), then according to non-member removal property, x_2 is moved into W and according to membership confirmation property, x_1, x_4 and x_5 are moved into V.

Integrating these three properties together, the central controller is able to remove

some of the non-neighbors from U_i to W_i, and it is also able to move the confirmed neighbors from U_i to V_i. Note that for non-member removal property, a normal Bloom filter is more efficient than a counting Bloom filter, whereas for membership confirmation property and counting removal property, a counting Bloom filter is better than a normal one because there is less probability of overflow. The challenge now is when to use a normal Bloom filter ($c = 1$) and when to use a counting Bloom filter ($c > 1$) and of what size. We answer this question in the next section.

4.6.3 Theoretical Analysis

In the following analysis, we assume the set of neighbor IDs of a node to be X, the confirmed neighbor set of this node at central controller to be V, and the non-confirmed neighbor set of this node at central controller to be U. Let $m = |X|$, $v = |V|$ and $u = |U|$. Also let $Y = V \cup U$ and $n = |Y| = v + u$. Note that $V \subseteq X$ and $X \subseteq Y$. Let $Z = V + U - X$ and $z = |Z| = v + u - m$. Z is therefore the set of non-members (but central controller is still not sure) in U.

For a Bloom filter sequence of $C = [c_1, \ldots, c_k]$, the number of total bits required is $b\sum_{j=1}^{k} c_j = m\log(e)\sum_{j=1}^{k} c_j$. This value increases linearly with k. To reduce the total number of bits required, k has to be bounded to a small value.

In this section, we investigate the effectiveness of how different sequences of Bloom filters C help in identifying neighbors and non-neighbors. We will first analyze the behavior of normal Bloom filters, followed by counting Bloom filters. The behavior of several rounds of mixed normal and counting Bloom filters will be studied at the end.

Normal Bloom Filters

Hashing m elements into a bit array of size $b = m\log(e)$, the probability that ith bit is set j times $P(m, b, s(i) = j)$ (or in short $P(m, b, j)$, or simply $P(j)$) is given by binomial distribution,

$$
\begin{aligned}
P(j) &= P(m,b,j) = \binom{m}{j}\left(\frac{1}{b}\right)^{j}\left(1-\frac{1}{b}\right)^{m-j} \\
&= \frac{\binom{m}{j}}{(b-1)^{j}}\left(1-\frac{1}{b}\right)^{m} \\
&= \frac{\binom{m}{j}}{(b-1)^{j}}P(m,b,0) \\
&= \frac{\binom{m}{j}}{(b-1)^{j}}P(0) \qquad (4.12)
\end{aligned}
$$

After a node hashes all its neighbor set X in a normal Bloom filter and sends the data to the central controller, the central controller will hash all nodes in U and V into the same size of bit array. The expected number of non-member nodes can be removed from U is given by $P(0)z$. The number of non-member nodes that still remain in U is then $(1-P(0))z$.

Without considering those nodes already in V (without considering counting removal property), the number of nodes that can be confirmed by the central controller to be neighbors is, the number of bits in the bit string that only one node in X hashes into ($P(m,b,1)b$), times the probability that for any bit, none of the node in Z (non-member nodes) is hashed into. This is given by

$$P(1)bP(z,b,0) \qquad (4.13)$$

where $P(z,b,0)$ is the probability that a bit remains 0 by hashing z nodes into bit array size of b. The percentage of nodes that will be confirmed by the central controller is then

$$p = \frac{P(1)bP(z,b,0)}{m} = P(1)P(z,b,0)(\log e) \qquad (4.14)$$

Similarly, in consecutive k rounds of normal Bloom filters (i.e., $c_j = 1 \forall j \in [1,k]$), the average non-member nodes in U at round j is $z_j = (1-P(0))^{(j-1)}z$, and the aver-

age percentage of nodes that can be identified by the jth round p_j is given by (without considering the nodes that have been confirmed by previous rounds),

$$p_j = P(1)P(z(1-P(0))^{(j-1)}, b, 0)(\log e) \tag{4.15}$$

Counting Bloom Filters

In general, a counting Bloom filter is not as space-efficient compared to a normal Bloom filter for the purpose of non-member removal. However, it can tolerate overflows so that the membership confirmation and counting removal property can be applied more efficiently. We will only consider counting Bloom filter of $c = 2$ in this section for the following two reasons. First, we want to use Bloom filter to provide better performance in reducing communication cost than direct transmission of compressed data. A Bloom filter of $c > 2$ is too costly. Second, in our application, when $c > 2$, the gain of tolerance on overflow is small compare to $c = 2$.

Without considering the set of confirmed neighbors, V, Equation 4.13 can be generalized to,

$$(P(1) + 2P(2) + 3P(3))bP(z, b, 0) \tag{4.16}$$

Note that for normal Bloom filters in sequence of two, the total number of nodes identified is (excluding V),

$$(p_1 + p_2 - p_1 p_2)m \tag{4.17}$$

where, p_1,p_2 are given in Equation 4.15.

Comparing Equations 4.16 and 4.17, we have the following theorem,

Theorem 4.2 *Two consecutive normal Bloom filter will be better than a counting Bloom filter with $c = 2$ in terms of number of neighbors can be confirmed by the central controller (without considering those that has already been confirmed in previous rounds), if and*

only if,

$$n > \frac{2\log 0.7661}{\log(1-\frac{1}{(\log e)m})} + m \tag{4.18}$$

where n is the total number of nodes in U and V, m is the size of X.

Proof: Assume m is large, so $P(m,b,j) \approx \frac{1}{2(\log e)^j j!}$ when j is small.

The number of identified nodes by the central controller by two consecutive Bloom filter is given by Equation 4.17, by 1 counting Bloom filter of $c = 2$ is given by Equation 4.16, let,

$$\begin{aligned} & (4.17) - (4.16) > 0 \\ \Rightarrow \quad & (0.5a^2 + 0.5a - 0.25a^3)m - 0.9667a^2 m > 0 \\ \Rightarrow \quad & 0.25a^2 + 0.4665a - 0.5 < 0 \\ \Rightarrow \quad & 0 < a < 0.7661 \end{aligned} \tag{4.19}$$

where $a = (1 - \frac{1}{(\log e)m})^{\frac{z}{2}}$.

Thus, when $z = n - m > \frac{2\log 0.7661}{\log(1-\frac{1}{(\log e)m})}$, two consecutive normal Bloom filter will be better than a counting Bloom filter with $c = 2$. □

However, what has been analyzed on counting Bloom filter is purely based on the non-member removal and membership confirmation properties. Counting Bloom filter has one more advantage: if the central controller has already identified some portion of elements to be in X, these elements can be deleted from the counting Bloom filters if there is no overflow.

Recall that $v = |V|$ is the number of nodes that have already been identified by the central controller. By deleting them from the counting Bloom filter, there will be more "0" entries available which can be useful in non-member removal property. Since $c = 2$, it is possible to delete those that have already been identified only when number of elements hashed into the bit is 1 or 2.

The probability that an identified node can be deleted from Bloom filter is given by,

$$\frac{P(1)+2P(2)}{\sum_{j=1}^{m} jP(j)} = \frac{P(1)+2P(2)}{\frac{1}{\log e}} \tag{4.20}$$

For those entries where one or two elements are hashed into, deleting the element(s) will give more 0 bits. Thus, the increase of percentage of 0 bits is,

$$\begin{aligned} & \frac{(\log e)P(1)v+(\log e)2P(2)\frac{v}{m}v}{(\log e)m} \\ = \quad & P(1)\frac{v}{m}+2P(2)\frac{v^2}{m^2} \end{aligned} \tag{4.21}$$

When v is close to m, the increase in number of 0 entries is large. Removing more non-members will help member confirmation and non-member removal in the next round, and will help to increase the chance of successfully using fingerprint hashing for identification explained in Section 4.7.

Thus, when the central controller already knows a large portion of the nodes, sending one counting Bloom filter is probably an advantage because firstly n has already been reduced to a small value when the central controller has already known a large portion of the neighbors of node i. Second, counting Bloom filter will remove more non-members, which is an advantage for the next round.

The discussion leads to the following heuristic. If the number of confirmed neighbors (v) is small comparing to the actual number of neighbors (m), two normal Bloom filters ($c = 1$) tends to perform better a counting Bloom filter with $c = 2$) of the same total size, and vice versa.

Bloom Filters of k Rounds

In this section, we will try to generalize the previously mentioned heuristic to multiple rounds (≥ 2) of normal and counting Bloom filter. The goal is to use just enough Bloom filter data so that the number of unconfirmed nodes $m-v$ is smaller than some pre-defined

value or small enough to utilize a simple fingerprint value for complete identification (Section 4.7).

For each round i, if it is normal Bloom filter, we have

$$\begin{aligned} p_i &= P(1)P(z,b,0)\log e \\ z' &= (1-P(0))z \\ p' &= p + p_i - p_i p \end{aligned} \tag{4.22}$$

If it is a counting Bloom filter of $c = 2$, we have

$$\begin{aligned} p_i &= (P(1)+2P(2)+3P(3))P(z,b,0)\log e \\ z' &= (1-P(0)-P(1)p-2P(2)p^2)z \\ p' &= p + p_i - p_i p \end{aligned} \tag{4.23}$$

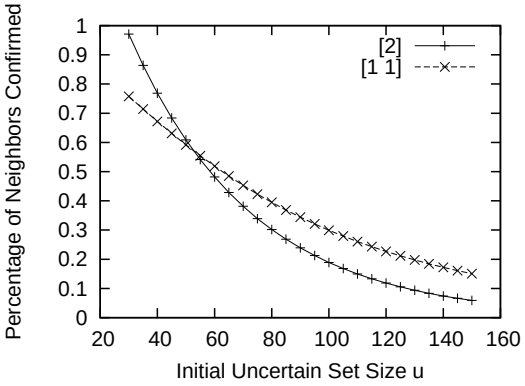
where z' and p' are the updated value of z and p for the next round respectively.

The final total percentage of nodes can be identified in average is then,

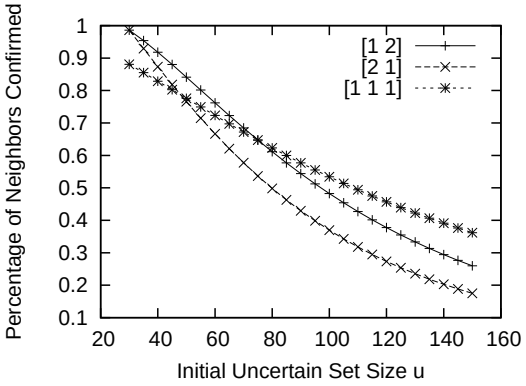
$$\sum_{i=1..k} p_i - \sum_{i=1..k, j=i+1..k} p_i p_j + \sum_{i=1..k, j=i+1..k, s=j+1..k} p_i p_j p_s - \ldots \tag{4.24}$$

It's direct form is hard to derive but numerical solutions are easy to calculate. As an illustration, we plot the percentage of neighbors confirmed by the central controller for $\overline{m} = 30$ in Figure 4.5 for different values of initial uncertain set size $\overline{u}$. Values of different $\sum_{j=1}^{k} c_j$ are plotted, where c_j is the value of c for the Bloom filter size at round j. We use $[c_1, \ldots, c_k]$ to represent Bloom filters of k rounds. In the plots, all combinations of using $c = 1$ or 2 to sum to the values 2, 3 and 4 are compared. The results for other values of m are similar to Figure 4.5 except that the crossover points occur at different values.

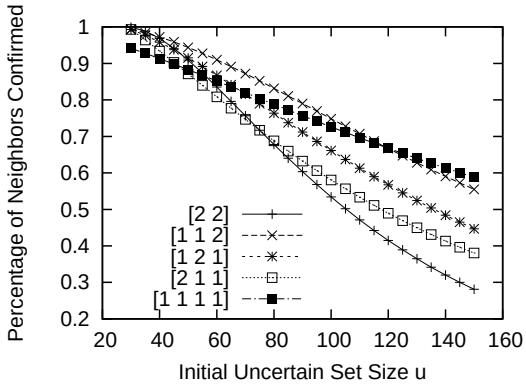
From the figure, it can be observed that the Bloom filter sequence that starts with $c_1 = 1$ always outperforms those that starts with $c_1 = 2$. This coincides with the theorem



(a) $\sum_j c_j = 2$



(b) $\sum_j c_j = 3$



(c) $\sum_j c_j = 4$

Figure 4.5: Comparison of consecutive Bloom filters ($\overline{m} = 30$).

shown in Section 4.6.3. From Section 4.5, we also see that with using hop vector distance as a filter, $\frac{\overline{u}}{\overline{m}}$ is between 2 to 2.5 when $S = 4$ (i.e., initial value of $\overline{u}$ input to Bloom filters is between 60 to 75). This corresponds to 40% to 60% of neighbors to be confirmed in the best case in Figure 4.5(a), 65% to 85% in Figure 4.5(b), and 85% to 95% in Figure 4.5(c). $\sum c_j = 4$ with sequence [1 1 2] or [1 1 1 1] are good choices under this situation.

Given the neighbor set size m_i (known at node i), as well as initial uncertain set size u_i, node i is able to estimate the best Bloom filter sequence it requires for the central controller to confirm at least the pre-defined percentage of neighbors given by Equation 4.24. Considering the limited computational resources, these equations may seem complex for implementation on the sensor nodes. Fortunately, in our applications, the ratio $\frac{\overline{u}}{\overline{m}}$ is almost always below 2.5 after the first technique is applied. Thus, the values of $\sum_{j=1}^{k} c_j$ tends to be small to give a good performance.

4.7 Fingerprint Hashing

Because Bloom filter is a probabilistic structure, as more neighbor nodes are recognized, fewer new members can be confirmed at the next round. To completely recognize all the neighbors, a large amount of rounds may be required.

However, if most of the neighbors have been recognized by the central controller, a node can just simply choose to hash all its neighbor IDs into a fingerprint value (say 32 bits), and append this fingerprint value to the bit arrays generated by the Bloom filter to the central controller. After applying Bloom filter on the bit arrays, the central controller may perform search using the hash value to obtain the complete list of neighbors if needed.

The cost of searching is upper bounded by $\binom{u}{m-v}$. If the central controller sets the search threshold per node to 10^6, then any value of $u \leq 22$ can be searched, independent of the value of $m - v$ (since $\binom{22}{11} < 10^6$). Similarly, if the search threshold is 10^4, then $u \leq 12$ is always fine.

In cases where the central controller finds it too expensive to perform the search, it

can ask for more rounds of Bloom filters data from those nodes, which would be very few in number. Finally, note that since hash values can collide, there is a very small probability that some neighbors are falsely identified from nodes in U. However, the collision probability is very small when u is small and hash value is sufficiently large.

It is also worth noting that, with fingerprint hashing, the complete neighborhood information of each node can be obtained with very high probability. However, its communication cost can be even smaller than the scheme without fingerprint hashing where only a percentage of neighbors could be obtained, because for the former case, fewer rounds of Bloom filters can be required. This is verified by comparing the simulation results in Sections 4.9.1 and 4.9.2.

4.8 Flow of H^{2}CM

4.8.1 Connectivity Initialization

There are two parts to the execution, one on the central controller and the other on the nodes monitored. We only show the pseudo code on the monitored nodes in Algorithm 1. In addition, for ease of explanation, we only show the algorithm for complete connectivity (with fingerprint hashing). If only a pre-defined percentage of neighbors is required and additional computation cost is also allowed at central controller, the node has to estimate and compare the communication costs of the schemes with and without fingerprint hashing, and chooses the one with lower cost. This is not included in Algorithm 1.

Each monitored node first estimates the required Bloom filter sequence such that the communication cost can be minimized while the searching threshold can be satisfied (line 1-15). Note that this estimation is based on the fact that fingerprint hashing will be applied. If the fingerprint hashing will not be applied, the estimation shall be based on the percentage of neighbors confirmed as analyzed in Section 4.6.3. This is not shown in the pseudo code.

The value of u (line 18) is closely related to the network parameters such as *link con-*

Algorithm 1 H^{2}CM at Each Monitored Node

Require: Neighbor table X, hop count location of neighbors, neighbor ratio α and searching threshold $thresh$.

function BFSEQUENCE($u, m, b, thresh$)
  $nBins \leftarrow 1, search \leftarrow \infty$
  while $search > thresh$ **do**
    for $c \in \{c | \sum c[i] = nBins, c[i] = 1 \text{ or } 2\}$ **do**
      Calculate z and v using Equation 4.24
      $u \leftarrow m - v + z$
      if $\binom{u}{m-v} < search$ **then**
        $search \leftarrow \binom{u}{m-v}, bfs \leftarrow c$
      end if
    end for
    $nBins \leftarrow nBins + 1$
  end while
  return bfs
end function

$j \leftarrow \arg\max_{i \in X} d_i$
$m \leftarrow |X|, u \leftarrow \lceil \alpha m \rceil, b \leftarrow \lceil \log(e)m \rceil, s \leftarrow m \log(|T|)$
$c \leftarrow$ BFSequence$(m, u, b, thresh)$
if $\log(|T|) + b(\sum c[i]) + sizeof(sig) \geq s$ **then**
  Send IDs directly
else
  Allocate space of $b(\sum c[i])$ bits
  for $i \in [1, len(c)]$ **do**
    Do Bloom filtering of each neighbor IDs
  end for
  $sig \leftarrow$ signatures of neighbor IDs
  Send ID$[j]$, Bloom filter and sig
end if

Figure 4.6: Packet format for connectivity monitoring.

nectivity and *link asymmetricity*, and it can be obtained in several ways. A node can either estimate locally based on the value of m or obtain from central controller's broadcast message. The simulation in Section 4.5 shows that with hop count vector based localization and with relatively low link connectivity and high link asymmetricity, $2m$ to $2.5m$ is the good approximation on upper bound of u (i.e., in line 18, $\alpha = 2$ to 2.5). We will also apply this settings in our evaluation of large scale sensor networks.

After computing the Bloom filter bit sequence, a check (line 21)[1] is performed to see if it is better to simply send the node IDs directly instead. Otherwise, the Bloom filter sequence and fingerprint of the neighbor IDs are sent to the central controller (line 24-29).

The packet format is shown in Figure 4.6. The two fields (F and BF) are used to identify if the data is encoded using directly neighbor IDs, bitmap, Bloom filter or any other compression schemes. The fields m and $\mathrm{ID}_{\max}$ are used to indicate the number of neighbors seen and largest hop vector distance. The remaining bytes (BFDATA and SIG) are used to store the Bloom filter sequence and fingerprint data.

4.8.2 Connectivity Update

Since the connectivity changes over time, the algorithm may need to be applied periodically. However, if the connectivity of the whole network does not change too much, collecting the connectivity information from scratch periodically may not be a good idea. The common way to perform incremental update is by differential method, i.e., a node will only report to the central controller about what has changed.

The proposed algorithm can be easily extended to this differential update process, as shown in Algorithm 2. For the set of neighbors that has been removed, the original

[1]Note that in the evaluation section, to compare the performance of H^2CM with other techniques like maximal compression of bitmap, this check is not performed.

Algorithm 2 Connectivity Update Algorithm (At Each Node)

Require: Old neighbor table X, added neighbor set X_1, removed neighbor set X_2, hop count vector (or location) information of neighbors, and $thresh$.

$i \leftarrow \arg\max_{i \in X+X1-X2} d_i$
$m_{old} \leftarrow |X|, m_{new} \leftarrow |X + X_1 - X_2|$
$m_1 \leftarrow |X_1|, u_1 \leftarrow \lceil \alpha m_{new} \rceil, b_1 \leftarrow \lceil \log(e) m_1 \rceil$
$m_2 \leftarrow |X_2|, u_2 \leftarrow m_{old}, b_2 \leftarrow \lceil \log(e) m_2 \rceil$
$s \leftarrow \min(|T|, m_{new} \log(|T|))$
$c_1 = \text{BFSequence}(m_1, u_1, b_1, tresh)$
$c_2 = \text{BFSequence}(m_2, u_2, b_2, tresh)$
if $\log(|T|) + b_1(\sum c_1[i]) + b_2(\sum c_2[i]) + 2sizeof(sig) \geq s$ **then**
    Send change of neighbor IDs (or Bitmaps) directly
else
    Allocate space of $b_1(\sum c_1[i]) + b_2(\sum c_2[i])$ bits
    for $i \in [1, len(c_1)]$ **do**
        Do Bloom filtering of each neighbor IDs
    end for
    for $i \in [1, len(c_2)]$ **do**
        Do Bloom filtering of each neighbor IDs
    end for
    $sig_1 \leftarrow$ fingerprint of added neighbor IDs
    $sig_2 \leftarrow$ fingerprint of removed neighbor IDs
    Send ID$[i]$, Bloom filter and sig
end if

neighbor set at the central controller becomes the initial uncertain set U. Each node knows the exact value of $|U|$ and number of neighbors that have been removed, so it may estimate the Bloom filter sequence required with good accuracy. For the set of new neighbors, the initial uncertain set U is the one constrained by the hop vector distance minus the original neighbor set. Same algorithm as introduced in previous sections can be applied.

4.8.3 Further Extensions

The algorithm presented is for a single connectivity threshold. It is possible to extend the approach to monitor discrete link quality values with a small number of discrete levels. For example, to retrieve the link quality information of a node, we can first apply the connectivity monitoring algorithm introduced starting from the lowest link quality. Once the neighborhood information for the lowest link quality is known, we proceed with the next higher link quality by setting the initial uncertain set to be the set of confirmed neighbors in the previous round. The algorithm proceeds till the highest link quality.

4.9 Evaluation

In this section, we show our evaluation results using both simulation and testbed experiments. In the simulations, we assume that the packets can be delivered without any loss. In fact, as long as the hop vectors of sensor nodes are known to the central controller, any subsequent packet losses only affect the information accuracy of the node that initiates the packet, and they do not affect the overall correctness and efficiency of the algorithm. Also note that since energy consumption of sensor nodes is dominated by wireless communication costs, in the simulations, we only consider the communication costs.

λ	[ ]		[1]		[1 1]		[1 1 1]		[1 1 2]	
-	u	m-v	u	m-v	u	m-v	u	m-v	u	m-v
5	19	10	10	6.6	5.3	3.3	2.3	1.4	1.0	0.6
10	46	21	27	15	15	9.8	8.1	5.3	3.7	2.0
20	105	42	64	34	38	23	21	13.9	10	5.9
30	166	63	103	52	57	36	31	22	17	9.2

Table 4.1: Average values of u_i and $(m_i - v_i)$ after applying Bloom filter.

4.9.1 Large Network without Fingerprint Hashing

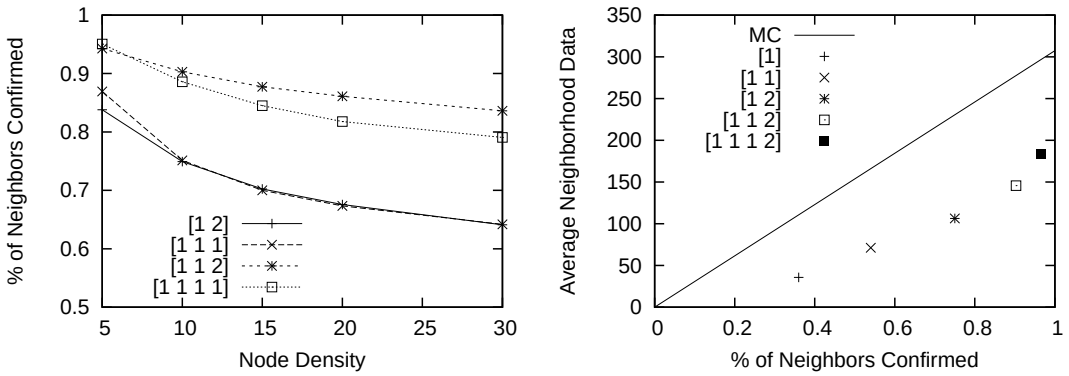
In the first set of experiments, we evaluate the performance of connectivity monitoring in a large network using hop vector distance filtering and Bloom filter. Fingerprint hashing is not performed.

In order to compare the simulation results with the analysis in Section 4.6.3, and to illustrate the performance of different sequences of Bloom filters, we choose to use a fixed sequence of Bloom filter. Therefore, the set of c_j used is the same for all nodes (instead of depending on m as proposed in the algorithm).

We simulate a large network of size 32×32 with node density varying from 5 to 30 (uniform distribution) per unit square. The maximum wireless communication range is normalized to 1. Each node has a unique ID of size 16 bits, which can support a network of size 2^{16}.

We do not take the communication cost of finding hop vector into consideration due to the following reasons. Firstly, it is a fixed cost. After the hop vector distances have been sent to the central controller, as long as the sensor nodes do not move, this hop vector information can be reused for all subsequent connectivity monitoring cycles. Secondly, the cost of hop vector is relative small when the number of nodes to be monitored is large and the cost can be amortized over many monitoring cycles.

While a wide range of link connectivity and asymmetricity have been evaluated, we will only show the result for the setting of link connectivity and link asymmetry equal to 0.8 and 0.2 respectively. The communication cost is set to $\sum_j c_j$ = 0, 1, 2, 3 or 4.



(a) $LC = 0.8, LA = 0.2$ (b) $\lambda = 10, LC = 0.8, LA = 0.2$

Figure 4.7: Performance hop vector and Bloom filter.

Table 4.1 shows the average values of u and $m - v$ for different node density after utilizing the bit patterns generated by different sequence of c_js. The integers within the square braces denote the values of c_j used. Note that [] means no Bloom filter data is utilized (only the hop vector distance based scheme is performed).

It can be observed that for low node density, such as 5, even without any Bloom filter, it is still possible to search based on fingerprint value if computational threshold is set to 10^6. Even for high node density, a Bloom filter sequence of [1 1 2] can still allow the central controller to confirm about 90% of the neighbors (without applying fingerprint hashing).

Figure 4.7(a) shows the results when different combinations of c_j are used with different node densities. The result shows that sequence [1 1 2] performs better than that of [1 1 1 1]. This coincides with the results in Figure 4.7(b). [1 1 1 1] is better than [1 1 2] only when initial uncertain set size u is much larger than number of neighbors m.

Figure 4.7(b) shows the average neighborhood information sent at each node versus the percentage of neighbors confirmed for different Bloom filter sequences. The line (MC) in the plot shows the neighborhood data required to let central controller confirm the same percentage of neighbors using maximal compression. The result shows that by using Bloom filters, the cost is strictly less than (about 50% to 60% of) the cost of maximal compression.

In summary, using only the hop vector distance based scheme and the same Bloom filters settings for all nodes, the central controller cannot obtain the complete neighborhood information when $\sum c_j \leq 4$. However, if one does not require 100% of neighborhood information, hop vector distance based scheme and Bloom filters are able to achieve up to 60% of savings in communication comparing to maximal compression for the same amount of confirmed neighbors. The result will also improve if each node chooses the sequence c_j based on the average number of neighbors, which can be implemented using the same flow shown in 4.8.

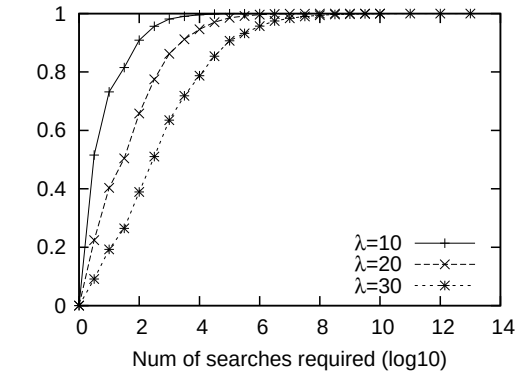
4.9.2 Performance in Large Network

In this simulation, we evaluate the performance of H^{2}CM in large sensor networks. All three techniques are used and the length of the fingerprint used is 32 bits. The network setting is same as previous section.

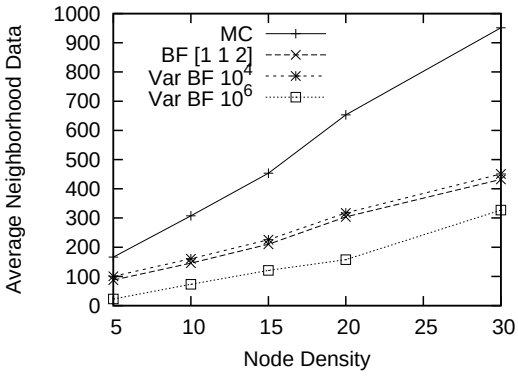
First, as an illustration of the utility of the fingerprint, the cumulative distribution function for number of searches required after the Bloom filter sequence [1 1 2] is applied is shown in Figure 4.8(a). It can be seen that a large portion of nodes (80%) requires little or no additional computational (10^4 or less) even for high node density of $\lambda = 30$. If one allows a search cost limit of 10^6, then for node density of 10 ($> 10,000$ nodes), close to 100% of all neighbors can be found in all our simulations. With node density of 30, less than 5% of nodes will require larger Bloom filter sequence ($\sum_j c_j > 4$).

The communication costs of different connectivity monitoring approaches are shown in Figure 4.8(b). They are maximal compression (MC), fixed c_j sequence of [1 1 2] for all nodes (BF [1 1 2]) and two cases where each node chooses its Bloom filter sequence such that the number of searches required at central controller is smaller than 10^6 and 10^4 respectively. These are labeled as VarBF(10^6) and VarBF(10^4). In the algorithm, we assume that $\frac{u_i}{m_i}$ is 2.5 after hop vector distance filtering.

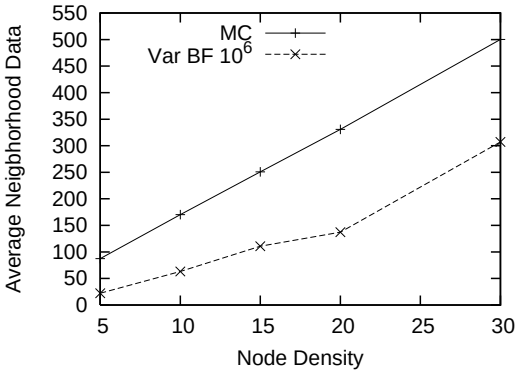
For fixed Bloom filter sequence, the savings is about half the cost of maximal compression. VarBF(10^6) achieves the most savings. At low node density, the savings is up to



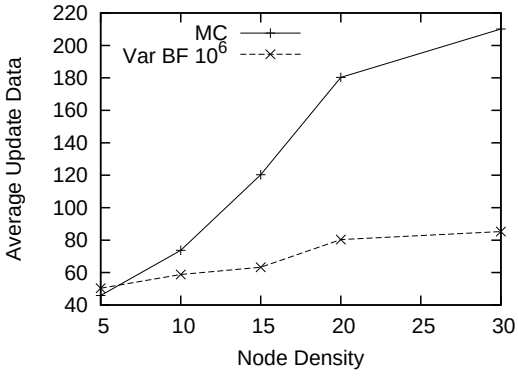
(a) CDF of number searches



(b) Data sent per node (large)



(c) Data sent per node (mid)



(d) Update per node (large)

Figure 4.8: Performance of H^{2}CM in large and midsize networks.

85%. Even at high node density, the reduction is 65%. The improvement of VarBF(10^4) over static sequence is small, indicating that significant search cost may be needed before the Bloom filter sequence can be shortened.

VarBF(10^6) is able to obtain all neighbor information in most cases except for the highest node density. When node density is 30, we observe that for less than 3% of the nodes, the computational cost needed at the central controller exceeds 10^6. Among these nodes with unconfirmed neighbors, the average number of unconfirmed links is 17 out of an average of 66 neighbors.

4.9.3 Performance in Mid-Size Network

In this section, we study the performance of our algorithm in a medium size sensor network using simulation. The network is a 4×4 square and node density varies from 5 to 30. The average total number of nodes in the network is from 80 to 500. As sensor testbeds with hundreds of nodes have been built (e.g. the Kansei sensor testbed [**?**]), networks of such sizes are of practical interest.

Figure 4.8(c) shows the result of average data required at each node for MC and VarBF(10^6). Each data point is an average of 100 runs. 4 beacons are used and fingerprint is 32 bits. The link connectivity used is 0.8 and asymmetricity is 0.2. The result shows that communication cost can be reduced by 40% to 70%. The number of unconfirmed links is very small. However, we observe that, among all the simulation instances, a very small number of nodes wrongly identify their set of neighbors due to collision of fingerprint (6 out of 160,000 cases).

4.9.4 Connectivity Update

In this section, we evaluate the performance of H^2CM for differential update where 10% of the existing links are removed and same number of new links among random chosen neighbor pairs are added. Simulation result is shown in Figure 4.8(d). With VarBF(10^6),

the communication cost varies from about 35% to 45% of the cost using MC at high density. At low node density, the cost of VarBF(10^6) can be higher as only few neighbors have changed. Nevertheless, the total cost is small as well.

4.9.5 Testbed Evaluation

In this section, we present evaluation on a 34 node testbed made up of a combination of Mica2 and Mica2Dot nodes installed in a typical indoor office environment. We show that H^2CM can be efficiently implemented in TinyOS and run in actual deployment using real sensor motes. In our implementation, we use only 80 lines of NesC code and 600 bytes of extra image size (code size).

In the evaluation, 33 nodes sent connectivity information to a single Mica2 mote (central controller) via the collection tree. Link layer packet retransmissions are also enabled to cope with possible packet losses. Since the number of nodes in the network is small, we do not consider hop count information and only apply Bloom filter and fingerprint based hashing. The total data size required per node for H^2CM is 40 bits (21 bit hash), which is the same as using bitmap. Note that sending neighbor IDs directly requires much larger data size compared to H^2CM and bitmap.

Also note that in TinyOS 2.x, each node at most maintains 10 most "useful" neighbors at link layer to save the memory (RAM) and maintenance cost. As a result, the Bloom filter size is always $10\log(e) \approx 15$ bits. Thus, each node sends the number of neighbors (4 bits), one round of Bloom filter of size 15 bits, and a fingerprint of size 21 bits. The total data size is 40 bits, which is the same as using optimal bitmap. Note that sending neighbor IDs directly requires at least 60 bits per nodes because each node ID requires at least 6 bits for 34 node network. Using default identifier size of 16 bits, the cost will be 160 bits instead. at least 60 bits per nodes because each node ID requires at least 6 bits for 34 node network. Using default identifier size of 16 bits, the cost will be 160 bits instead.

We run the experiment over 12 hours and obtained over 4000 snapshots of the overall connectivity. Due to the small fingerprint size, the collision probability of the fingerprint

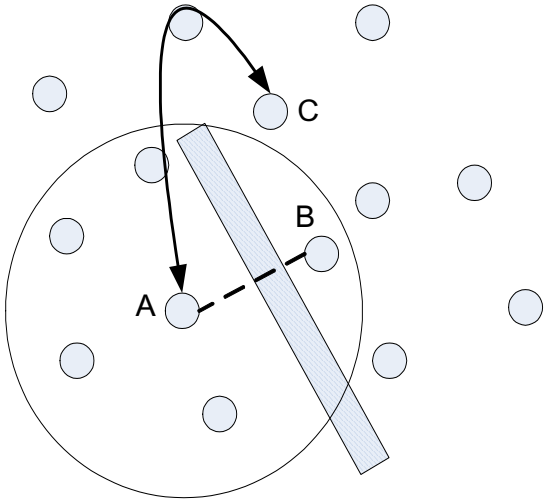


Figure 4.9: Distributed node failure detection.

is about 0.4%. When we increase the data size to 48 bits and set the fingerprint to 29 bits, we do not find any collisions during the experiments.

4.10 A Simple Application – Node Failure Detection

Knowing the connectivity information at the central controller can greatly facilitate various management tasks such as root-cause analysis and protocol debugging. In this section, we present a simple application of connectivity monitoring – detecting node failures in the network.

4.10.1 Node Failure Detection

The simplest approach to node failure detection is to let each sensor node periodically send heartbeat messages to the central controller [81]. Once the central controller does not hear heartbeat messages from a particular node over a period of time, it concludes that the node has failed. A major disadvantage of this approach is that it is not bandwidth-efficient.

Distributed node failure detection algorithms can incur lower communication but require collaboration among neighbor nodes for decision making. However, such ap-

proaches may experience inevitable false positives. An example is shown in Figure 4.9. When an obstacle (the shadowed bar region in the figure) blocks the direct communication link between node *A* and node *B*, node *A* may falsely conclude that node *B* fails. The correct decision can only be made when node *A* collaborates with another node say node *C*, who knows the presence of node *B*. However, nodes *C* and *A* may not be able to communicate directly. Enabling coordination among these potentially disconnected nodes in a distributed manner is a challenging problem. In [85], the authors propose a protocol that each node locally monitors its 1-hop neighbors and the information aggregates along the path to the central controller. However, this approach utilizes the bitmap structure and is not scalable because the packet size will increase linearly with the total number of nodes in the network.

Using connectivity information collected from all nodes using H^2CM, node failure detection can be trivially done at the central controller. However, if all the nodes in the network send their connectivity, the amount of redundant information is excessive. For the purpose of node failure detection, only a small subset of nodes is needed. In the rest of this section, we present an algorithm to select the subset of nodes to send and update their connectivity information to the central controller so that the node failure can be detected efficiently and accurately.

4.10.2 Connectivity-based Node Failure Detection

The proposed algorithm is based on the concept of dominating set. For a communication graph $G(V,E)$ of a sensor network, where V represents the sensor nodes and E represents the direct communication links, a dominating set is a subset of V where each node in V is either in the dominating set or has at least a neighbor in the dominating set. An example of dominating set is shown in Figure 4.10, where the grey nodes belong to the dominating set.

It is clear that for the purpose of node failure detection, only the nodes in the dominating set need to send and update their neighborhood information to the central controller.

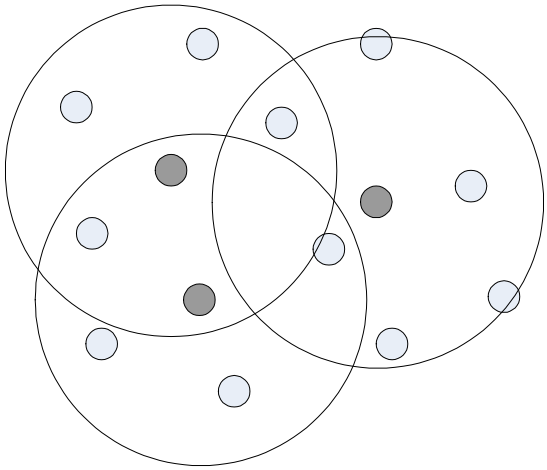


Figure 4.10: Illustration of a dominating set.

The detailed algorithm is discussed below.

Initialization: In the initialization phase, each node distributively elect themselves to join the dominating set. All the nodes in the dominating set send their neighbor table to the central controller, utilizing the H^2CM algorithm proposed. The problem of minimum dominating set is NP-complete. However, finding the minimum is not necessary for this specific application. This is because the network connectivity constantly fluctuates due to the unstable wireless links, and for each time instant (a snapshot) of the network connectivity, the minimum dominating set may contain a large portion of different subset of nodes, which is hard to maintain. Although a minimum dominating set can save the communication cost in initialization phase, it is likely to cost more in maintenance phase.

In this paper, we utilize the simple idea of building a maximal independent set proposed in [25]. A subset of the nodes in G is said to be independent if it does not contain two adjacent nodes. It is maximal if it does not have a proper independent superset. A maximal independent set is also a dominating set. The distributed maximal independent set election algorithm is straightforward and is shown in Algorithm 3.

Each node maintains two states: whether it is a dominator (belongs to the dominating set), and if it is not a dominator, whether it is dominated (has a direct neighbor who is a dominator). If a node decides to join the dominating set, it broadcasts a JOIN message to its direct neighbors to announce that it is a dominator. All its direct neighbors will

Algorithm 3 Dominating Set Initialization

Require: Neighbor table X

1: $dominator \leftarrow flase$, $dominated \leftarrow false$
2: **while** $dominator = false$ and $dominated = false$ **do**
3: $N \leftarrow$ non-dominated neighbors in X
4: **if** $ID < \min\{n \in N\}$ **then**
5: Send JOIN message
6: $dominator \leftarrow true$
7: **end if**
8: **end while**
9:
10: Upon receive JOIN message
11: **if** $dominator = false$ **then**
12: $dominated \leftarrow true$
13: **end if**

mark themselves as dominated. Only a non-dominated node who has smallest ID among all its non-dominated neighbors can nominate itself as a dominator. Note that the algorithm finds an independent set, where no two direct neighbors both elect themselves as dominators.

Dominating Set Maintenance: As wireless links are not stable, a dominated node may temporally lose connection to a dominator, and two dominators may be temporally connected to each other and thus breaking the property of independent set. Under these situations, the dominating set maintenance protocol is needed.

We require that the neighbor table of each node includes one more field: the degree of domination. A node with k distinct direct neighbors who are dominators has a dominating degree of k (k-dominated). This information can be easily exchanged among all 1-hop neighbors. Once more than one dominators become direct neighbors, the dominator with smallest number of 1-dominated neighbors will choose to leave the dominating set. It will broadcast a QUIT message and declare that it is not a dominator any more. Note that after a dominator leaves the dominating set, it is dominated. Its 1-dominated neighbors will become non-dominated.

A dominated node can become non-dominated if it loses the direct connections to all dominators. All non-dominated nodes in the maintenance phase elect themselves for

new dominators so that they are all dominated. The election is based on two criteria: (1) whether those non-dominated nodes have been dominators (and left the dominating set) before; (2) the smallest node ID. The first criterion is to help reduce the communication cost, because for those who have been dominators before, only the update on neighborhood information are required to be sent to the central controller.

Dominators periodically send neighborhood information or neighborhood update to the central controller using the proposed H^2CM protocol (including the update protocol). Note that due to the link instability, some nodes may be in non-dominated states temporally. This will cause temporal false positives in node failure detection. These temporal false positives can be resolved soon because once a non-dominated nodes is dominated (or becomes a dominator), its status will be sent to the central controller immediately. The central controller can effectively reduce the false positive rate by observing over some time period before announcing the failure of nodes.

4.10.3 Evaluation

We simulate a network of size 8×8, where the node transmission range is normalized to 1. Unlike the previous section, packet losses are introduced to indicate link fluctuations. The packet losses over any direct communication pairs are controlled by a uniform random variable (independent geographically) with mean equals to the defined packet loss rate. Each node broadcast "HELLO" messages periodically and if a node does not hear "HELLO" messages from a neighbor over three cycles, it will consider that the link is broken. Note that HELLO message are broadcasted and there is no retransmission. We assume that link level retransmission for unicast is able to deal with the possible message losses. Thus the neighborhood data can still be reliably collected at the central controller.

We compare the communication cost of the proposed protocol to the standard data collection method where each node periodically send heartbeat messages to the central controller. We do not consider the cost of retransmissions because retransmission has same impact on both protocols. The average data generated per node for the standard

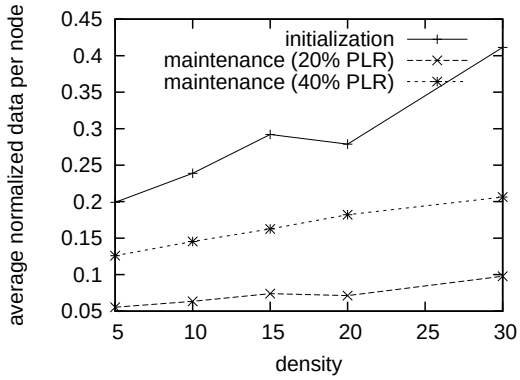


Figure 4.11: Communication cost for node failure detection.

method (each round) is the size of node ID. Therefore, in Figure 4.11, we only show the normalized average data generated per node for the proposed protocol. Note that for H^2CM, the variable size Bloom filter with search limit of 10^6 is utilized.

Since the simulated area is fixed, the number of nodes in the dominating set elected in the initialization phase is about 35-40 nodes for all node densities evaluated.

The communication cost for the initialization phase and maintenance phase are different since in the initialization phase, the dominators are required to send all neighborhood information to the central controller. The average data required for each node is only 20% to 40% of the normal data collection method. In the maintenance phase, the average data per node is much smaller. With 20% of packet loss rate, the communication overhead (including both update data for old dominators and new neighborhood information for new dominators) is only 5% to 10% of the simple heartbeat method. When the packet loss rate is increased to 40%, the average overhead per node is still only 13% to 15% of the heartbeat approach.

When the loss rate is 20%, the instant false positive is only 1.3%. When loss increases to 40%, instant false positive increases to 6.7%. If the central controller announces the failure of a node only when not hearing any information of that node over two cycles of connectivity information gathering, the false positive rate becomes 0.3% and 0.8% for packet loss rate of 20% and 40% respectively.

4.11 Summary

In this chapter, we presented H^2CM that can efficiently monitor connectivity of wireless sensor networks for various sizes. Given estimates of the network size and node density, H^2CM selects one or more techniques to obtain connectivity. Simulation results show that H^2CM works best for large network ($>$ 1000 nodes) achieving savings of up to 85% compare to maximal compression of neighborhood information, even to achieve the complete connectivity information. We also have demonstrated that the algorithm is practical and can be easily implemented on TinyOS with little overhead. Finally, an application of connectivity monitoring is presented.

Chapter 5

Macroscale Topological Hole Detection and Monitoring

5.1 Introduction

A topological hole in wireless sensor networks is a kind of network topology anomaly. It is the phenomenon that the routing path between two nodes is unnecessarily long relative to their physical locations. In the continuous domain, a hole is simply interpreted as a phenomenon that the geodesic path between some pair of two points is not a straight line. The causes of holes include fire, explosion, jamming attacks [5] introduced by intruders or impairment of wireless links due to obstacles.

In this chapter, the problem of dynamic detection and monitoring of macroscale topological holes is investigated. Specifically, we would like to detect the formation of a hole in the network, estimate its size (in terms of breadth and depth defined later) and continuously monitor its transformation (e.g. expansion, contraction, or movement), if any. Knowing the answers to these questions can greatly facilitate decisions related to public safety and network administration. For example, with knowledge of the topological hole, we can quickly gauge the impact (e.g. extent of fire damage), decide if deployment of more sensor nodes is needed, and possibly identify where a jamming attacker is and its

activity. Note that we focus on the detection of *large-scale* topological holes as it may not be worthwhile detecting holes that do not cause significant network changes [100]. Also note that, although the focus in this chapter is on topological holes, as introduced in Chapter 1, in most cases, a large-scale topological hole is "equivalent" to a coverage hole due to the lack of nodes in the same area.

Finding holes in sensor networks has been extensively studied in literature [100, 40, 36]. Most of the work focuses on identifying static holes (e.g., recognizing all the boundary nodes of a static hole). To detect and monitor the dynamics of holes, these protocols have to be run periodically, which is neither cost-efficient nor feasible in real-time because they normally involve many rounds of global message flooding [100, 40]. The approach in this chapter is based on the observation that hole formation creates irregularities in the network connectivity and the changes in the network connectivity contains important information about the hole. The approach is reactive and communication is triggered only when a hole is formed, unlike a polling/sampling based method where communication needs to be performed periodically. In addition, we do not attempt to map the boundary of the hole, which is expensive since many nodes need to be identified. Instead, only a small number of dynamically identified *indicator nodes* are required to report their status to the sink nodes. We believe that this is the first attempt to provide such reactive detection and monitoring mechanism for topological holes.

The main contributions in this chapters are as follows. (1) An approach to detect holes dynamically based on only connectivity changes is presented. (2) The topological properties of the *indicators nodes* are identified. How indicator nodes can be locally elected efficiently is also shown. The proposed algorithm only involves the "local" nodes around the hole and thus the communication cost is small compared to global message flooding. (3) Algorithms on identifying the type of hole transformation (e.g. expansion, contraction or movements), and estimating the hole (or a snapshot of a transforming hole) based on connectivity changes detected by the indicator nodes are proposed. (4) Lastly, some additional properties of indicator nodes are shown. How these properties can be

used to estimate the hole size even without localization information is proposed.

5.2 Simple Hole Detection

In this section, a simple hole detection algorithm based on *changes* in connectivity is presented. Hence, if a hole is already present in the initial deployment, the algorithm will not be triggered and existing hole detection algorithms [100, 39] will be needed.

5.2.1 Network Connectivity Model

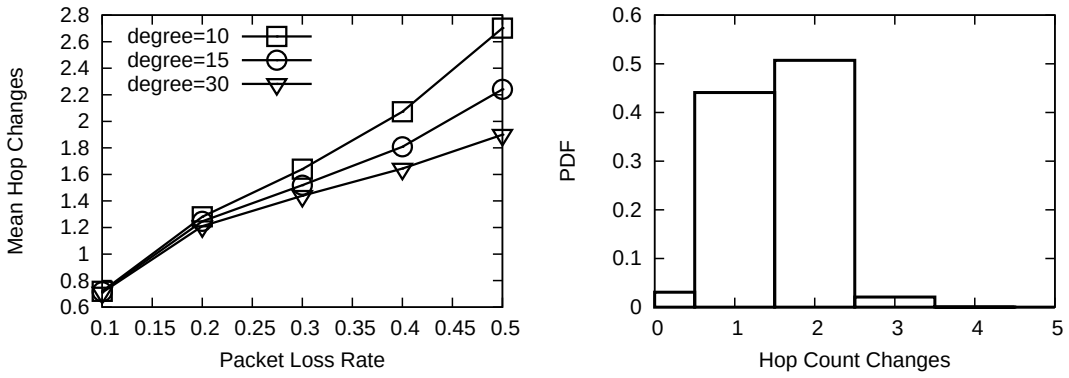
A large number of sensor nodes is assumed to be deployed in a region. During the initial deployment, network connectivity information is distributed in the form of n ($n \geq 1$) shortest path trees rooted at n *source nodes* (or anchors). The source nodes should be well-separated from each other. They can be centrally allocated or distributively selected using the proposed algorithms in [61].

Each node locally maintains a hop count vector to the n source nodes, and periodically broadcasts this hop count vector to its neighbors. These messages can be embedded in the "hello" messages required for link maintenance and thus incur minimum extra communication cost. For example, for a system with n source nodes and limited to 255 hops or less, the extra data required for each "hello" message is n bytes. The "hello" message broadcast interval is T_{hello}. A node switches parent when it does not hear from its current parent (on the tree rooted from a particular source node) after a timeout value of T_{pto}.

For the time being, we will assume that one and only one hole forms and stays static afterwards. The problem of dynamic hole and multiple holes will be addressed later.

5.2.2 Connectivity Based Hole Detection

Intuitively, when a hole forms, connectivity information maintained at the sensor nodes changes. By letting the sensor nodes observe their own connectivity changes, the formation of the hole can be detected.



(a) (b)

Figure 5.1: Hop count changes versus link fluctuations.

The basic hole detection mechanism is very simple. If a node finds that its hop count to any source node increases by at least a threshold value H, the node concludes that a hole has formed and sends a message to the sink nodes.

The first question that arises is: *what should the value of H be?* The appropriate threshold depends on temporary link quality fluctuations that result in changes of the connectivity information over time. We determine this value through simulation.

The simulation result for average maximum hop count change seen by nodes due to link quality fluctuations, which results in packet losses, are shown in Figure 5.1(a). In the simulations, T_{pto} is set to $3T_{hello}$, and the node density and packet loss rate are varied. Figure 5.1(b) shows the distribution of maximum hop count changes for all the nodes for the scenario where the average node degree is 15 and packet loss rate is 30%. Simulation time is 3000 seconds.

The simulation results show that the maximum hop count change for any node in a large network affected by the link fluctuations is small (only 1 or 2 hops for most nodes).

There is a trade-off in selection of the threshold value. If it is too small, there will be a lot of false positives, and if it is too large, the algorithm can only detect relatively large holes. We set the hop count change threshold H to be **5**. This is the value at which it is unlikely to have false positives in hole detection, and where the size of the hole starts to have significant impact on the network (a packet has to be transmitted at least 5 more hops

to go around the hole towards some source node). A smaller threshold of 4 or even 3 can be used if some false positives can be tolerated and detection of smaller holes is required.

This simple hole detection is obviously insufficient. There can be many nodes that detect changes in hop counts more than H. In the rest of the paper, we will introduce the idea of indicator nodes and how they can be utilized.

5.3 Indicator Nodes and Their Properties

When a hole forms, many nodes in the network detect changes in hop count. Letting all of them send information to the sink nodes is expensive.

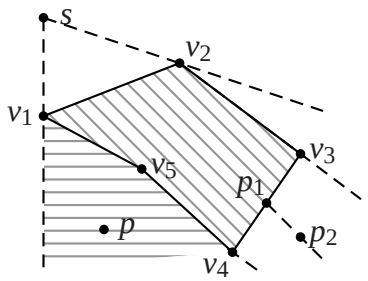
Unlike many previous protocols [100, 40, 36, 58, 24] that try to identify all nodes on the boundary, we only require a few nodes to convey information about the hole to the sinks. We call these nodes the *indicator nodes*. This section describes the topological properties of these nodes and explain why they are unique and important.

Definition 5.1 *After the formation of a new hole, the set of* ***indicator nodes*** I_i *(*$i \in [1,n]$*) are the nodes that have the largest changes in hop counts relative to a source node* s_i.

In the rest of this section, we show several important properties of indicator nodes which will be useful later. In particular, we show that (1) an indicator node must lie on the boundary of a new hole, (2) the convex hull of all indicator nodes provides a lower bound on the convex hull of the hole.

These properties will be discussed and proven in the continuous domain, in which the indicator nodes will be referred to as the *indicator points*. With sufficient node density, the properties of indicator points in continuous domain can be approximated to sensor deployment in discrete domain. The properties of indicator points in continuous domain can be viewed as an approximation to sensor development in discrete domain with sufficient node density.

First, we introduce some basic definitions in computational geometry in the continuous domain [71].



(a)

Figure 5.2: Illustrations in continuous domain.

5.3.1 Definitions and Preliminaries

Let $\mathcal{F}$ be a closed polygonal space in a 2-D plane referred to as *free space* and let $s \in \mathcal{F}$ be a point called the source point. Let O consists of m (open, bounded) simple polygonal obstacles/holes. In the rest of the section, we assume $\mathcal{F}$ to be unbounded and O is the complement of free space $\mathcal{F}$. We will focus on polygon holes as many other shapes can be approximated as polygons. Let V denotes the set of vertices in $\mathcal{F}$. V also denotes the vertices of the boundary of the inner holes/obstacles O. This is shown in Figure 5.2(a) where O is the shadowed area enclosed by v_1 to v_5 and V is the set $\{v_1, v_2, v_3, v_4, v_5\}$.

Geodesic path is defined as the shortest obstacle-avoiding path. Let $\pi(p,q)$ denote a geodesic path from a point p to a point q, where $p,q \in \mathcal{F}$. Let $l(p,q)$ be the length of $\pi(p,q)$ and $|pq|$ be the Euclidean length between p and q. In Figure 5.2(a), $\pi(s,p)$ is the path $\{s, v_1, p\}$.

The point r is *root* of p if for some geodesic path $\pi(s,p)$, r is the last vertex along $\pi(s,p)\backslash\{p\}$ at which $\pi(s,p)$ turns. The set of all roots of p is denoted by $\mathcal{R}(p)$. In Figure 5.2(a), v_1 is the root of p and $\mathcal{R}(p) = \{v_1\}$.

The *Shortest path map*, SPM(s,O), is a partition of $\mathcal{F}$ into maximal regions (called cells) that correspond to sets of points with the same root or set of roots with respect to s. More formally, SPM(s,O) is the partitioning of $\mathcal{F}$ into cells $C(\mathcal{R}) = P\{x \in \mathcal{F} \,|\, \mathcal{R} = \mathcal{R}(x)\}$ corresponding to subsets $\mathcal{R} \subseteq V \bigcup \{s\}$.

If $\mathcal{R} = \{v\}$ is a singleton, it is easy to show that $C(\{v\})$ is two-dimensional (i.e., a

region in the plane) and connected. As shown in Figure 5.2(a), $C(\{v_1\})$ is the area shaded by horizontal lines, i.e., all points in that area have same root v_1. If $\mathcal{R} = \{v_i, v_j\}$ is a pair, then one can show that $C(\mathcal{R})$ is one-dimensional (i.e., a curve) and possibly disconnected.

We call $C(\{v_i, v_j\})$ the *bisector* of vertices v_i and v_j. The intersection of the bisector and the boundary of a hole is called the *bisector point*. In Figure 5.2(a), the curve that contains p_1 and p_2 is the bisector of v_3 and v_4, i.e., all points along the curve have same root set $\{v_3, v_4\}$. p_1 is the bisector point. If $\mathcal{R}$ has cardinality of at least three, then $C(\mathcal{R})$ is either empty or a single point, called an *SPM-vertex*.

Finally, two results that will be useful later are stated below

- Each bisector is the union of a finite set of closed subarcs of a common hyperbola. (A straight line is considered to be a degenerate case of a hyperbola.) *There is at least one bisector point on the boundary of each obstacle* [71].
- Shortest path from s to any point in $\mathcal{F}$ among the set of polygon obstacles O is a polygonal path whose inner vertices are vertices of O [26].

5.3.2 Properties of Indicator Points

Let O and $\mathcal{F}$ be the initial hole space and free space respectively, and O' and $\mathcal{F}'$ be the new hole space and free space after a new hole o' is formed. We assume there is only one new hole and leave the discussions on more than one new hole in Section 5.7. We further assume that o' does not intersect with any of the existing holes in O.

Definition 5.2 *For any point p in $\mathcal{F}'$, the geodesic distance change of p relative to a source point s upon the formation of the new hole is $l_{\mathcal{F}'}(s,p) - l_{\mathcal{F}}(s,p)$. The **indicator points** are defined as the points with largest change in geodesic distance among all points in $\mathcal{F}'$.*

Theorem 5.1 *Upon formation of a new hole o', the indicator points must lie on the boundary of o'. If o' is a convex polygon, the indicator points are also the bisector points of o'.*

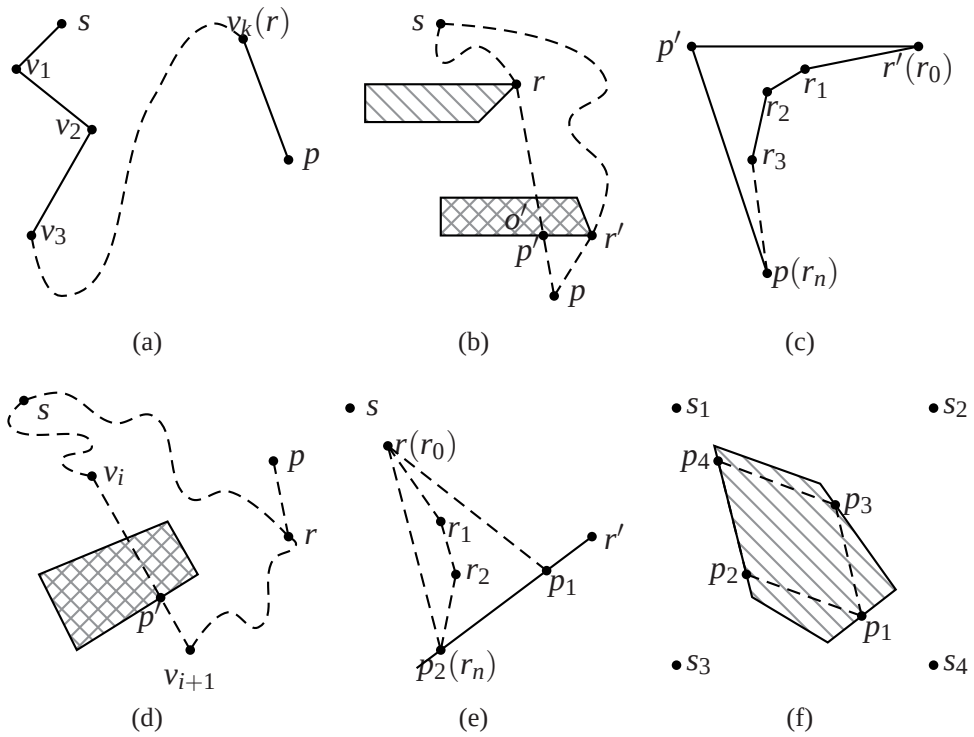


Figure 5.3: Proof of Theorem 5.1.

Proof: As shown in Figure 5.3(a), let s be an arbitrary source point and the geodesic path $\pi_{\mathcal{F}}(s,p)$ passes through v_1, v_2,..., and v_k. We also use r to represent v_k as it is the root of p in $\mathcal{F}$. Assume that the geodesic distance of p to s changes due to the new hole o', i.e., o' "cuts" $\pi_{\mathcal{F}}(s,p)$ at some places.

The basic idea here is to prove that for any such point p, it is always possible to locate a point p' on the boundary of the hole o' such that p' has a larger geodesic distance change compared to p.

(Case 1) We first look at the situation when o' has direct impact on the geodesic distance of p, i.e., the last segment in the path $\pi_{\mathcal{F}}(s,p)$ v_kp is blocked by o'. Let r (the point v_k) be the root of p in $\mathcal{F}$. Let p' be the intersection point of the boundary of o' and the segment rp such that p' is closest to p along rp, i.e., pp' is fully in $\mathcal{F}'$. Let r' be the root of p' in $\mathcal{F}'$. An example is shown in Figure 5.3(b), where r is the root of p in $\mathcal{F}$ and r' is the root of p' in $\mathcal{F}'$.

The geodesic distance from s to p in $\mathcal{F}$ is given by definition

$$l_{\mathcal{F}}(s,p) = l_{\mathcal{F}}(s,r) + |rp|. \tag{5.1}$$

Since p' is on the segment rp, it is easily seen that the geodesic distance from s to p' is given by

$$l_{\mathcal{F}}(s,p') = l_{\mathcal{F}}(s,r) + |rp'|. \tag{5.2}$$

Similarly, the geodesic distance from s to p' in space $\mathcal{F}'$ is

$$l_{\mathcal{F}'}(s,p') = l_{\mathcal{F}'}(s,r') + |r'p'|. \tag{5.3}$$

For the geodesic distance from s to p in $\mathcal{F}'$, we consider the following two cases. When $r'p$ is also fully in $\mathcal{F}'$, i.e., neither the original holes in O, nor the new hole o' intersects with the segment $r'p$,

$$l_{\mathcal{F}'}(s,p) \leq l_{\mathcal{F}'}(s,r') + |r'p|. \tag{5.4}$$

We then have,

$$\begin{aligned} & (l_{\mathcal{F}'}(s,p') - l_{\mathcal{F}}(s,p')) - (l_{\mathcal{F}'}(s,p) - l_{\mathcal{F}}(s,p)) \\ \geq\ & |r'p'| - |rp'| + |rp| - |r'p| \\ =\ & |r'p'| + |pp'| - |r'p| \geq 0 \end{aligned} \tag{5.5}$$

The last inequality is given by triangle inequality. Equality holds only when p is the same as p', i.e., p is on the boundary of o'.

The second case is when $r'p$ is not fully in $\mathcal{F}'$, i.e., it is either blocked by the holes in original hole space O, or by some parts of the new hole o', or both.

Since both $r'p'$ and pp' are in $\mathcal{F}'$, one of the obstacle avoiding path (needs not to be the minimum) from r' to p is shown in Figure 5.3(c). This path can be constructed by

going around the *convex hull* of all the obstacles inside the triangle $r'p'p$. In this case,

$$l_{\mathcal{F}'}(s,p) \leq l_{\mathcal{F}'}(s,r') + \sum_{i=1}^{n} |r_{i-1}r_i|. \tag{5.6}$$

We then have,

$$\begin{aligned} & (l_{\mathcal{F}'}(s,p') - l_{\mathcal{F}}(s,p')) - (l_{\mathcal{F}'}(s,p) - l_{\mathcal{F}}(s,p)) \\ \geq & \ |r'p'| - |rp'| + |rp| - \sum_{i=1}^{n} |r_{i-1}r_i| \\ = & \ |r'p'| + |pp'| - \sum_{i=1}^{n} |r_{i-1}r_i| \geq 0 \end{aligned} \tag{5.7}$$

The last inequality in Equation 5.7 can be proven in many ways. The simplest intuition behind (also used in [71]) is to consider Figure 5.3(c) and by imagining an elastic rubber band that is initially around three nails on the board at r', p' and p. If the p' is removed, the length of the rubber band will "shrink" to the nails around $r_1, \ldots, r_n$. Again, the equality only holds when p is the same as p'.

From Equations 5.5 and 5.7, we can see that for any point p (not on the boundary) that the last segment $v_k p$ or rp is blocked by o', there is always a point p' on the boundary of o' that has larger geodesic change than p.

(Case 2) If the hole o' does not block $v_k p$, it must intersect at some other places with $\pi_{\mathcal{F}}(s,p)$. Assume p' is the closet intersection point to p along path $\pi_{\mathcal{F}}(s,p)$, and assume p' is on segment $v_i v_{i+1}$. This is shown in Figure 5.3(d).

The geodesic distance in $\mathcal{F}$ of point r and p are $l_{\mathcal{F}}(s,r)$ and $l_{\mathcal{F}}(s,p) = l_{\mathcal{F}}(s,r) + |rp|$ respectively. Similarly, the geodesic distance of r and p in $\mathcal{F}'$ are $l_{\mathcal{F}'}(s,r)$ and $l_{\mathcal{F}'}(s,p) \leq l_{\mathcal{F}'}(s,r) + |rp|$ respectively. The last inequality is based on the fact that the segment rp is not blocked by any hole.

Thus,

$$(l_{\mathcal{F}'}(s,r) - l_{\mathcal{F}}(s,r)) - (l_{\mathcal{F}'}(s,p) - l_{\mathcal{F}}(s,p)) \geq 0 \tag{5.8}$$

It can be seen that the geodesic distance change of point r (v_k) is at least as large as p. Similarly, the geodesic distance change of v_{k-1} is at least as large as v_k, and so on. We have already proven that the geodesic distance change of p' is larger than the geodesic distance change of v_{i+1}. Thus p' has a larger geodesic distance change than p.

For a convex hole, as shown in Figure 5.3(e), assume p_1 and p_2 are two points on the same edge of the convex hole o', and both p_1 and p_2 are affected by o'. Further assume that p_1 and p_2 lie on the same side of the bisector introduced by o'. Since o' is a convex hole, p_1 and p_2 must have the same root r' which is one of the vertex of o'. Briefly, using similar techniques as above, it can be shown that the geodesic distance change of p_1 is smaller than p_2. □

Corollary 5.1 If a point p moves farther away from the boundary of o' along the direction of the last segment of geodesic path $\pi_{\mathcal{F}}(s,p)$, the change in geodesic distance becomes monotonically smaller.

This is a natural extension of Theorem 5.1. Intuitively, when a point is farther away from a hole, the impact of the hole on that point is smaller.

For n source points, there are n sets of indicator points (I_1,...,I_n). These indicator points provide a natural size estimate for the hole. D

Theorem 5.2 *The convex hull of all the indicator points in I_1 to I_n gives the lower bound on the convex hull of the hole. If the hole is a convex hole, this polygon lower bounds the hole itself.*

Proof: The set of indicator points are on the boundary of the hole. The convex hull of all the indicator points must locate inside the convex hull of the hole. If the hole is a convex polygon, the convex hull of the hole is the boundary of the hole. Therefore, for a convex hole, the convex hull of the indicator nodes must locate inside the hole boundary. □

One example is shown in Figure 5.3(f), the convex hole is denoted by the shadowed polygon. For 4 source points from s_1 to s_4, the 4 indicator points p_1 to p_4 are on the boundary of the hole. The convex polygon bounded by p_1 to p_4 is the lower bound of the hole.

Corollary 5.2 Given a source point *s*, there is one and only one indicator point for a convex hole.

Proof: For a new convex hole and a given source point *s*, there must be some part on the boundary of the hole that is not affected by the hole, i.e., their geodesic distance does not change after the hole is formed. As proven in the proof of Theorem 5.1, if we go either clockwise or counterclockwise along the boundary of the convex hole from this part, the geodesic distance change increases. Since the change of geodesic distance is continuous, according to intermediate value theorem, there must be one and exactly one crossing point on the boundary, which is the indicator point. □

Corollary 5.2 coincides with one conclusion drawn in [71]: there is at least one bisector point on the boundary of each obstacle. Note that the "bisector point" in [71] is slightly different from our definition, it may also be the intersection of bisectors caused by other holes on the boundary of the new hole. Therefore our conclusion of "one and only one" does not conflict with the "at least one" finding.

In the next few sections, we will illustrate how indicator nodes can be dynamically identified and used.

5.4 Indicator Node Election and Hole Detection

5.4.1 Indicator Node Election

When a node detects the presence of a hole as indicated by sufficient hop count change (see Section 5.2), it will enter *indicator node election* phase. In this phase, each node locally maintains the maximum hop count changes to the *n* source nodes in the network.

A node will also broadcast *indicator election update messages* to its direct neighbors when (1) it just enters the indicator election phase, (2) it has new updates on hop counts or maximum hop count changes it knows to any of the source nodes, or (3) the *indicator election update* timer with period $T_{ieupdate}$ fires. The conditions are used to control the speed of the election process and also deal with possible message losses. The indicator election message includes a node's own hop counts to the n source nodes, as well as the maximum hop count changes it knows so far. The size of such a packet is $2n$ bytes (assuming hop count does not exceed 255).

When a node does not receive any new updates after T_{ieto} seconds and believes it has the maximum hop count change in its neighborhood with respect to some source node, it declares itself an indicator node.

From Theorems 5.1, such a node always exists and can be determined locally because hop count change is continuous. There is a tradeoff in selection of the value of T_{ieto} between speed of detection and false positives of early detection. If it is too large, election time is long, which will cause large delay. If it is too small, many nodes may prematurely declare themselves as the indicator nodes, causing unnecessary false positives.

Once an indicator node is elected, it will send its initial and final hop counts relative to all n source nodes to the sinks. Its immediate neighbors can optionally suppress their own messages even if they are indicator nodes. If the n source nodes also act as multiple sinks, the elected indicator node can smartly send the information to the sink that has least hop count change (avoiding holes). Note that when nodes prematurely declare themselves as the indicator nodes, there will be false positives and extra overhead. However, this does not affect the result as long as messages from the actual indicator nodes are received by the sink nodes. The sink nodes can filter out the false positives easily. One can either let the indicator nodes wait before they send final hop count changes, or let the indicator nodes send their hop count changes whenever there is some updates.

If a node finds that it has neighbors who have larger hop count changes than itself, it will quickly enter inactive state until it receives new updates. Once an inactive node does

not receive any update for some time, it will exit the indicator election phase.

5.4.2 Hole Detection

In Figures 5.4(a) to 5.4(d), we illustrate four cases of applying the indicator election algorithm using 4 source nodes for different hole shapes. The network consists of 4300 nodes randomly placed in a unit square of size 30×30 (unit is maximum communication range). The average node degree in these examples is 15. The numbers on the 4 corners show the location of the source nodes and numbers on the boundary of the hole show the location of the elected indicator nodes corresponding to the appropriate source nodes.

It can be observed that for all the holes (a circle, a normal convex polygon, a line barrier and a concave polygon), the indicator nodes are on the boundary of the hole. While not obvious from the figures, the nodes are indeed the nodes that change most in hop counts in the whole network.

In Figure 5.4(c), there are no indicator nodes for source nodes 1 and 3 because no node in the network has hop count change exceeding the threshold 5 with respect to these source nodes. Figures 5.5(a) and 5.5(b) are the enlarged region enclosed by a square in Figures 5.4(b) and 5.4(d) respectively. It can be seen that the indicator nodes are also close to the bisector (shown as a curve in Figure 5.5(a)) for a convex hole.

5.4.3 Delay and Communication Cost

Using a circular hole placed in the middle of the network, we measure the average time and total message overhead needed to elect the indicator nodes. There are 4 source nodes at the corners and the average node degree is set to 15.

The time taken to identify an indicator node, after a hole is formed, is dominated by the following timers/time intervals: (1) T_{pto}, timeout before switching parent; (2) T_{hello}, hello packet interval; (3) $T_{transmit}$, average per-hop transmission time; (4) $T_{ieupdate}$, interval for rebroadcasting the indicator election update message; and (5) T_{ieto}, interval before

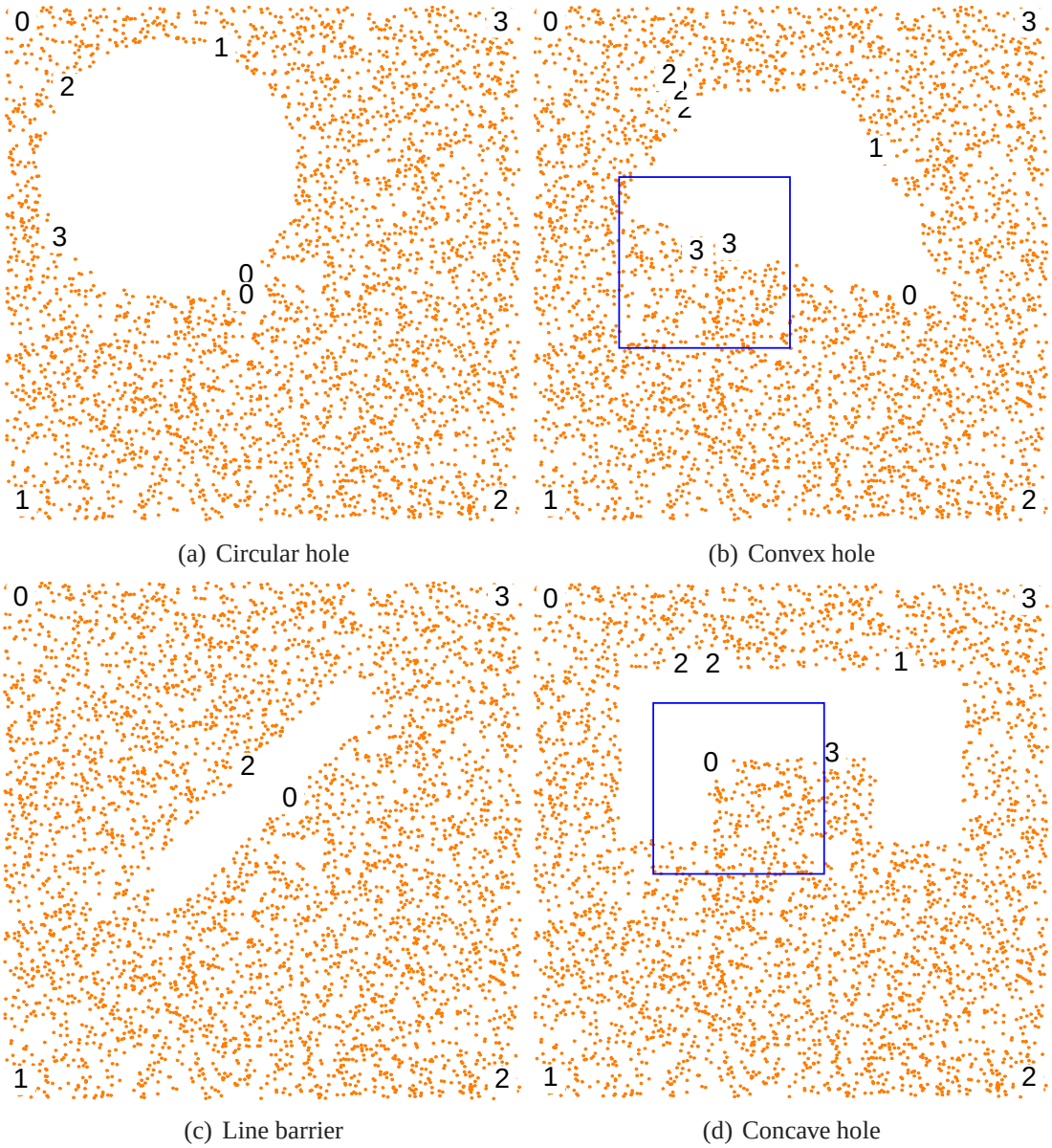
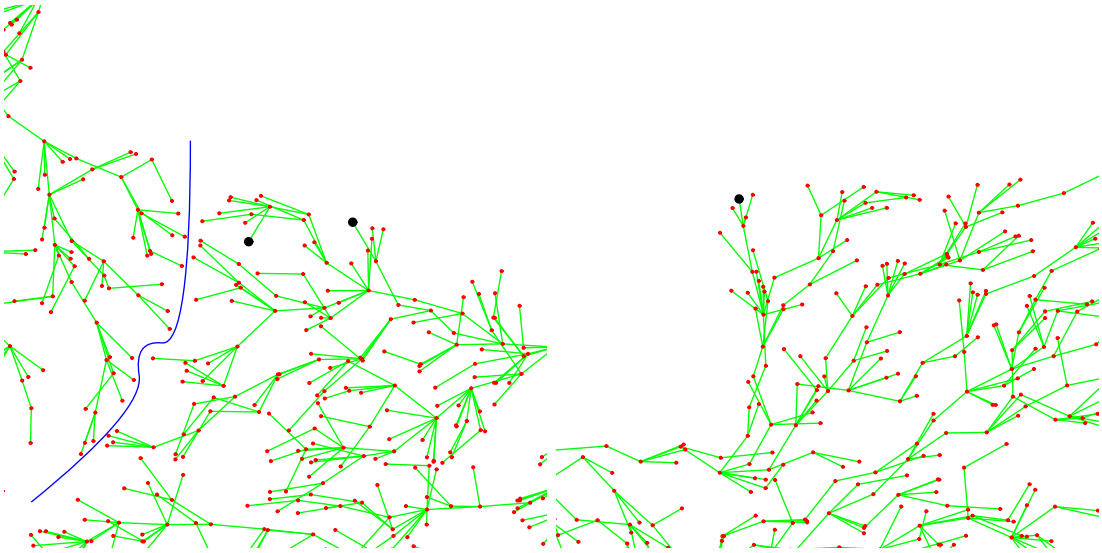


Figure 5.4: Holes and indicator nodes elected for different holes.



(a) Indicator nodes of the convex hole in Figure 5.4(b)

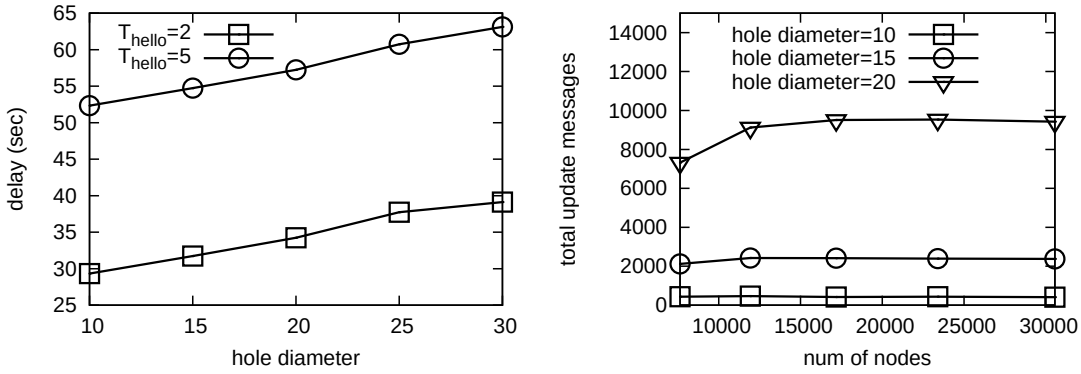
(b) Indicator nodes of the concave hole in Figure 5.4(d)

Figure 5.5: Locations of indicator nodes. Blue line shows the bisector cut.

declaring as the indicator node. In the simulations, we set $T_{hello} = 2s$ and $T_{hello} = 5s$. The values of T_{pto} is always 3 times T_{hello} as stated before. The values of $T_{ieupdate}$ and T_{ieto} are set to 1 and 6 seconds respectively. The $T_{transmit}$ is set to $2ms$ which is about the packet transmission time of a MicaZ mote.

The average delay is shown in Figure 5.6(a). We can see that the delay increases slowly and linearly with the size of the hole (for a circle hole example, the size is determined by its diameter). Due to the immediate update policy in indicator election phase, the delay is dominated by the T_{hello} in hole detection phase whereby a node has to wait for sufficient change in connectivity. Once the hole is detected, the indicators will be elected quickly.

The communication cost of the algorithms is only affected by the size of the hole. A larger size hole can cause more nodes to enter *indicator election* phase. It can be seen in Figure 5.6(b) that the total communication cost is almost invariant to number of nodes in the network or the size of the network (for a fixed node density). The cost only increases with the size of the hole. The total number of nodes that enter indicator election phase are about 150, 810 and 2100 for hole diameter (measure in unit of maximum transmission



(a) Delay vs. hole diameter (b) Overhead vs. num of nodes

Figure 5.6: Delay and communication cost

range) of 10, 15 and 20 respectively. The average number of messages sent for an *active node* is thus only about 3 to 5 messages. Overhead is low because the majority of the nodes will quickly enter inactive state.

Normalized to the total number of nodes for the largest network simulated (network size of 80×80), the overhead for the hole diameter of 20 is only 0.3 message per node per detection. Therefore, compare to other approaches based on topology method (e.g. [100, 40]) which require multiple rounds of message flooding, our approach is much more efficient.

5.5 Continuous Indicator Node Election and Its Application

Each node has to continuously monitor the hop count changes throughout the whole hole-monitoring period. Optimization techniques such as logging only the key event points can be adopted, but this is not our main purpose of this paper. We simply assume that each node has enough memory to log its hop count changes over very long time.

In the previous section, it is assumed that a new hole forms and stays static afterwards. Examples of such holes include jamming holes caused by intruders [5], sudden failure of a large potion of the sensor nodes, or a suddenly appearing obstacle. However, many other types of holes are dynamic in nature, e.g., holes caused by spreading of fire,

or by a moving jamming attacker. A hole can expand, contract or move (i.e., hole transformation) continuously. Such hole dynamics can be monitored by continuous indicator node election.

5.5.1 Continuous Indicator Node Election

While the basic election algorithm is similar to the one proposed in the previous section, two new issues have to be addressed for the case of continuous connectivity changes.

First, different delays are incurred in locating the indicator nodes corresponding to different source nodes for a particular *hole instance* (i.e., the snapshot of the hole at a particular time). Messages from different sets of indicator nodes corresponding to different hole instances will be interleaved when they reach the source nodes.

In order to solve this problem, the sink has to synchronize the messages, by knowing which messages correspond to which hole instance. In our approach, this is accomplished by adding a round number to the *indicator node election message*. For each indicator node elected, the indicator node will increase the round number. Whenever a node finds that it is using a round number smaller than what its neighbor is broadcasting, the node will update its round number. The sinks will then relate the events using the round numbers.

The second issue arises because it takes time to detect an indicator node and some indicator nodes may not be detected if the transformation is too fast relative to the detection time.

In order to address the second problem, the indicator node election needs to be "fast enough". Therefore, the indicator node detection period should be less than the time it takes the hole to expand/contract/move by one average hop distance. This is because, from the time that the original parent of one real indicator node is "destroyed", to the time that the indicator node is elected and its hop counts to the source node are fully updated, the indicator node cannot be "destroyed". In order to accurately monitor holes with a faster transformation speed, the delay for indicator node election needs to be reduced correspondingly by reducing values of $T_{ieupdate}$ and T_{ieto}. With the default values

($T_{ieupdate} = 1s$ and $T_{ieto} = 6s$), the maximum hole transformation speed that the system can monitor is about 0.1 average hop count per second. If these values are reduced to $T_{ieupdate} = 0.2s$ and $T_{ieto} = 1s$, the maximum hole transformation speed is about 0.4 average hop count per second. For a typical wireless communication range of 50 meters, 0.1 average hop count corresponds to a hole transforming speed of $18km/h$, and 0.4 average hop count corresponds to a hole transforming speed of $72km/h$.

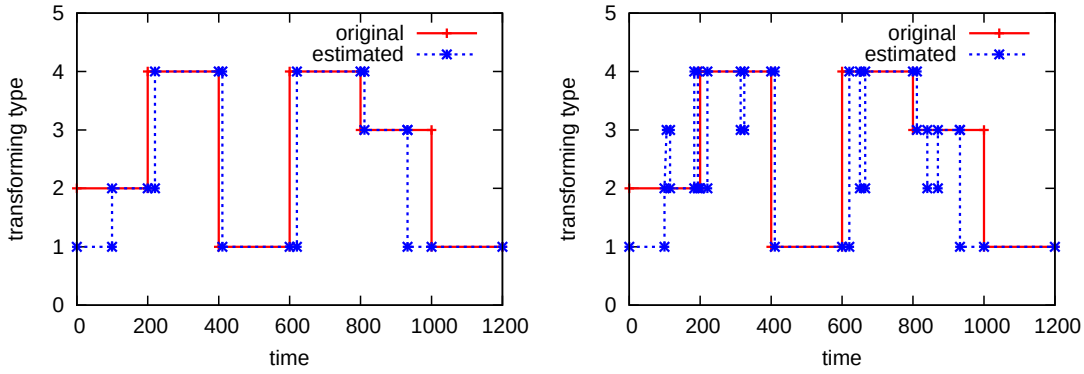
Finally, it is interesting to note that if we expand the definition of indicator nodes to include nodes whose hop count change is 1 hop less than (or equal to) the maximum hop count change, a much faster hole transformation speed can be supported since many more nodes would report their connectivity. This is of course at a cost of higher communication overhead and lower estimation accuracy as well.

5.5.2 Hole Transformation Application

Continuous election of indicator nodes can be used to track transformation of holes over time. We assume that there are only three possible transformations, namely expanding, contracting and moving. Furthermore, only one transformation may occur at any time. An example of hole expansion is the case when fire spreads and the sensor nodes are destroyed. An example of contraction or movement of a hole is the case where an interferer varies its power or move.

The transformation type identification algorithm is simple. If the original hop counts of the elected indicator nodes relative to their source nodes are decreasing (when the round number increases) for all n source nodes, the hole is expanding; and vice versa. If the original hop counts of some indicator nodes relative to their source nodes are increasing, and some are decreasing (when the round number increases), the hole is moving.

Note that it is possible to estimate the velocity of hole transformation by observing the location (locations can be estimated through connectivity information) changes of different indicator nodes at different rounds. The value of the velocity is estimated by finding the location differences at different rounds, and the angle of the velocity is estimated by



(a) Circular hole (b) "Irregular" hole

Figure 5.7: Transformation type identification

the direction of location changes at different rounds.

5.5.3 Evaluation

In the evaluation, we set the average node degree to be 15, the indicator election update timer to be 1*s* and hole transformation speed to be 0.05 of average communication range per second (this is about 0.06 hop per second). In the simulations, change in hole transformation type occurs every 200s. The goal is for the sink to identify what type of transformation is occurring quickly.

Figure 5.7(a) shows the transformation of a circle hole. The red solid line shows the original transformation type and the blue dotted one shows the estimation at the sink. The numbers representing the states of the hole are 1 for idle, 2 for expanding, 3 for contracting and 4 for moving. It can be clearly seen that except for a large initial delay, the sink has an accurate view on what type of transformation the hole is doing. Measured in terms of correctness over time, the accuracy is about *94%* (without considering the initial and final phases).

Figure 5.7(b) shows the case for an irregular hole where initially a circle is divided into eight (45°) sectors and each sector experiences different speed of expanding/contracting (0.04 to 0.06 of average communication range per second) or moving (0.05 of average communication range per second). We can see that even with sudden changes in the indicator nodes from one sector to the adjacent sector, the average accu-

racy is still about *88%*.

5.6 Hole Estimation Using Indicator Nodes

5.6.1 Estimation with Localization Information

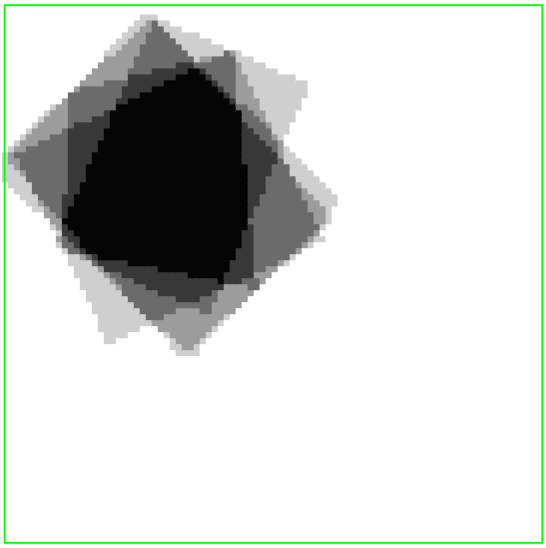
A final application of the indicator node is its use to estimate hole location and size. This estimation works for both static and dynamic holes, as long as the indicator nodes for the hole or hole instance can be successfully elected.

In order to provide information on hole location and size, some form of localization information is needed. In [74], a simple localization method based on only connectivity information (hop counts) is proposed. In [62], Li et al. presents a method on connectivity-based localization in the presence of holes. These work provides an accuracy of 50% to 100% of communication range, which is acceptable given that the hole to be monitored is relatively large. In this section, we will utilize such localization schemes for indicator node location estimation. If the locations of source nodes are known, they can be used as localization beacon nodes as well.

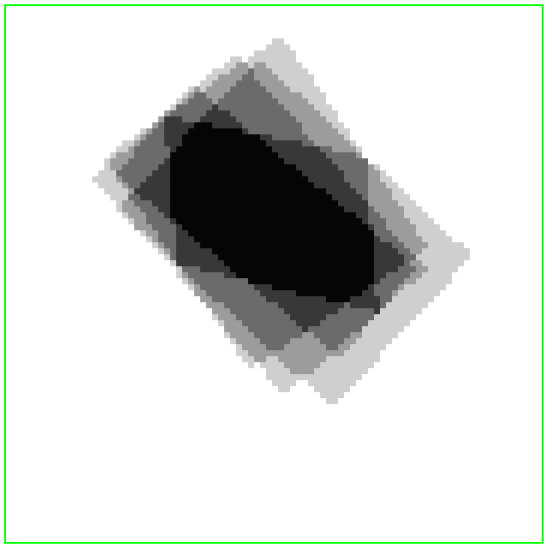
The proposed hole estimation algorithm is based on two factors. First, from Theorem 5.2, the convex hull of indicator nodes provides a lower bound on the convex hull of the hole. Second, the hop count changes of indicator nodes can also be used to estimate the hole. The second factor is based on the intuition that when a hole is larger, the hop count changes of its indicator nodes are also larger.

We propose a grid based algorithm to integrate both factors. The area is divided into grids, and if the grid is inside the estimated area, weight is added to that grid. Note that the algorithm runs on the central controller.

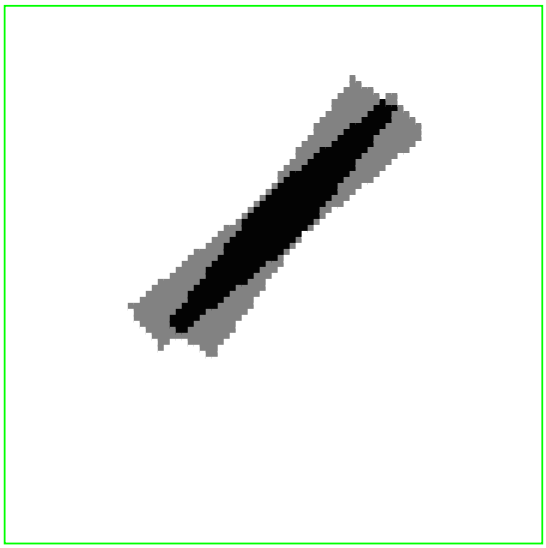
Firstly, for all grids within the convex hull of all indicator nodes (using the estimated location), a weight of w_1 is added. Secondly, for each source node s_i, identify the corresponding indicator node p_i. If there are more than one indicator nodes, use their center of gravity as p_i. For each s_i, plot a rectangle such that:



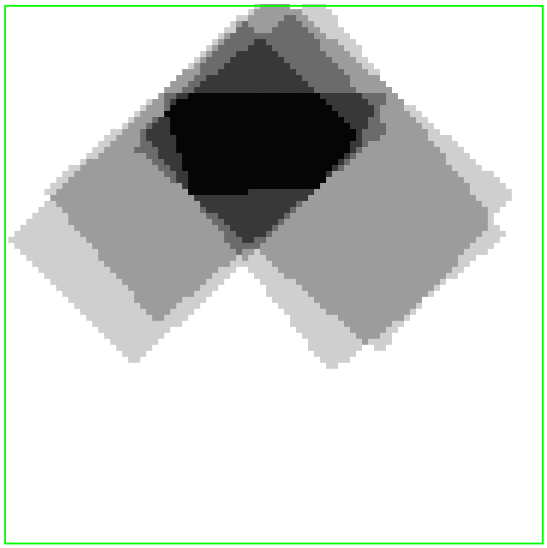
(a) Estimate of Figure 5.4(a)



(b) Estimate of Figure 5.4(b)



(c) Estimate of Figure 5.4(c)



(d) Estimate of Figure 5.4(d)

Figure 5.8: Hole estimation

1. One edge is perpendicular to $s_i p_i$ and passes through p_i.

2. One edge is perpendicular to $s_i p_i$ and passes through p_j where p_j is another indicator node on the hole boundary closest to s_i.

3. The two remaining edges are parallel and perpendicular to the two previous edges. It is also symmetric with respect to $s_i p_i$ and the distance between them is estimated using the hop count changes of the indicator node p_i.

For all grids within this rectangle, a weight of w_2 is added.

One shall note that the proposed algorithm gives more accurate result for convex holes. For concave holes, the contribution from the second factor may over estimate the hole size.

5.6.2 Evaluation

In the simulations, we assume the value of w_1 and w_2 are the same and the weights from all contributors sum to 1.

Figures 5.8(a) to 5.8(d) show the results for estimation of the holes in Figures 5.4(a) to 5.4(d) respectively. The results in Figures 5.8(a) to 5.8(c) show that for convex holes, the locations, sizes and shapes of the holes can be fairly well approximated. The result is not as good for the concave polygon shown in Figure 5.8(d) because the estimation due to the second factor may be larger than the actual hole size when the indicator nodes are at the concave edges. Nevertheless, even for arbitrary shapes, Theorem 5.2 states that the convex hull of the indicator points is contained within the convex hull of the hole.

5.6.3 Estimation Without Localization Information

Without localization information, estimation of a hole's property relating to size and using only a small number of indicator nodes is difficult. In this section, we show that the changes in geodesic distance of indicator points, i.e., the maximum changes of geodesic

distance of all the points in the network, can provide an estimate on the size of the hole formed.

However, the proposed estimation is applicable only to convex holes. To see why it is difficult to provide estimate on hole size using only indicator nodes for arbitrary shapes, consider a spiral like hole. The indicator point may lie deep inside the spiral. The geodesic distance change of the indicator point can be proportional to the total "length" of the spiral and thus be much larger than the "size" of the hole. Nevertheless, we believe that the result is still interesting as many of the "natural" holes of interest, for example, those due to fire, explosion or jamming can be approximated as convex.

We first formally define the size of a convex hole in continuous domain.

Definition 5.3 *Consider a line that joins the source point s and its corresponding indicator point p. The* ***breadth*** *of a convex hole with respect to the source point s is the length of projection of the hole onto the direction perpendicular to the line sp. Similarly, the* ***depth*** *of a convex hole is the length of projection of the hole onto the direction parallel to the line sp.*

This is illustrated in Figure 5.9(a), where b represents breadth and d represents depth.

We first present the results for the special case of convex holes that are self-symmetric with respect to the line sp (e.g., a circle-like hole is always self-symmetric with respect to sp).

We first present the results for the special case of convex holes that are self-symmetric with respect to the line sp.

Lemma 5.1 For a new convex hole o' that is self-symmetric with respect to the line connecting the source point s and its corresponding indicator point p, the change in geodesic distance of the indicator point increases monotonically when (i) the distance between source point s and the boundary of the hole decreases along the line sp; (ii) the breadth of the hole increases; and (iii) the depth of the hole decreases.

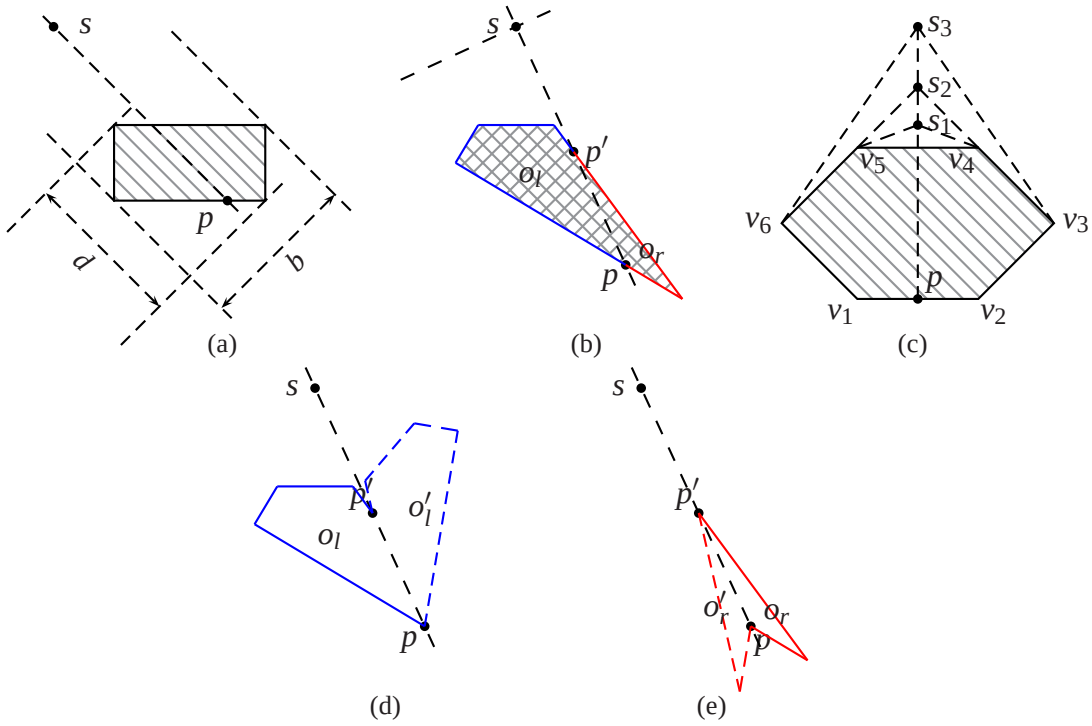


Figure 5.9: Breadth and depth

Proof: (i) Since the hole is symmetric with respect to sp, the indicator point (also the bisector point) does not change when s moves closer to the hole.

As shown in Figure 5.9(c), the geodesic distance change of point p when s is at the position of s_3 is $\Delta l_3 = |s_3v_6| + |v_6v_1| + |v_1p| - |s_3p|$. Similarly, the geodesic distance change of p when s is at s_2 and s_1 are $\Delta l_2 = |s_2v_6| + |v_6v_1| + |v_1p| - |s_2p|$ and $\Delta l_1 = |s_1v_5| + |v_5v_6| + |v_6v_1| + |v_1p| - |s_1p|$ respectively. It is easy to see that $\Delta l_1 - \Delta l_2 > 0$ and $\Delta l_2 - \Delta l_3 > 0$.

(ii) and (iii) can be proven in a similar way. □

Theorem 5.3 *For a convex hole that is self-symmetric with respect to the line sp, the change in geodesic distance of the indicator point Δl is a lower bound on the breadth of the hole with respect to s. I.e., $\Delta l \leq b$.*

Proof: This comes naturally from the Lemma 5.1. When s is on the boundary of the new hole, and the depth of the hole is close to 0, i.e., the hole is a line barrier with infinitely small interior, the change in geodesic distance of point p is largest (for the same

breath b), and $\Delta l = b$. □

Before stating the next theorem, we need another definition.

Definition 5.4 *Consider a hole, o', a source point s and an indicator point p relative to s (on the boundary of o'). We say s is **well separated** from o' if o' is completely located on one side of the line that passes through s and is perpendicular to sp.*

This is illustrated in Figure 5.9(b). s is well separated from the hole because the hole is located completely on one side of the dashed line perpendicular to sp.

Now, we can state the result for arbitrary convex hole.

Theorem 5.4 *For an arbitrary convex hole, if the source point s is well separated from the hole, $\Delta l < 2b$.*

Theorem 5.4 states that the breadth of a convex hole is lower bonded by half of the largest geodesic distance change ($b > l/2$). The intuition behind is that the longer the largest change in geodesic distance, the bigger the size of hole. The proof of the theorem is as follows.

Proof: Consider an arbitrary convex hole, o, the source point s which is *well separated* from o, and the corresponding indicator point p. The line sp separates o into two parts o_l and o_r. This is shown in Figure 5.9(b), where sp separates the hole (the cross hatched area) into o_l (enclosed by blue lines and $p'p$) and o_r (enclosed by red lines and $p'p$).

Create the mirror image of o_l and o_r with respect to line sp. Name them o'_l and o'_r respectively. Let the union of o_l and its virtual image o'_l be o_1 (Figure 5.9(d)), and let the union of o_r and its virtual image o'_r be o_2 (Figure 5.9(e)). If either o_1 or o_2 is treated as the new hole (rather than o), it has the same indicator point p and the same geodesic distance change Δl at the point p as the hole o. More importantly, both o_1 and o_2 are self-symmetric with respect to the line sp. Assume the breath of o_1 is b_1, the breath or o_2 is b_2 and the breath of the real hole o is b.

Since sp intersects the boundary of o at either an edge or a vertex at point p, it is easy to see that at least one of o_1 and o_2 have an interior angle of less than or equal to 180° at the point p. Without loss of generality, we assume o_1 (Figure 5.9(d)) has an interior angle of less than 180° at p.

If s is well separated from o, s must lie outside the convex hull of o_1. If we treat the convex hull of o_1 as the new hole, the indicator point is still at p due to the self-symmetry property, the breath of the convex hull of o_1 is the same as the breath of o_1 (b_1) and the change in geodesic distance of p is still Δl. From Theorem 5.3, $\Delta l \leq b_1$. Since $b = \frac{b_1+b_2}{2}$, we have $\Delta l \leq 2b - b_2 < 2b$. □

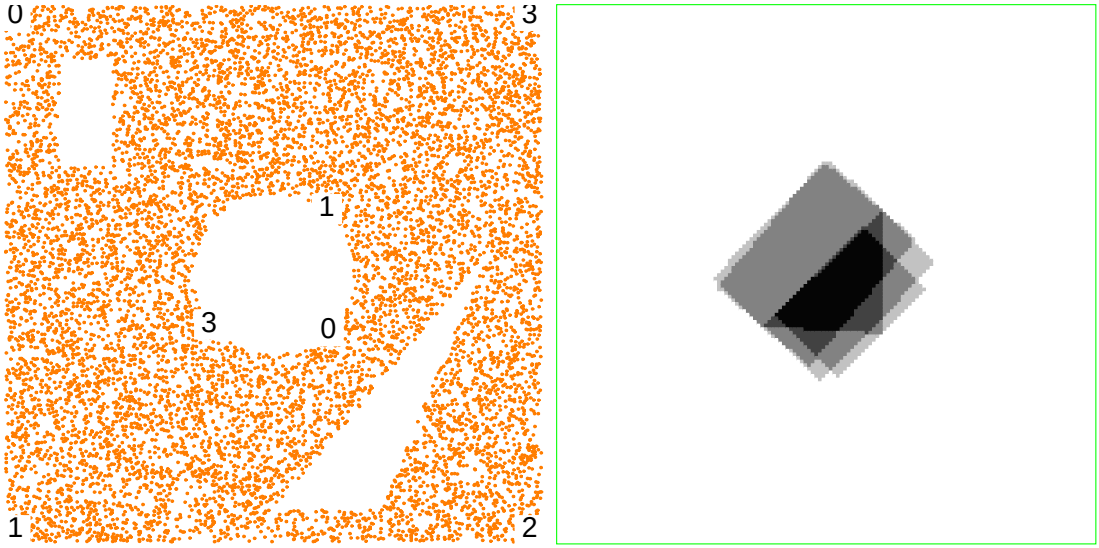
Theorems 5.3 and 5.4 show that change in geodesic distance of the indicator point provides a lower bound for the breadth of the convex hole. If the geodesic distance change of an indicator point is Δl, then one can conclude that the breadth of the hole with respect to s is at least $\frac{\Delta l}{2}$ if we assume the hole is convex. If the hole to be detected is known to be a circle (always self-symmetric with respect to the line sp), the breadth (diameter) of the hole is then at least Δl. The geodesic distance Δl can be estimated using hop counts times the average hop progress.

5.7 Discussions

Effects of existing holes

The presence of existing holes does not affect the correctness of indicator node identification. However, if the existing hole is between the new hole and the source node, the change in hop count may be reduced thus making the detection granularity coarser than expected.

Figure 5.10(a) shows a network of 12,000 nodes placed on a 50×50 square. The newly created hole is the circle shown in the middle and is blocked by existing holes to source node 0 and 2. The indicator node for source node 0 can still be correctly elected because the size of existing hole that blocks the new hole is small. The indicator node for



(a) Case with existing holes (b) Estimate with existing holes

Figure 5.10: Effect of existing holes

source 2 fails to be elected (no node in the network has hop count change larger than 5) due to the large existing hole on the bottom-right corner. Nevertheless, the hole can still be detected through three other indicator nodes.

In general, a new hole will always be detected unless existing holes disrupt hopcount change detection to all source nodes. However, in terms of hole size estimation, the size can possibly be underestimated as shown in 5.10(b).

One way to deal with this problem is to utilize more source nodes, so that the possibility of new hole being "blocked" from all the source nodes becomes smaller. The source nodes can be randomly distributed within the network. The cost is higher communication overhead.

In order to maximize the probability that the new holes can be detected and accurately estimated, one can also manually allocate the locations of source nodes based on the boundary information of existing holes, which can be detected using static hole boundary recognition protocols [100, 39]. The manual allocation of source nodes is similar to the well-known art gallery problem which has been extensively studied in literature.

Formation of more than one holes

When multiple holes are formed at the same time, properties of the indicator nodes still hold and the identification process is the same. In fact, the number of indicators detected per source node provides a quick answer to the number of the holes formed in the network.

However, the issue of accuracy arise when the new and existing holes are too close together. When holes are not sufficiently well separated, they may be considered as a single hole. This is a natural consequence of our model as it is limited by the granularity of detection. When the holes are sufficiently far apart, all previously presented results hold.

5.8 Summary

In this chapter, the problem of topological hole detection and monitoring using only connectivity information is considered. The detection of hole formation is done by observing the connectivity changes of the network. The location, size and shape of the hole can be estimated using only information from a few indicator nodes. An algorithm that identifies the hole transformation type is also proposed. These algorithms are simple to implement and efficient. The estimation accuracy is also satisfiable for the administrators to detect the significance of the hole.

Chapter 6

The Coverage and Connectivity Management System

Several important microscale and macroscale coverage and connectivity management protocols have been proposed in the previous chapters. In this chapter, we show how these individual management protocols and functions can work together to form a management system based on the unified network assumptions. This chapter only serves a design of the management architecture and is not implemented.

6.1 Basics of WSN Management

Typically speaking, network management is a service that employs a variety of tools and devices to assist the human users to monitor and maintain the network. However, the management protocols on traditional wired networks do not directly apply to WSNs. For example, monitoring and controlling each individual component are common practices in wired networks, while they are not energy-efficient nor scalable for sensor networks.

Figure 6.1 shows a simple management architecture for sensor networks. In this example, each sensor node is treated as a network device and has an agent software running on itself. The "sink" can be treated as an intermediate router. The manager is able to send

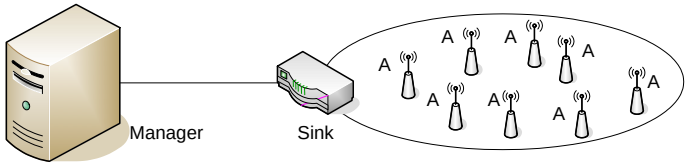


Figure 6.1: A simple management architecture for wireless sensor networks

polling messages to any sensor node, and any sensor node is able to send alert messages to manager. Depending on different applications, there could be different management architectures.

Management functions are the key components of the management system. Any service to the users will need to make use of one or several management functions to complete the task. An example list of possible management functions that need to be provided by a sensor network management architecture is listed below [87].

- Environmental monitoring function
- Topology discovery function
- Node deployment function
- Network connectivity discovery function
- Energy map generation function
- Synchronization function
- Coverage area supervision function
- Node localization discovery function

6.2 A Unified Coverage and Connectivity Management System

6.2.1 System Model

As illustrated in Chapter 1, WSN management is configuration oriented. Before the coverage and connectivity management system is proposed, the system assumptions, models, and configurations are introduced first.

This thesis focuses on the middle-size and large-size networks that consist of hundreds and thousands of sensor nodes. The management of a small network with only tens of nodes is generally less challenging. These large amount of sensor nodes are randomly deployed in the region of interest with higher than necessary density, and the network is assumed to be at least connected and covered. All the proposed management solutions in this thesis are scalable and can work with very high node density. Each node has a unique node ID. These sensor nodes cooperate among themselves in an ad hoc and distributive manner. As shown in Figure 6.1, there are one or more root nodes (sink nodes or central controllers) who act as gateways between the sensor network and the outside world.

Localization is assumed to be available to every coverage and connectivity management component. Most of the proposed protocols in this thesis assume that the connectivity-based localization scheme is utilized. A connectivity-based localization scheme can provide enough localization accuracy for most of the coverage and connectivity management functions, such as microscale connectivity discovery and macroscale hole monitoring. To maintain a connectivity-based localization, we assume that there are several anchor nodes (or source nodes) in the network. The absolute or at least the relative locations among these anchors are known. The anchor nodes can also act as the root nodes. Each sensor node locally maintains its hop counts to all of these anchor nodes so that they can be coarsely localized based on any trilateration algorithms [74, 62].

As mentioned in Chapter 3, microscale coverage management functions require more

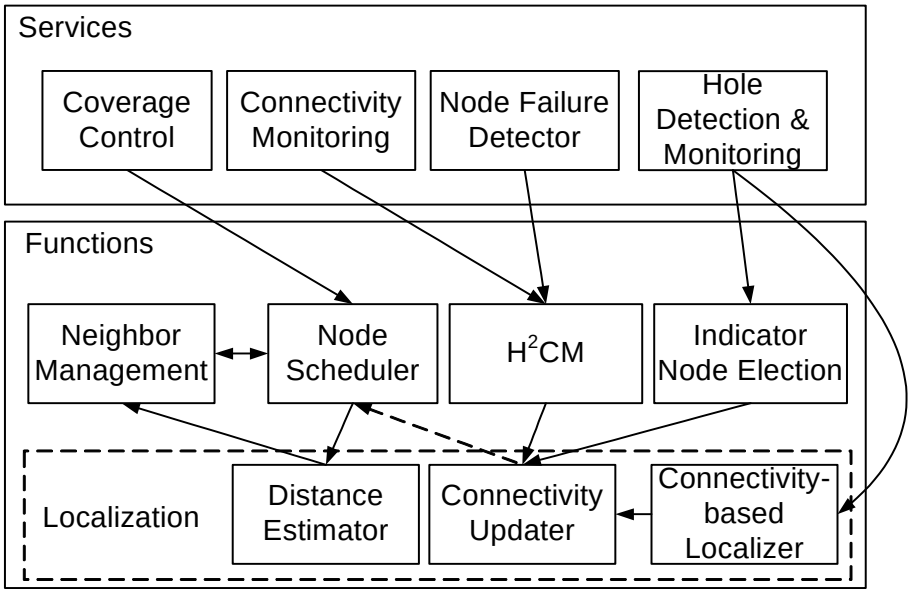


Figure 6.2: The coverage and connectivity management system.

accurate localization information. In this thesis, distance estimation is assumed to be available for microscale coverage management. The distance estimation error is assumed to be well bounded, i.e., to be within a small percentage of the sensor nodes' sensing range. It should be noted that distance (with errors) is a weaker requirement than localization since it only reflects the relative locations among the neighboring sensor nodes. Distance estimation cannot be converted to global level localization easily due to the possible error aggregation.

The network is assumed to maintain a tree-based information collection model, where various information, once distributively processed, is sent to the root nodes via the information collection trees. Communication reliability is also assumed in this thesis, unless explicitly stated otherwise. The reliability can be supported by link layer retransmissions for unicast packet losses.

6.2.2 The Coverage and Connectivity Management System

Several efficient coverage and connectivity management protocols have been proposed in the previous chapters, including microscale coverage and connectivity monitoring and

controlling, as well as macroscale topological hole detection and monitoring. In this section, we show how these individual management protocols cooperatively form a unified coverage and connectivity management system, based on the system model illustrated in the previous section. The aim of the management system is to provide the users and administrators a range of services and tools to achieve the coverage and connectivity management goals, from both microscale and macroscale levels.

The proposed management system is shown in Figure 6.2. The system is constructed using various coverage and connectivity management functions, which in turn support different management services to the users or administrators in the upper layer. In other words, the management services shown in the figure represent the services provided by the central controller to the users or administrators; and the management functions run on each individual sensor node to support the management services. The solid arrows in the figure represent the relation of *"supports"*, i.e., if an arrow is drawn from X to Y, it indicates that the function or service X is supported by the function or service Y. A dashed arrow from X to Y indicates that X is passively affected by Y. These individual management components are explained in details as follows.

Localization Management Functions

Localization functions, although do not fit in the category of network coverage and connectivity management, are the fundamental support for various coverage and connectivity management tasks. They are therefore included in the proposed management system.

- **Connectivity Updater:** The connectivity updater updates a node's hop counts to the anchor nodes by listening for the periodic "HELLO" messages from its neighbors. This hop count information is maintained as a hop count vector. Note that the connectivity updater is passively affected by the node scheduler because different sets of active nodes results in different network topology and thus influences the hop counts to the anchors.

- **Connectivity-based Localizer:** The connectivity-based localizer estimates a node's location utilizing the hop count vector provided by the connectivity updater [74, 62].

- **Distance Estimator:** The distance estimator component proposed in Section 3.7 is utilized to support node scheduling for microscale coverage control. Since the proposed solution is based on the number of common neighbors among the directly commutable pairs, it has to be aware of the neighbor management protocol.

Coverage and Connectivity Management Functions

The coverage and connectivity management functions form the basis for all the coverage and connectivity management services in the upper layer.

- **Node Scheduler:** The node scheduler schedules the active or inactive states of sensor nodes based on the required network coverage and connectivity (the parameter α). The user or network administrator can adjust the parameter α. This is supported by a control message flooding component which is not shown in the figure.

- **H^2CM:** The H2CM component compresses a node's neighbor table for the purpose of connectivity monitoring (and node failure detection). It is supported by the hop vectors provided by the connectivity updater component. It should be noted that H^2CM only requires the hop vectors collected during the initialization phase. Any subsequent hop vector changes needs not to be known by H^2CM. This is different from the indicator node election component who requires the hop vectors to be updated periodically.

- **Indicator Node Election:** The indicator node election component monitors the network connectivity change based on the updated hop count information. It maintains a history of connectivity update and starts the indicator election process once the connectivity changes beyond some threshold.

- **Neighbor Management:** The neighbor management component is responsible for maintaining each sensor node's neighbor table. Different strategies in managing the neighbor tables affect the network connectivity, and consequently affect the node scheduler and distance estimator. In this thesis, each node is assumed to maintain all the active neighbors in its neighbor table regardless of the network node density, which simplifies the design of the coverage and connectivity management functions affected by the neighbor management component.

Management Services

The coverage control services utilizes the node scheduler to save the network energy. The user or administrator controls the network coverage by setting the α parameter. The connectivity monitoring service monitors the complete (or partial) network connectivity information, using the H^2CM component. As stated in Chapter 4, the node failure detector service is only active when the connectivity monitoring service is inactive. The monitoring of large holes in macroscale level is provided by the hole detection and monitoring service. It utilizes the indicator node election component and the connectivity-based localizer to identify and estimate the location and size of the hole.

6.3 Management System Operation

6.3.1 System Initialization

This section explains the system initialization process. The neighbor management component and the connectivity updater are first initialized. Both of them can be initialized by "HELLO" message broadcasting. Once the hop count vectors to the anchor nodes (before the node scheduler is run) are available, they are collected via the information collection tree to the root nodes. This information is used by the H^2CM component later.

Some nodes will then turn into inactive mode according to the decision of the node

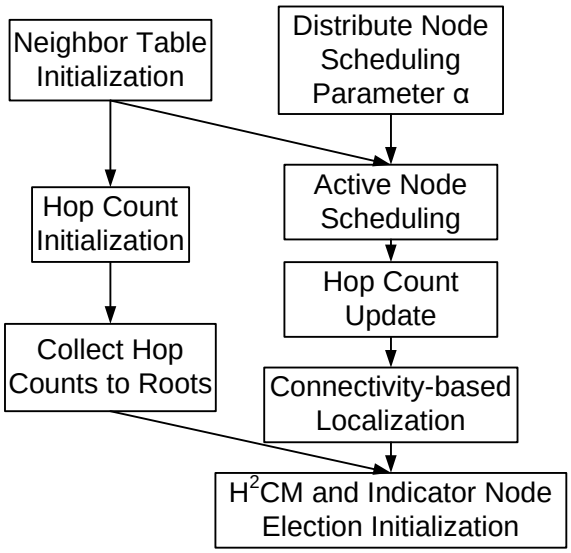


Figure 6.3: The flow diagram of the system initialization process.

scheduler. They will stay inactive for a long period so that other components are not frequently affected, until the node scheduler is re-run over a relatively long period or disabled. The connectivity updater and the connectivity-based localizer will then start running, right after the node scheduling process has finished. The H^2CM component and the indicator node election component are the last two management functions to start operating.

The flow diagram of the above described initialization process is illustrated in Figure 6.3.

6.3.2 Normal System Operation

Once the system is initialized, the individual functions and services can operate on their own to a large extent, which has already been explained in detail in the previous chapters. In this section, the flow diagram of these components are summarized, which is shown in Figure 6.4.

The active node scheduling process is started by either the node scheduling timer or the re-scheduling request from users or administrators. The node scheduling protocol

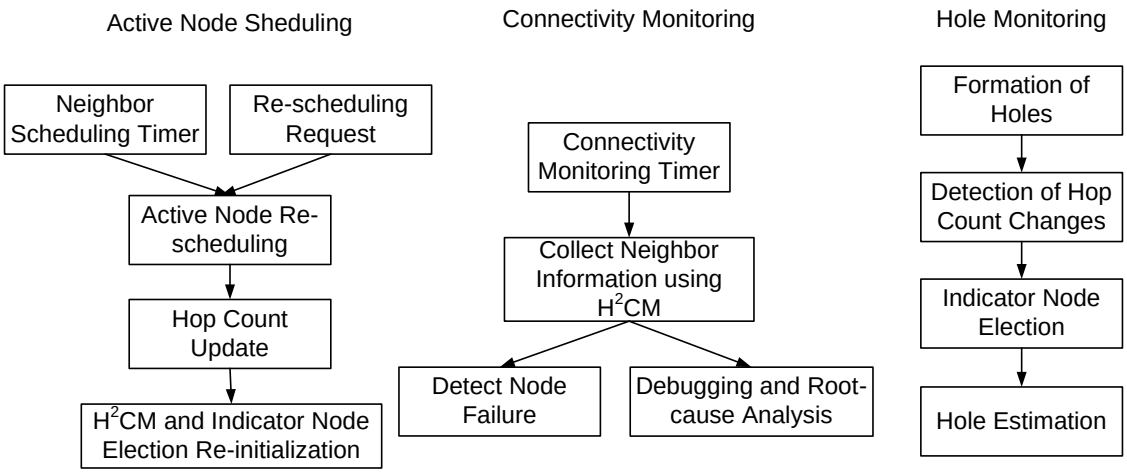


Figure 6.4: Illustration of normal system operation.

for coverage and connectivity control will then be performed. After the active node re-scheduling, those components that rely on the node scheduler component, such as the H^2CM and indicator node election components, are re-initialized.

The connectivity monitoring process is started by the connectivity monitoring timer. The neighbor tables of the selected active sensor nodes are collected to the root nodes using H^2CM protocol. Connectivity-based debugging and root-cause analysis or the node failure detection will then be performed.

The hole monitoring process is started by the event that a large hole forms in the network and the network topology (reflected by the hop counts) has significantly changed. The indicator node election process will then be performed. Upon receiving the information from the elected indicator nodes, the root nodes will estimate the location and size of the hole.

Chapter 7

Conclusion and Future Work

The modern research on wireless sensor networks started around 1980 [23] driven by the military applications, however, the technology for small sensors was not quite ready at that time. During the last decade, due to the rapid development of various enabling technologies for sensors, research on sensor networks has regained significant attention. Besides the original military applications, wireless sensor networks are now used in many industrial and civilian applications, including industrial process monitoring and control, environment and habitat monitoring, healthcare applications, home automation, traffic control, and etc. The idea of wireless sensors is in fact so exciting that the small sensors are expected to be everywhere in the future world [84].

However, research in wireless sensor networks encounters many challenges due to their unique characteristics: large-scale deployment, distributed protocol design, limited resources, harsh environments, and many more. Wireless sensor networks are also configuration oriented. Different application requirements, different types of sensor nodes, and different system model and assumptions may result in completely different problem formulation and protocol design.

This thesis focuses on the management aspect of a sensor network, particularly, the coverage and connectivity management. Several protocols on monitoring and controlling the network coverage and connectivity, both in microscale and macroscale levels, were

proposed. The detailed research results and contributions of this thesis are summarized in the next section.

7.1 Research Summary

As stated in Chapter 1, the coverage and connectivity management functions are categorized into monitoring and controlling of the network coverage and connectivity, in both microscale level and macroscale level. In this thesis, these components were studied in separate chapters. The integration of these components into a unified coverage and connectivity management framework was then proposed in Chapter 6.

The Configurable Coverage Protocol (CCP) proposed in Chapter 3 serves the purpose of micrascale coverage and connectivity control. Meanwhile, the vacancy estimation scheme proposed in CCP also provides a way to compute the microscale vacancy of the given network, and thus also serves as a management function for microscale coverage monitoring. CCP allows the trade-off between coverage and node usage (i.e., the number of active nodes). It can be configured to use a small number of active nodes to cover at least α portion of the area with high probability. CCP only makes uses of the distance between two nodes rather than their actual locations.

CCP is a completely distributed and lightweight protocol where each node makes decision based on the collaboration between its local neighbors. For complete coverage ($\alpha = 1$), CCP was comparable to the near optimal OGDC protocol [107] in terms of coverage and number of active nodes required. By relaxing the constraints of complete coverage, CCP was able to generate a subset of sensor nodes which was smaller than the number of nodes required for a complete coverage, e.g.,when $\alpha = 90\%$, 22% node savings could be achieved comparing to the case of full coverage, and when $\alpha = 80\%$, 29% savings could be achieved. E.g., for the node density of 10, about 400 active nodes can support 90% coverage while about 530 active nodes are required to support full coverage. The reduction in the number of active nodes is much more than the reduction of coverage.

The complete network connectivity graph is formed by aggregating the microscale connectivity information (neighbor tables) of all the sensor nodes in the region of interest. An efficient microscale connectivity monitoring protocol H^2CM was proposed in Chapter 4. H^2CM is an efficient way to encode the neighborhood information of each sensor nodes, such that the communication cost of microscale connectivity collection can be much reduced. By varying the amount of information exchanged, H^2CM is able to provide different level of connectivity information accuracy. The H^2CM algorithm is practical and can be easily implemented on TinyOS with little overhead.

Simulation results showed that for a large network (> 1000 nodes) with node densities varying from 5 to 30, over 99.99% of all links were discovered and the communication savings varied from 65% to 85% compare to maximal compression of neighborhood information. For a medium size network (a few hundred nodes), about 40% to 70% savings could be achieved. We implemented H^2CM in a sensor testbed with 34 MICA2 nodes. The algorithm was implemented using less than 80 lines of TinyOS code and about 600 bytes of ROM image size (code size). Even with such a small network, the total communication cost was comparable to the cost of using maximal compression.

Node failure detection is also a simple application of H^2CM. By combining H^2CM with the concept of dominating set, the communication cost can be drastically reduced compare to traditional data collection method. The average communication cost was only 20% to 40% of the normal data collection method. This is a significant improvement since H^2CM achieves much better performance even compare to the theoretical maximal data compression in information theory.

Chapter 5 presented an efficient macroscale topological hole detection and monitoring protocol. The protocol is based on the observation that hole formation creates irregularities in the network connectivity and the changes in the network connectivity contains important information about the hole. The approach is reactive and communication is triggered only when a hole is formed, unlike a polling/sampling based method where communication needs to be performed periodically. In addition, the aim of the protocol

is not to map the boundary of the hole, which is expensive since many nodes need to be identified. Instead, only a small number of dynamically identified *indicator nodes* are required to report their status to the sink nodes. The properties of these indicator nodes are investigated and utilized to estimate the location and size of the hole, as well as the possible hole transformation types.

Simulation results showed that the location and size of the holes could be fairly accurately estimated with only the information from indicator nodes. For a large network (more than $30,000$ sensor nodes), the communication overhead for the hole diameter of 20 was only about 0.3 message per node per hole detected, which was small compare to existing methods.

All these proposed solutions to the coverage and connectivity management components described in Chapter 3, 4 and 5 are distributed algorithms. They are efficient in communication and energy cost and scalable to a very large and dense wireless sensor network. A unified coverage and connectivity management framework was proposed in Chapter 6. The framework maps all these described individual components into the management functions and services. The dependencies among these functions and services were carefully investigated.

7.2 Future Work

There are several possible extensions to the research work presented in this thesis. Although the management framework proposed in Chapter 6 includes many microscale and macroscale coverage and connectivity management functions and services, it can hardly be considered as a complete framework.

- Although the CCP protocol proposed in Chapter 3 controls the microscale coverage and connectivity, there are apparently many other formulations of the problem coverage and connectivity control. The most obvious extension is to extend CCP to support k-coverage and k-connectivity.

- The CCP protocol only focuses on the microscale coverage and connectivity control. An extension to this work is to use the proposed vacancy estimation method for macroscale coverage monitoring. Macroscale coverage monitoring is an important management service that is not included in the management framework proposed in this thesis.

- The macroscale hole monitoring is investigated in Chapter 5. The problem of mitigating macroscale holes, such as the node deployment schemes to avoid the formation of large holes, as well as the node redeployment schemes to eliminate the existing holes are not studied in this thesis. These components can be investigated as future work to make the proposed management framework more complete.

- The research work in this thesis heavily relies on connectivity based localization, which in turn relies on the assumption of Poisson random placement of sensor nodes. This assumption does not cause trouble for the protocols proposed for microscale coverage and connectivity monitoring. However, it is an important assumption for macroscale connectivity and coverage monitoring, especially for hole monitoring and estimation. The impact of other distributions of node placement to the macroscale connectivity and coverage monitoring can be investigated as future work.

At last, although the individual management functions and services have been extensively simulated, and some of them have been implemented and tested on real sensor network testbed, the simulation and testbed implementation of the proposed framework has not been evaluated and can be considered as future work.

Bibliography

[1] African Internet report. http://comet.columbia.edu/~nemo/netmap.

[2] ARGO - Global Ocean Sensor Network. http://www.argo.ucsd.edu/.

[3] Internet mapping project. http://www.cheswick.com/ches/map/index.html.

[4] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. Link-level measurements from an 802.11b mesh network. In *the ACM Annual Conference of the Special Interest Group on Data Communication (SIGCOMM)*, 2004.

[5] Nadeem Ahmed, Salil S. Kanhere, and Sanjay Jha. The holes problem in wireless sensor networks: A survey. *Mobile Computing and Communications Review*, 9(2):4–18, April 2005.

[6] Anish Arora, Emre Ertin, Rajiv Ramnath, Mikhail Nesterenko, and William Leal. Kansei: A high-fidelity sensing testbed. *IEEE Internet Computing, special issue on Large-Scale Sensor Networks*, pages 35–47, March 2006.

[7] Jerry Banks, Manuel A. Pachano, Les G. Thompson, and David Hanny. *RFID Applied*. John Wiley & Sons, 2007.

[8] Amitabh Basu, Jie Gao, Joseph S. B. Mitchell, and Girishkumar Sabhnani. Distributed localization unsing noisy distance and angle information. In *the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2006.

[9] Pratik Biswas and Yinyu Ye. Semidefinite programming for ad hoc wireless sensor network localization. In *Information Processing in Sensor Networks (IPSN)*, 2004.

[10] Burton H. Bloom. Space/time tradeoffs in hash coding with allowable errors. *Communications of the ACM*, 1970.

[11] Andrei Broder and Michael Mitzenmacher. Network applications of Bloom filters: A survey. *Internet Mathematics*, 2004.

[12] Nirupama Bulusu, John Heidemann, and Deborah Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communication Magazine*, (5):28, 2000.

[13] Chiranjeeb Buragohain and Sorabh Gandhi. Contour approximation in sensor networks. In *the International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2006.

[14] John Byers, Jeffrey Considine, Michael Mitzenmacher, and Stanislav Rost. Informed content delivery across adaptive overlay networks. In *the ACM Annual Conference of the Special Interest Group on Data Communication (SIGCOMM)*, 2002.

[15] Gruia Calinescu and Peng jun Wan. Range assignment for high connectivity in wireless ad hoc networks. In *the International Conference on Ad hoc and Wireless Networks (AdHoc-NOW)*, 2003.

[16] Mihaela Cardei and Jie Wu. Coverage in wireless sensor networks. In Mohammad Ilyas and Imad Mahgoub, editors, *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, chapter 19. CRC, 2004.

[17] Larry Carter, Robert Floyd, John Gill, George Markowsky, and Mark Wegman. Exact and approximate membership testers. In *the Annual ACM Symposium on Theory of Computing (STOC)*, 1978.

[18] Alberto Cerpa, Naim Busek, and Deborah Estrin. SCALE: A tool for simple connectivity assessment in lossy environments. Technical Report 21, CENS UCLA, 2003.

[19] Ranveer Chandra, Christof Fetzer, and Karin Högstedt. A mesh-based robust topology discovery algorithm for hybrid wireless networks. Technical report.

[20] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *ACM Wireless Networks Journal*, pages 85–96, 2001.

[21] Yuan-Chieh Cheng and Thomas G. Robertazzi. Critical connectivity phenomenon in mulithop radio models. *IEEE Transactions on Communications*, 1989.

[22] Krishna Kant Chintalapudi and Ramesh Govindan. Localized edge detection in sensor fields. In *the IEEE International Workshop on Sensor Network Protocols and Applications (SNPA)*, 2003.

[23] Chee Yee Chong and Srikanta P. Kumar. Sensor networks: evolution, opportunities, and challenges. In *Proceedings of the IEEE*, pages 1247–1256, 2003.

[24] Chee-Yee Chong and Srikanta P. Kumar. Sensor networks: Evolution, opportunities, and challenges. In *Proceedings of the IEEE*, pages 1247–1256, 2003.

[25] Peter Corke, Ron Peterson, and Daniela Rus. Finding holes in sensor networks. In *the IEEE Workshop on Omniscient Space: Robot Control*, 2007.

[26] Mathieu Couture, Michel Barbeau, Prosenjit Bose, and Evangelos Kranakis. Incremental construction of k-dominating sets in wireless sensor networks. *Ad Hoc & Sensor Wireless Networks*, 5(1-2):281–293, 2008.

[27] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, 1997.

[28] Budhaditya Deb, Sudeept Bhatnagar, and Badri Nath. Multiple-resolution state retrieval in sensor networks. In *the IEEE Workshop on Sensor Network Protocols And Applications (SNPA)*.

[29] Budhaditya Deb, Sudeept Bhatnagar, and Badri Nath. A topology discovery algorithm for sensor networks with applications to network management. In *the IEEE CAS Workshop on Wireless Communications and Networking*, 2002.

[30] Budhaditya Deb, Sudeept Bhatnagar, and Badri Nath. STREAM: Sensor topology retrieval at multiple resolutions. *Journal of Telecommunications, Special Issue on Wireless Sensor Networks*, June 2004.

[31] Lance Doherty, Kristofer S. J. Pister, and Laurent El Ghaoui. Convex position estimation in wireless sensor networks. In *the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, April 2001.

[32] Benoit Donnet, Philippe Raoult, Timur Friedman, and Mark Crovella. Efficient algorithms for large-scale topology discovery. In *the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2005.

[33] T. Eren, D. Goldenberg, W. Whiteley, Y. R. Yang, A. S. Morse, B. D. O. Anderson, and P. N. Belhumeur. Rigidity, computation, and randomization in network localization. In *the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2004.

[34] Deborah Estrin, Lewis Girod, Greg Pottie, and Mani Srivastava. Instrumenting the world with wireless sensor networks. In *the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2033–2036, 2001.

[35] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 1999.

[36] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. Summary cache: A scalable wide-area web cache sharing protocol. *IEEE/ACM Transactions on Networking*, pages 281–293, 2000.

[37] Qing Fang, Jie Gao, and Leonidas J. Guibas. Locating and bypassing routing holes in sensor networks. In *the Conference of the IEEE Communications Society (INFOCOM)*, 2004.

[38] Sándor P. Fekete, Michael Kaufmann, Alexander Kröller, and Katharina Lehmann. A new approach for boundary recognition in geometric sensor networks. In *the Canadian Conference on Computational Geometry (CCCG)*, 2005.

[39] Sándor P. Fekete, Alexander Kröller, Dennis Pfisterer, Stefan Fischer, and Carsten Buschmann. Neighborhood-based topology recognition in sensor networks. In *the Workshop on Algorithmic Aspects of Sensor Networks (ALGOSENSORS)*, 2004.

[40] Stefan Funke. Topological hole detection in wireless sensor networks and its applications. In *the ACM/SIGMOBILE International Workshop on Foundation of Mobile Computing (DIAL-M-POMC)*, 2005.

[41] Stefan Funke and Christian Klein. Hole detection or: 'how much geometry hides in connectivity?'. In *the Annual ACM Symposium on Computational Geometry (SCG)*, 2006.

[42] Sorabh Gandhi, John Hershberger, and Subhash Suri. Approximate isocontours and spatial summaries for sensor networks. In *the International Conference on Information Processing in Sensor Networks (IPSN)*, 2007.

[43] Sorabh Gandhi, Subhash Suri, and Emo Welzl. Catching elephants with mice: Sparse sampling for monitoring sensor networks. In *the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2007.

[44] Yong Gao, Kui Wu, and Fulu Li. Analysis on the redundancy of wireless sensor networks. In *the ACM International Workshop on Wireless Sensor Networks & Applications (WSNA)*, September 2003.

[45] Sinan Gezici, Zhi Tian, Georgios B. Biannakis, Hisashi Kobayashi, Andreas F. Molisch, H. Vincent Poor, and Zafer Sahinoglu. Localization via ultra-wideband radios. *IEEE Signal Processing Magazine*, 22(4):70–84, 2005.

[46] Amitabha Ghosh and Sajal K. Das. Coverage and connectivity issues in wireless sensor networks. In Rajeev Shorey, A. Ananda, Mun Choon Chan, and Wei Tsang Ooi, editors, *Mobile, Wireless, and Sensor Networks - Technology, Applications and Future Directions*, chapter 9. Wiley-Interscience, 2006.

[47] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, and Tarek Abdelzaher. Range-free localization schemes for large scale sensor networks. In *the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2003.

[48] Joseph M. Hellerstein, Wei Hong, Samuel Madden, and Kyle Stanek. Beyond average: Toward sophisticated sensing with queries. In *the International Conference on Information Processing in Sensor Networks (IPSN)*, 2003.

[49] Nojeong Heo and Pramod K. Varshney. A distributed self-spreading algorithm for mobile wireless sensor networks. In *the IEEE Wireless Communications and Networking Conference*, 2003.

[50] Bernhard Hofmann-Wellenhof, Herbert Lichtenegger, and James Collins. *Global Positioning System: Theory and Practice*. Springer Verlag, 1997.

[51] Andrew Howard, Maja J Matarić, and Gaurav S Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *the International Symposium on Distributed Autonomous Robotics Systems (DARS)*, 2002.

[52] Chi-Fu Huang and Yu-Chee Tseng. The coverage problem in a wireless sensor network. In *the ACM International Workshop on Wireless Sensor Networks & Applications (WSNA)*, September 2003.

[53] Jennifer C. Hou Hyuk Lim. Localization for anisotropic sensor networks. In *the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2005.

[54] Rajagopal Iyengar, Koushik Kar, and Suman Banerjee. Low-coordination topologies for redundancy in sensor networks. In *the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, May 2005.

[55] Fernán Izquierdo, Marc Ciurana, Francisco Barceló, Josep Paradells, and Enrica Zola. Performance evaluation of a TOA-based trilateration method to locate terminals in WLAN. In *International Symposium on Wireless Pervasive Computing (ISWPC)*, 2006.

[56] Xiang Ji and Hongyuan Zha. Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling. In *the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2004.

[57] Xiaohua Jia, Dongsoo Kim, Sam Makki, Peng-Jun Wan, and Chih-Wei Yi. Power assignment for k-connectivity in wireless ad hoc networks. *Journal of Combinatorial Optimization*, 9(2):213, 2005.

[58] Jie Jiang and Wenhua Dou. A coverage-preserving density control algorithm for large wireless sensor networks. In *the International Conference on Ad Hoc and Wireless Networks (AdHoc-NOW)*, 2004.

[59] Alexander Kröller, Sándor P. Fekete, Dennis Pfisterer, and Stefan Fischer. Deterministic boundary recognition and topology extraction for large sensor networks. In *the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.

[60] Sven O. Krumke, Rui Liu, Errol L. Lloyd, Madhav V. Marathe, Ram Ramanathan, and S.S. Ravi. Topology control problems under symmetric and asymmetric power thresholds. In *the International Conference on Ad hoc and Wireless Networks (AdHoc-NOW)*, 2003.

[61] Abishek Kumar, Jim Xu, and Li Li. Space-code Bloom filter for efficient traffic flow measurement. In *ACM SIGCOMM Conference on Internet Measurement*, 2003.

[62] Ben Leong, Barbara Liskov, and Robert Morris. Greedy virtual coordinates for geographic routing. In *the IEEE International Conference on Network Protocol (ICNP)*, 2007.

[63] Mo Li and Yunhao Liu. Rendered path: Range-free localization in anisotropic sensor networks with holes. In *the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2007.

[64] Pei-Kai Liao, Min-Kuan Chang, and C.-C. Jay Kuo. Contour line extraction with wireless sensor networks. In *the IEEE International Conference on Communications (ICC)*, 2005.

[65] Hyuk Lim, Lu-Chuan Kung, Jennifer C. Hou, , and Haiyun Luo. Zero-configuration, robust indoor localization: Theory and experimentation. In *the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2006.

[66] Errol L. Lloyd, Ram Ramanathan, Rui Liu, S. S. Ravi, and Madhav V. Marathe. Algorithmic aspects of topology control problems for ad hoc networks. *Mobile Networks and Applications*, 10:19, 2005.

[67] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, and John Anderson. Wireless sensor networks for habitat monitoring. In *the ACM Interna-*

tional Workshop on Wireless Sensor Networks and Applications (WSNA), Atlanta GA USA, September 2002.

[68] Seapahn Meguerdichian, Farinaz Koushanfar, Miodrag Potkonjak, G. Qu, and Mani B. Srivastava. Exposure in wireless ad-hoc sensor networks. In *the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, July 2001.

[69] Seapahn Meguerdichian, Farinaz Koushanfar, Miodrag Potkonjak, and Mani B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, July 2001.

[70] Seapahn Meguerdichian, Sasa Slijepcevic, Vahag Karayan, and Miodrag Potkonjak. Localized algorithms in wireless ad-hoc networks: location discovery and sensor exposure. In *the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2001.

[71] Florian Michahelles, Peter Matter, Albrecht Schmidt, and Bernt Schiele. Applying wearable sensors to avalanche rescue. *Computers and Graphics*, 27(6):839–847, 2003.

[72] Joseph S.B. Mitchell. A new algorithm for shortest paths among obstacles in the plane. *Annals of Mathematics and Artificial Intelligence*, 3(1):83–105, 1991.

[73] David Moore, John Leonard, Daniela Rus, and Seth Teller. Robusted distributed network localization with noisy range measurements. In *the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[74] Dragos Niculescu and Badri Nath. Ad hoc positioning system (APS) using AOA. In *the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2003.

[75] Dragos Niculescu and Badri Nath. DV basd positioning in ad hoc networks. *Journal of Telecommunication Systems*, 2003.

[76] Robert Nowak and Urbashi Mitra. Boundary estimation in sensor networks: Theory and methods. In *the International Conference on Information Processing in Sensor Networks (IPSN)*, 2003.

[77] ElMoustapha Ould-Ahmed-Vall, Douglas M. Blough, Bonnie S. Heck, and George F. Riley. Distributed unique global ID assignment for sensor networks. In *the IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, November 2005.

[78] Anna Pagh, Rasmus Pagh, and S. Srinivasa Rao. An optimal Bloom filter replacement. In *the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2005.

[79] Sameera Poduri and Gaurav S. Sukhatme. Constrained coverage for mobile sensor networks. In *the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[80] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *the International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[81] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

[82] Nithya Ramanathan, Kevin Chang, Rahul Kapur, Lewis Girod, Eddie Kohler, and Deborah Estrin. Sympathy for the sensor network debugger. In *the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[83] Ram Ramanathan and Regina Rosales-hain. Topology control of multihop wireless networks using tranmit power adjustment. In *the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2000.

[84] Technology Review. 10 emerging technologies that will change the world. *MIT Enterprise Technology Review*, February 2003.

[85] Warren Robak. Futureworld: Sensors for Soil, Air, Everywhere. http://www.universityofcalifornia.edu/news/article/9729.

[86] Stanislav Rost and Hari Balakrishnan. Memento: A Health Monitoring System for Wireless Sensor Networks. In *the Annual IEEE Communications Society Conference on Sensor, Mesh and Ad hoc Communications and Networks (SECON)*, September 2006.

[87] Alex Rousskov and Duane Wessels. Cache digests. *Computer Networks and ISDN Systems*, pages 2155–2168, 1998.

[88] Linnyer Beatrys Ruiz, José Marcos Nogueira, and Antonio A. F. Loureiro. MANNA: A management architecture for wireless sensor networks. *IEEE Communication Magazine*, pages 116–125, Feburary 2003.

[89] Paolo Santi. *Topology Control in Wireless Ad hoc and Sensor Networks*. John Wiley & Sons, 2005.

[90] Yi Shang and Wheeler Ruml. Improved MDS-based localization. In *the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2004.

[91] Yi Shang, Wheeler Ruml, Ying Zhang, and Markus P. J. Fromherz. Localization from mere connectivity. In *the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003.

[92] Nisheeth Shrivastava, Subhash Suri, and Csaba D. Tóth. Detecting cuts in sensor networks. In *the International Conference on Information Processing in Sensor Networks (IPSN)*, 2005.

[93] Mitali Singh, Amol Bakshi, and Viktor K. Prasanna. Constructing topographic maps in networked sensor systems. In *the AWorkshop on Algorithms for Wireless and Ad-hoc Networks (ASWAN)*, 2004.

[94] Ignacio Solis and katia Obraczka. Efficient continuous mapping in sensor networks using isolines. In *the Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous)*, 2005.

[95] Di Tian and Nicolas D. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *the First ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.

[96] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *the International Conference on Embedded Networked Sensor Systems (SenSys)*, November 2003.

[97] Giacomino Veltri, Qingfeng Huang, Gang Qu, and Miodrag Potkonjak. Minimal and maximal exposure path algorithms for wireless embedded sensor networks. In *the International Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[98] Guiling Wang, Guohong Cao, , and Tom La Porta. A bidding protocol for deploying mobile sensors. In *the IEEE International conference on Network Protocol (ICNP)*, 2003.

[99] Guiling Wang, Guohong Cao, , and Tom La Porta. Movement-assisted sensor deployment. In *the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2004.

[100] Xiaorui Wang, Guoliang Xing, Yuanfang Zhang, Chenyang Lu, Robert Pless, and Christopher Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *the International Conference on Embedded Networked Sensor Systems (SenSys)*, November 2003.

[101] Yue Wang, Jie Gao, and Joseph S.B. Mitchell. Boundary recognition in sensor networks by topological methods. In *the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2006.

[102] Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed energy conservation for ad hoc routing. In *the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, July 2001.

[103] Wenwei Xue, Qiong Luo, Lei Chen, and Yunhao Liu. Contour map matching for event detection in sensor networks. In *the ACM International Conference on Management of Data (SIGMOD)*, 2006.

[104] Ting Yan, Tian He, and John A. Stankovic. Differentiated surveillance for sensor networks. In *the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 2003.

[105] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient MAC protocol for wireless sensor network. In *the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2002.

[106] Wei Ye, John Heidemann, and Deborah Estrin. Media access control with coordinated, adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking (ToN)*, 2004.

[107] Beichuan Zhang, Raymond Liu, Daniel Massey, and Lixia Zhang. Collecting the Internet AS-level topology. *ACM SIGCOMM Computer Communication Review (CCR), special issue on Internet Vital Statistics*, 2005.

[108] Honghai Zhang and Jennifer C. Hou. Maintaining sensing coverage and connectivity in large sensor networks. *International Journal of Wireless Ad Hoc and Sensor Networks*, 1(1-2):89–124, 2005.

[109] Mingze Zhang, Mun Choon Chan, and A. L. Ananda. Location-Aided Topology Discovery for Wireless Sensor Networks. In *the Annual IEEE Communications Society Conference on Sensor, Mesh and Ad hoc Communications and Networks (SECON)*, 2007.

[110] Yi Zou and Krishnendu Chakrabarty. Sensor deployment and target localization based on virtual forces. In *the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2003.

[111] Yi Zou and Krishnendu Chakrabarty. Sensor deployment and target localization based in distributed sensor networks. *IEEE Transactions on Embedded Computer Systems*, 3(1):61, 2004.