DISSERTATION

ANCHOR CENTRIC VIRTUAL COORDINATE SYSTEMS IN WIRELESS SENSOR NETWORKS: FROM SELF-ORGANIZATION TO NETWORK AWARENESS

Submitted by

Dulanjalie C. Dhanapala

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2012

ABSTRACT

ANCHOR CENTRIC VIRTUAL COORDINATE SYSTEMS IN WIRELESS SENSOR NETWORKS: FROM SELF-ORGANIZATION TO NETWORK AWARENESS

Future Wireless Sensor Networks (WSNs) will be collections of thousands to millions of sensor nodes, automated to self-organize, adapt, and collaborate to facilitate distributed monitoring and actuation. They may even be deployed over harsh geographical terrains and 3D structures. Low-cost sensor nodes that facilitate such massive scale networks have stringent resource constraints (e.g., in memory and energy) and limited capabilities (e.g., in communication range and computational power). Economic constraints exclude the use of expensive hardware such as Global Positioning Systems (GPSs) for network organization and structuring in many WSN applications. Alternatives that depend on signal strength measurements are highly sensitive to noise and fading, and thus often are not pragmatic for network organization. Robust, scalable, and efficient algorithms for network organization and reliable information exchange that overcome the above limitations without degrading the network's lifespan are vital for facilitating future large-scale WSN networks.

This research develops fundamental algorithms and techniques targeting self-organization, data dissemination, and discovery of physical properties such as boundaries of large-scale WSNs without the need for costly physical position information. Our approach is based on Anchor Centric Virtual Coordinate Systems, commonly called Virtual Coordinate Systems (VCSs), in which each node is characterized by a coordinate vector of shortest path hop distances to a set of anchor nodes. We develop and evaluate algorithms and techniques for the following tasks associated with use of VCSs in WSNs: (a) novelty analysis of each anchor coordinate and compressed representation of VCSs; (b) regaining lost directionality and identifying a 'good' set of anchors; (c) generating topology preserving maps (TPMs); (d) efficient and reliable data dissemination, and boundary identification without physical information; and (f) achieving network awareness at individual nodes.

After investigating properties and issues related to VCS, a Directional VCS (DVCS) is proposed based on a novel transformation that restores the lost directionality information in VCS. Extreme Node Search (ENS), a novel and efficient anchor placement scheme, starts with two randomly placed anchors and then uses this directional transformation to identify the number and placement of anchors in a completely distributed manner. Furthermore, a novelty-filtering-based approach for identifying a set of 'good' anchors that reduces the overhead and power consumption in routing is discussed. Physical layout information such as physical voids and even relative physical positions of sensor nodes with respect to X-Y directions are absent in a VCS description. Obtaining such information independent of physical information or signal strength measurements has not been possible until now. Two novel techniques to extract Topology Preserving Maps (TPMs) from VCS, based on Singular Value Decomposition (SVD) and DVCS are presented. A TPM is a distorted version of the layout of the network, but one that preserves the neighborhood information of the network. The generalized SVD-based TPM scheme for 3D networks provides TPMs even in situations where obtaining accurate physical information is not possible. The ability to restore directionality and topology-based Cartesian coordinates makes VCS competitive and, in many cases, a better alternative to geographic coordinates. This is demonstrated using two novel routing schemes in VC domain that outperform the well-known physical information-based routing schemes. The first scheme, DVC Routing (DVCR) uses the directionality recovered by DVCS. Geo-Logical Routing (GLR) is a technique that combines the advantages of geographic and logical routing to achieve higher routability at a lower cost by alternating between topology and virtual coordinate spaces to overcome local minima in the two domains. GLR uses topology domain coordinates derived solely from VCS as a better alternative for physical location information. A boundary detection scheme that is capable of identifying physical boundaries even for 3D surfaces is also proposed.

"Network awareness" is a node's cognition of its neighborhood, its position in the network, and the network-wide status of the sensed phenomena. A novel technique is presented whereby a node achieves network awareness by passive listening to routine messages associated with applications in large-scale WSNs. With the knowledge of the network topology and phenomena distribution, every node is capable of making solo decisions that are more sensible and intelligent, thereby improving overall network performance, efficiency, and lifespan.

In essence, this research has laid a firm foundation for use of Anchor Centric Virtual Coordinate Systems in WSN applications, without the need for physical coordinates. Topology coordinates, derived from virtual coordinates, provide a novel, economical, and in many cases, a better alternative to physical coordinates. A novel concept of network awareness at nodes is demonstrated.

ACKNOWLEDGEMENTS

First and foremost I offer my sincere gratitude to my advisor, Prof. Anura P. Jayasumana, for his guidance, support and encouragement in every single step of my graduate life. Special thanks to Prof Michael Kirby, Prof Ali Pezeshki and Prof Indrakshi Ray for being in my PhD committee and for their valuable guidance.

I would also like to acknowledge my husband, Thilan Ganegedara and my parents for their constant support, guidance, encouragement and scarifies to make my dream come true. Special thanks to my colleagues, Zheng Wang, Vidarshana Bandara, Saket Doshi and Dilum Bandara for their support in many respects during my PhD study.

TABLE OF CONTENTS

NETWORK-AWARENESS IN WIRELESS SENOR NETWORKS - FOR INTELLIGENT ORGANIZATION AND EFFICIENT DATA DISSEMINATION

LIST OF TABLES

LIST OF FIGURES

CHAPTER 01

INTRODUCTION

The availability of economical, low-power miniature processors and sensors integrated with wireless communication in a single chip has resulted in systems called Wireless Sensor Networks (WSNs). WSNs applications [114] cover a large number of domains spanning physiological, habitat and environmental monitoring, condition-based maintenance, smart spaces [107], military, precision agriculture, transportation, and inventory tracking [56]. WSNs differ considerably from current networked and embedded systems. Although WSNs share the large-scale and distributed nature of other networked systems such as the Internet, their resources and capabilities are extremely limited in comparison. Increasing computing and wireless communication capabilities will expand the role of the sensors from mere information dissemination to more demanding tasks as sensor fusion, classification, and collaborative target tracking. Moreover, since WSNs do not rely on any hard-wired communication links, nodes can be deployed in places without any pre-deployed communication infrastructure.

Future WSNs can be envisioned consisting of hundreds to millions of nodes deployed in inaccessible terrains or structures, monitoring and interacting with environment for long periods of time. Due to the enormous size of such networks, use of costly and highly capable sensor nodes becomes infeasible. The main challenge with such large-scale WSNs is extending the lifespan of the entire sensor network while constructing intelligent dataaccumulating systems. In the process of designing such systems, the following restrictions and constraints have to be considered: (a) limited energy, memory, computational capabilities, and communication bandwidth; (b) cost and complexity of any hardware integrated to each node such as GPS; and (c) susceptibility of measurements such as signal strength to conditions such as fading and noise and the corresponding errors. Advances in technology have produced a vast spectrum of sensor nodes with diverse capabilities. Examples spanning from low-end motes as Radio-Frequency Identification (RFIDs) units priced from $1 to $25, with very little memory (16 bits) [97], to high-end powerful motes such as Imote2 [61] with 32MB RAM, 13MHz to 416MHz processor (Marvell PXA271), which uses three AAA batteries, alleviate the challenge on researchers. Having scalable and less complex algorithms that are capable of structuring networks and supporting data dissemination under the hardware constraints opens up the path for a range of new applications in the area of cyber physical systems [80] as well as ubiquitous networks [80]. This research develops energy- and memory-efficient algorithms as well as techniques that do not require costly additional hardware, thereby facilitating such applications with high reliability.

Virtual coordinates provide an economical alternative to geographical coordinates for routing and self-organization of large-scale WSNs. Geographical coordinate-based protocols such as Geographical Routing (GR) require physical location of nodes, which may be obtained by GPS or a localization algorithm [14]. Use of GPS is too costly and/or is infeasible for many applications. Moreover, localization using analog measurements such as signal strength and time delay is difficult and susceptible to noise, fading, and multipath interferences. Need for accurate power control and signal strength measurements increase the hardware complexity as well as the cost. Even when geographic coordinates are available, routing is carried out using Line-of-Sight distance information derived from geographic coordinates. Hence, concave physical voids and concave boundaries in the network degrade the performance of GR schemes. Traditional Virtual Coordinate Systems (VCS) characterize each node by a coordinate vector consisting of the shortest path hop distances to a subset of nodes, named anchors. These anchors are a set of ordinary sensor nodes with no additional capabilities than the other nodes in the network. Coordinates can be obtained using a controlled/organized flooding mechanism initiated by the anchors. The number of anchors becomes the network's dimensionality in the virtual coordinate space. Thus, VCS is a higher dimensional abstraction of a partial connectivity map of sensors. Features such as ease of generation and facilitation of connectivity-based routing without the need for geographical information make VCS an attractive scheme for large-scale, resource-starved WSNs. As the network's connectivity information is embedded in VCs, the physical voids are transparent in virtual space.

There are three major problems associated with VCS. First, the number of anchors required and their placement for a given network play a crucial role in the performance of VC-based routing algorithms. However, identification of the optimal number of anchors and their proper placement remain as major challenges. Under-deployment of anchors causes identical node coordinates, while over-deployment and improper placement worsen the local minima problem [23],[36], resulting in logical voids**Error! Reference source not found.**. A node unable to identify a neighbor closer to the destination than itself is called the local minima problem. Not having a proper distance estimation is the second issue. The most commonly used distance estimation is norm 2 without any justification. The third deficiency is due to the fact that VCS lose the directional information related to the node positions. Thus, existing algorithms based on VCS are limited to routing, and it is difficult to use in many applications as target tracking, boundary detection, etc., where such directional information is crucial. In this research, we go beyond the traditional VCS approaches/algorithms and derive coordinate systems with directional properties that provide efficient replacement of physical coordinate systems for various applications.

This research significantly advances the VCS domain to state-of-the-art by developing fundamental results, efficient algorithms, and techniques to achieve reliable WSN organization and data dissemination techniques. The contributions of this research can be categorized into six main sectors: (a) compressed representation of VCS and novelty analysis of each anchor coordinate; (b) regaining lost directionality and identifying a ‘good’ set of anchors; (c) generating TPMs; (d) efficient and reliable data dissemination schemes; (e) boundary identification without physical information; and (f) self-evolving sensor nets, i.e., each node in the network starts learning about the network and becoming network aware.

In traditional VCSs, an average of 6% of the nodes are used as anchors ; thus, when the network size scales the length of the VC, which is also used as node ID, the VC becomes significantly long. We proposed a compression scheme that provides a compressed representation of the VCS without degrading the original performance. Furthermore, a novelty filtering-based technique for evaluating the amount of novel information contained in an ordinate on the coordinate space created by the rest of the anchors is analyzed. This method can be used to identify unnecessary or inefficient anchors as well as good anchor locations, lowering overhead and power consumption in routing.

When a VCS is generated, the directionality information that allows characterizing relative and absolute node positions is lost due to the radial propagation of the ordinates. A Directional Virtual Coordinate System (DVCS) is proposed based on a novel nonlinear transformation that restores the lost directionality information in the VCS. The introduced virtual directionality alleviates the local minima issue present in the original VCS. The ability to specify cardinal directions based on vector-form representation and use of angle estimations among directions are radical changes from the traditional VC system approaches. Performance of VCS-based algorithms is highly dependent on the anchor selection. A novel anchor-placement scheme, based on directional transformation on two randomly selected anchors, called Extreme Node Search (ENS) is proposed. VCSs based on ENS anchors show significant improvement in underlying algorithms' performance (routing, map generation, etc.) compared to that of VCSs generated using traditional anchor selection approaches.

Physical layout information such as physical voids, boundaries, and even relative physical position of sensor nodes with respect to X-Y directions is absent in a VCS description, and obtaining the physical topology of such networks has not been possible until now. Two novel techniques to generate Topology Preserving Maps (TPMs) from VCS, based on Singular Value Decomposition (SVD) and DVCS, are presented. In addition, the proposed extension of SVD-based TPMs for 3D networks is applicable not only in WSN context but also in emerging nano-network applications[1][6] where obtaining a physical map is impractical.

The impact of proposed TPM generation without localization is immense in developing new algorithms and concepts. A technique that combines the advantages of geographic and logical routing called Geo-Logical Routing (GLR) to achieve higher routability at a lower cost is designed. It uses topology domain coordinates, derived solely from virtual coordinates

(VCs), as a better alternative for physical location information. A novel routing scheme called Directional Virtual Coordinate Routing (DVCR), which illustrates the effectiveness of the DVCS, is proposed. DVCR significantly outperforms existing VCS routing schemes, while achieving performance similar to the geographical routing scheme, but without the need for node location information. A TPM of a network is an identification of the correct embedding out of all the possible embeddings in the higher dimensional virtual domain, which facilitates the identification of network's internal and external boundaries without the need of physical location information. The proposed boundary detection is capable of identifying physical/event boundaries in 2D and 3D surfaces. Even though we illustrate the capability of the boundary detection using connectivity-based approach, this scheme can be used with physical information as well.

The future of WSNs can be envisioned as networks that evolve over time, becoming smarter while improving the performance by learning and inferring information about the network and sensed phenomena based on information gleaned from ongoing packet transmissions. Finally, a novel concept of nodes achieving such network awareness by resorting to passive listening at individual nodes' normal messages associated with data dissemination or network management in large-scale WSNs is discussed. With traditional methods, achieving network awareness requires costly localization integrated with global distribution of location information based on network flooding.

Moving beyond the existing sensor network, applications such as cyber physical systems, smart grids, and ubiquitous networks, require adaptive, self-evolving, reliable algorithms. This research lays a solid foundation for such an algorithm development platform.

The flow of the dissertation is as follows: Chapter 02 discusses the problem statement of the research. A dimension reduction technique and novelty filtering based novel information measure for each anchor in VCSs are proposed in Chapter 03. Furthermore, Chapter 04 discusses a transformation to regain the lost directionality in VCS. After investigating the properties of the proposed Directional Virtual Coordinate System (DVCS), a routing scheme that makes use of the directionality information, is discussed. Singular Value Decomposition (SVD) based TPM generation technique for 2D and 3D networks is addressed in Chapter 05, while Chapter 06 discusses energy efficient partial Multi-Dimensional Scaling based approach for TPM generation using RSSI-based VCSs. Anchor selection is one of the main challenges in VCS. Chapter 07 proposes a scheme called Extreme Node Search that provides a ‘good’ set of anchors, improving the performance not only in routing but also in TPM generation. Moreover, Chapter 07 proposes a less computationally complex DVCS-based TPM generation scheme as well. Applications of TPMs such as routing and boundary detection without the need for physical information are discussed in Chapter 08 and 09, respectively. A proposed routing scheme in Chapter 08, Geo Logical Routing, is a family of routing schemes that probabilistically combines the topological and virtual information to overcome the weaknesses in each other’s domain. A self-learning approach where each node learns and becomes aware of its environment without requiring an expensive and dedicated setup phase is discussed in Chapter 10. Finally, mathematical formulation of limitations in compressive sensing based phenomena reconstruction and its distributed implementation are addressed in Chapter 11. Chapter 12 summarizes and concludes the research work.

CHAPTER 02

PROBLEM STATEMENT

Many emerging Wireless Sensor Network (WSN) applications rely on networks of randomly deployed, wirelessly connected smart sensor nodes that are capable of self-organizing and cooperatively monitoring physical and environmental phenomena. The size of a sensor node (mote) can be as small as a millimeter-scale object that consists of a small-scale processor, memory, and radio. These smart networks (e.g., smart RFID networks, smart grids, etc.) and their spectrum of applications can be characterized by the network size and node capabilities such as communication range, memory, energy, and computational power. Thus, protocol design requirements and constraints vary depending on the target application. Subsection 2.1 discusses the variety of sensor nodes available, their limitations, and challenges in algorithm development. Subsection 2.2 addresses the importance of data dissemination and fusion and the existing challenges. Finally, Subsection 2.3 explains the problem statement and the research contributions.

2.1 Limitations and Challenges in Sensor Networks

The number of nodes in a sensor network may vary from hundreds to tens of thousands depending on the application. Future technology evolutions may result in networks with millions of nodes. Hence, algorithms for WSNs should be scalable in terms of performance and complexity. For instance, routing/structuring algorithm's communication, computation, and memory complexities should not drastically increase as the number of nodes increases. Also, in routing algorithms, routing performance should not degrade, and address the length of the addressing scheme should not grow significantly as the network's size increases. Mostly the network size depends on the deployment area, transmission, and sensing coverage of a sensor node, as well as overall budget. Sensor nodes, in general, are considered to have low memory, low energy, and low computational capabilities. However, such characteristics are highly dependent on the application requirements, network size, and the associated budget.

Sensor nodes that are currently available include RFIDs that are priced from $1 to $25, with very little memory (16 bits) [97], and TelosB [111][74] and Mica2 [60][74], which are priced around $99 with 10KB memory, 250 Kbps radio (CC2420), and a 16-bit processor (MSP430). There are powerful motes such as Imote2 [61] with 32MB RAM and 13MHz to 416MHz processor (Marvell PXA271). These low-end nodes are usually powered using AA or AAA batteries. Hence, the amount of energy available for processing and communication is strictly limited. It can be thought of as a tradeoff between the number of nodes and their capabilities, under the given budget constraints. Advancements in technology have enabled the existence of miniature and economical sensor nodes that are capable of sensing various types of physical and environmental conditions, data processing, and wireless communication. Large networks can be heterogeneous so that they contain a subset of nodes, which have larger memory, higher battery life, as well as higher computational capabilities. In this research, we are focusing on homogeneous, large-scale static sensor networks that consist of low-capability sensor nodes that operate with AA batteries, such as MICA2 and TelosB.

As mentioned earlier, most of the low-end sensor nodes are battery powered; therefore, in general it is important to limit power consumption of battery-operated devices in order to extend network's life span. A sensor node consumes power for sensing, communicating, passive listing, and data processing. Of all the processes running on sensor nodes, data/control message transmission requires the most amount of energy. The cost of wirelessly transmitting 1 Kb of data for a distance of 100 meters, in terms of energy, is approximately the same as that for the execution of three million instructions by a processor of 100 million instructions per second per Watt [74]. Evolving power harnessing technologies will allow sensor nodes to renew their energy from solar sources, temperature differences, or vibration, providing more flexibility in energy conservative algorithm design. Under energy harvesting strategies, for a perpetual sensor node operation, it must satisfy [112], $P_g \geq P_c = (1 - d)P_{idle} + d \, . P_{active}$, where P_g and P_c are the generated and consumed average powers, respectively. d, P_{idle} and P_{active} are duty cycle, power consumptions when the node is idle and active, respectively. When d is large, (i.e., the node is active for a long period) P_c would be high. Hence P_g may not be sufficient for the long-term operation of the sensor node. Even with certain battery-operated devices, one may need to limit the peak power consumption. As stated in [112], a device with a coin-size battery restricted to consume 200 µW of power on the average, has a lifespan of 167 days, which is equivalent to half a year. In such cases, the power is limited, but nodes are able to pursue long-term strategies to enhance its lifetime. The time it takes to harvest energy required to transmit a packet is E_c/P_g, where E_c is the energy per packet and P_g is the power generated by the harvesting scheme. The duration that a network is not available for sensing would grow at least linearly with the number of nodes, and it is inversely proportional to the harvested power.

Though computer networks and sensor networks have some similarities, algorithms and protocols used in computer networks cannot be directly used in limited memory and low computational power environments. Thus, novel algorithms/protocols focused on WSNs need to be developed. For instance, storing complete routing tables in a sensor node is impractical. Hence, algorithms should be designed in such a way that they address memory and computational limitations while supporting large-scale networks. Most of the large sensor networks are randomly deployed [116]. In order to perform data dissemination in an efficient manner, networks need be organized. This is also important for other applications such as boundary identification, target tracking, malfunction monitoring of equipments, and habitat monitoring. Manual configuration is not practical due to the size and, in some applications, especially those related to military, due to security purposes. Hence, sensor nodes need to organize themselves, and self-organization should be performed in an efficient and distributed manner even in large-scale networks.

2.2 Efficient and Reliable Network Self-Organization and Data Dissemination

The ability to self-organize and route messages among sensor nodes is key to the deployment of future large-scale Wireless Sensor Networks (WSNs). Routing protocols [4] play a crucial role in information fusion and dissemination in WSNs. They can be broadly categorized as content-based routing and address-based routing [23]. The former uses content-based attributes in the packet to define the destination set [18], while the latter uses some sort of position information, physical or virtual, to identify or to reach the nodes.

Physical domain schemes rely on location or physical position information [4] obtained using localization algorithms or GPS. Equipping nodes with GPS is costly and infeasible for many applications. Localization based on parameters such as RSSI is error-prone and difficult for a network of millions or even hundreds of sensors. If location information is available, Geographic Routing (GR) schemes (also called position-based routing or Geometric Routing) [69] can be used. GR possesses the advantage of having directional information, but its performance is highly correlated with localization errors [106]. GR also suffers from dead-end problems, also known as local minima problems, due to physical voids. A local minima problem is simply when a node currently holding the packet is unable to find a closer neighboring node to the destination than itself.

Virtual Coordinate System (VCS) is an alternative addressing scheme that does not require GPS or additional hardware for signal strength measurements. In VCS, each node is characterized by a M length vector, where each ordinate corresponds to the shortest path hop distance to a subset of nodes, called anchors. Therefore, the number of anchors becomes the networks' dimensionality. This hop-based coordinate system is independent of noise and interference. Furthermore, VCS has connectivity information embedded in each ordinate, so VC routing is less susceptible to 'local minima' caused by physical voids.

Performance of algorithms based on VCSs is highly sensitive to the number of anchors and their placement. The coordinates of a node may not be unique due to the insufficient number of anchors and/or their improper placement. The existence of nodes with identical coordinates degrades the routability, which is defined as the percentage of packets that successfully reach their intended destination. Although one can argue that proper anchor selection and placement prevent this problem, guaranteeing such a scenario is difficult and challenging. In virtual domain routing, similar to its counterparts in physical domain, a packet is greedily forwarded to the closest neighbor to the destination. Greedy forwarding decisions are based on information locally available to a node, i.e., coordinates of its neighbors. However, the distance function from the current node to the destination does not monotonically decrease due to the imperfections of virtual space, causing local minima that degrade the routability. These minima are referred to as logical voids and are analogous to physical voids in the physical domain. As identified, improper anchor placement and over/under placement of anchors are the main causes for this local minima problem. While increasing the number of anchors decreases the likelihood of having multiple nodes with identical coordinates, it may increase the local minima encounters during routing. Finding the overall best anchor placement is extremely challenging and, even if found, it is not likely to eliminate the local minima problem except in a limited set of network topology classes as 1D networks and 2D full grids.

2.3 Research Goal and Objectives

The goal of this research is to facilitate the deployment of large-scale sensor networks and applications by developing and evaluating reliable, efficient algorithms and techniques that do not require physical localization of nodes.

Moving beyond the discussed issues, VCSs also suffer from (a) loss of directionality and (b) lack of physical information such as relative position, voids, and boundaries in virtual domain. Thus, it is impossible to develop algorithms such as boundary detection, geographical routing, target tracking, etc., solely based on VCS. In contrast, if such information can be extracted from VCS, sophisticated algorithms that are based on physical information can be replaced with location-free VCS-based approaches. Moreover, algorithms that combine the advantages of connectivity information from VCS and derived physical and/or directional information will outperform physical-information-based algorithms. With this knowledge and understanding of VCS issues and their causes, we state the objectives of this research and accomplishments under each objective as follows.

1. **Efficient, Scalable and WSN-Friendly, Virtual Coordinate System (VCS) Based Approaches for Self-Organization**

Virtual coordinate-based algorithms, such as those for routing, are significantly beneficial by having the connectivity information embedded in the VCs. As explained above, if the number of anchors is insufficient or if they are improperly placed, the network will suffer from identical coordinates and local minima problems. Finding the optimal number of anchors and the proper placement of anchors are difficult problems to solve, especially because they are interrelated. Furthermore, some of the anchors may carry redundant information for a given node pair, and others may provide incomplete information resulting in inaccurate distance values degrading routability. Higher numbers of anchors lessen the problem due to identical coordinates, yet it increases the overall energy consumption due to increased VC address, which is used as node ID, and packet length. None of the existing literature, to our knowledge, presents a method to identify the useful information content of each anchor and reduce the dimensionality of the virtual coordinate space, without degrading the performance of the original coordinate system while decreasing the energy consumption.

A novel technique based on novelty filtering that quantifies the novel information content of each ordinate in the remaining space, thus identifying 'good' anchors for routing, is proposed in Chapter 03. Moreover, as the network size increases, the number of anchors required also increases. The higher the coordinate length, the higher the energy consumption is. A compressed representation of VCS, which provides more or less the same or sometimes better performance in routing, is proposed in Chapter 03.

2. Regaining Lost Directionality in VCS

Inadequacies associated with VCS are due to the loss of directionality information and lack of information about the physical topology. A novel transformation with which the VCS can regain its lost directionality (notion of left/right or south/east) is proposed in Chapter 04. Thus, nodes acquire some sense of physical location to supplement the connectivity information embedded in the original VCS. No additional transmission cost is involved as each node can evaluate the directional coordinate values with VCs available locally. First, properties of directional domain are derived. The ability to specify cardinal directions and use angles is a radical change from the traditional VCS approaches. Acquiring directionality provides new information hitherto not available in VCS and empowers a new approach for designing a broad spectrum of WSN algorithms. The technique to identify ‘good’ anchors alleviating the issues in VCS, called Extreme Nodes Search (ENS), is one algorithm among the proposed DVCS-based algorithms that are discussed in Chapter 07.

3. Restoring Physical Layout Information That are Absent in VCS

Many of the problems associated with VCS are due to the lack of information about physical network. Physical layout information such as physical voids, relative physical positions of sensor nodes with respect to X-Y directions, and even explicit connectivity information are absent in a VCS description. These difficulties can be overcome if information on physical topology is available. Combining VC information and position or direction information in one network essentially would combine the advantages of VC routing and geographic routing schemes. However, this has to be achieved without inheriting the disadvantages associated with obtaining location information or localization. Obtaining topological and directional information from VCS, i.e., VCS to Cartesian coordinate transformation, is challenging. Obtaining the physical topology of a network from the set of VCs has not been possible until now. We developed the theoretical basis and techniques to obtain a topology map that preserves the 2D as well as 3D physical topology of a sensor network, including the geographical voids and relative Cartesian directional information (see Chapters 05 and 06).

A vector-based representation is proposed for the DVC domain, which is then used to introduce the concept of angles between virtual directions in the transformed domain. The ability to specify cardinal directions is a radical change from the traditional VC system approaches. A novel TPM generation scheme is developed by selecting two near orthogonal directions in DVCS, as proposed in Chapter 07. This new TPM generation scheme requires significantly less computations than the existing Principle Component Analysis based methods.

4. Reliable Data Dissemination and Boundary Detection with Localization-Free Approaches

Many difficulties associated with VCS-based schemes are attributable to the lack of information about the physical network. Layout information such as physical voids and relative positions of sensor nodes with respect to X-Y directions is absent. Even though VCS is based on connectivity, explicit information on hop distances between any pairs of nodes is not available and is difficult to estimate. The absence of connectivity information, on the other hand, is the cause for local minima problems in physical domain routing. Combining connectivity information in VCS and position or direction information in a network would essentially combine the advantages of VC routing and geographic routing schemes overcoming the disadvantages in each other's domains.

A family of routing schemes called Geographic-Logical Routing (GLR), which combines the topological coordinates solely derived from VCS and VC information to achieve significantly high performance in routability, is developed as discussed in Chapter 08. The proposed non-linear directional transformation that unlocks the directional information in VCS is further investigated and is used in routing. The developed routing algorithm (discussed in Chapter 04), called Directional Virtual Coordinate Routing, provides a significant improvement in routing over existing routing approaches for WSNs.

Boundary detection plays a crucial role in information fusion and dissemination in 2D and 3D WSN applications such as target tracking, plume tracking [64], forest fires, animal migration, underwater WSNs, and surveillance applications. Identifying the boundary is also often important for self-organization of networks. Any network has a specific embedding (configuration) and can have three different types of boundaries: (1) network's outer boundary, (2) inner boundary, and (3) event boundary. Currently available boundary detection schemes that have been targeted exclusively for 2D networks can be broadly categorized into (1) physical information-based and (2) topological /connectivity information-based. The former uses the physical position of nodes to identify the boundary while the latter uses topological/connectivity information of the network. Physical domain schemes rely on node location or physical position information obtained using localization algorithms or GPS. A connectivity domain description of a network can have more than one valid embedding, even though only one of them corresponds to the physical network. Identifying the correct embedding, which is the physical topology, solely based on the connectivity information is challenging. Due to such difficulties, there is no connectivity-based approach available to identify boundaries of 3D surfaces, to the best of our knowledge. In Chapter 09, we propose a novel-connectivity-based approach for network and dynamic event boundary detection for 2D and 3D networks.

5. Intelligent Sensor Networks by Self-Learning Approaches

We envision future sensor networks as networks that evolve by long-term learning and inference, achieving over time increasing levels of network awareness, thus becoming smarter and better at what they do. We use the term “network/topology awareness” to indicate a node’s cognizance of the topology, shape, and boundary of the network and its position in that network. Upon initial deployment, the nodes would be quite oblivious of their environment, their location in the network, and the nature of the network that they belong to. The only information that the nodes are aware of is the number of neighbors they have and their IDs. Thus, a network has to rely on random routing schemes for data dissemination. In order to improve the reliability of data dissemination, network organization is critical. On the other hand, having an energy-consuming dedicated network organization phase in resource-limited WSNs reduces the network lifespan. However, over time, the nodes listen to the information disseminated in the network and infer additional knowledge and state information, leading the nodes and therefore the network to achieve network-awareness. In Chapter 10, we discuss our distributive scheme where nodes start learning from the packets that they forward until each node becomes network-aware. Then we investigated compressive-sensing-based approaches for a node to discover sensed phenomena in the network, thus making nodes phenomena-aware as we discuss in Chapter 11.

CHAPTER 03

DIMENSION REDUCTION OF VIRTUAL COORDINATE SYSTEMS IN WIRELESS SENSOR NETWORKS

3.1 Introduction

Routing protocols for Wireless Sensor Networks (WSNs) can be broadly classified into physical coordinate-based and virtual coordinate-based schemes. Physical domain routing relies on the physical (geographic) position information for routing, e.g., as in geometrical routing [4]. Virtual domain (or logical) routing is based on a set of virtual coordinates that capture the position and route information, e.g., hierarchical/clustering schemes [4] and Virtual Coordinate (VC) based routing [22][23][81]-[101]. The focus of this work is to reduce dimensionality of Virtual Coordinate Systems (VCSs) and thus enhance the energy efficiency without degrading the routability.

Virtual Coordinate based Routing (VCR) relies on a set of anchor nodes. The VCs of a node consist of the hop distance from the node to each of a set of *M* anchors. The cardinality of the coordinate is the number of anchors. The major advantage of VCR over physical domain routing is that connectivity information is embedded in the VCs; therefore, physical voids that degrade physical domain routing no longer exist in the virtual domain. Moreover, high routability can be achieved without requiring physical localization or GPS.

Most of the VCR schemes [22][23][81]-[101] use Greedy Forwarding (GF) combined with a backtracking algorithm. In GF, a packet is simply forwarded to a neighbor that is closer to the destination than the node holding the packet. VCs of nodes are used for distance evaluation between nodes as well as for node identification (ID).When a closer neighbor cannot be found, i.e., the packet is at a local minima, backtracking is employed to climb out of it. Existing anchor placement strategies [22][23][101] cannot guarantee unique IDs or 100% Greedy Ratio (GR). GR is defined as the percentage of routing requests that can be completed using GF alone.

The use of logical coordinates for routing has its own drawbacks. If the number of anchors is not sufficient or if they are not properly placed, the network will suffer from identical coordinates and local minima problems. As [34] and [36] explain, the anchors may cause local maxima in the distance function at their locations, and hence minima at other node locations. To avoid local minima problem in routing, most of the anchor placement techniques seek to select the furthest apart nodes as anchors in an attempt to push those to the network boundaries. Reference [101], for example, proposes to have all the perimeter nodes as anchors. However, identification of boundary nodes is not trivial, and it also consumes a lot of energy. Beacon vector routing uses a random set of nodes as anchors while GPS free coordinate assignment algorithm [22] uses the farthest apart triplet of anchors for routing. The latter solution results in having a large number of identical coordinates due to under deployment of anchors. Identification of farthest apart anchors also involves flooding the network several times. The anchor placement scheme in Logical Coordinate Routing [23] also follows the argument that anchors should be placed the farthest apart. In addition, the number of anchors required is network topology dependent.

Finding the optimal number of anchors and the proper placement of anchors are difficult problems to solve, especially because they are interrelated. The evaluation of the distance between nodes from their VCs is another challenge. L^1 and L^2 norms, typically used for distance evaluation, are accurate on orthogonal coordinate systems such as Euclidean space but not on radial VC systems. Furthermore, some of the anchors may carry redundant information for a given node pair, and others may provide incomplete information resulting in inaccurate distance values and degraded routability.

Higher number of anchors lessens the problem due to identical coordinates, yet it increases the overall energy consumption due to increased address (node ID) and packet lengths. None of the existing literature, to our knowledge, presents a method to reduce the dimensionality of the virtual coordinate space, to preserve the performance of the original coordinate system while decreasing the energy consumption.

Our contributions in this research are two-fold. The first is a method to evaluate the amount of novel information provided by a new anchor. As unnecessary anchors and poor anchor placement degrade the routability, this novelty parameter is useful in many ways, e.g., to determine good anchor positions and to remove redundant anchors. The second contribution is a coordinate-length reduction method based on Singular Value Decomposition (SVD) , which replaces the original VC of each node, i.e., the M-tuple with distances to each of the M anchors, with a reduced vector (an R-tuple, with $R << M$) that contains almost all the information contained in the original coordinate set. Due to the information conservation property of SVD, we are able to achieve a similar routability with new, lower-dimensional coordinates. The two steps are evaluated individually to assess their impact, and then we present results to demonstrate the net effect of the two steps combined. Using a reduced set of coordinates with minimal redundancy brings several benefits, including shorter message headers and less local minima, resulting in better routability and power efficiency. We also present centralized and distributed realizations of the algorithms in Section 3.4. The centralized implementation of the two schemes, application of novelty to eliminate redundant anchors, and reducing dimensionality of the VCs all require global information while the proposed distributed, online implementation requires coordinates of the anchors only.

Section 3.2 proposes a method to evaluate novelty of an ordinate in the subspace of remaining VCs. Section 3.3 and 3.4 discuss SVD-based method of reducing VS dimensions and dimension selection criterion respectively, while Section 3.5 explains the derivation of algorithms based on anchors' VCs. Section 3.6 explains the centralized and distributed realizations of the algorithm and their complexities. Performance evaluation is in Section 3.7, with conclusions in Section 3.8.

3.2 Novelty of a New Anchor

Let the number of anchors in the network be M. Consider the selection of a node, randomly or by some scheme, to be the $(M+1)^{th}$ anchor. We are interested in finding out what is novel in $(M+1)^{th}$ dimension defined by anchor $(M+1)$. Denote the subspace formed by the initial M anchors be S_M. If we project $(M+1)^{th}$ ordinate on to S_M, it gives us the component of $(M+1)^{th}$ ordinate that resides in S_M. The other component, which is perpendicular to the S_M, is the novel information in the $(M+1)^{th}$ dimension. That projection is called complimentary projection or residual. We use the Novelty Filter Decomposition method to evaluate it [72].

Let the coordinate matrix of the network of N nodes, with respect to M anchors be $\underline{X}$. Each row of the $N \times M$ matrix $\underline{X}$ represents a coordinate of a node, and each column of $\underline{X}$ is an ordinate with respect to an anchor. The orthonormal projection matrix [81] for the subspace S_M is given by

$$\underline{P}_M = \underline{X}\,(\underline{X}^T\underline{X})^{-1}\underline{X}^T \tag{1}$$

By definition $\underline{P}_M$ is an orthonormal matrix and $\underline{P}_M = \underline{P}_M{}^2$. Let $\underline{Y} \in \mathcal{R}^N$, a column vector, be the ordinate of the set of nodes with respect to (M+1)th anchor. Projection of $\underline{Y}$ on to the subspace formed by prior M anchors is given by

$$\underline{Y}_{P_M} = \underline{P}_M\underline{Y} \tag{2}$$

Norm of $\underline{Y}_{P_M} \in \mathcal{R}^N$, $\|\underline{Y}_{P_M}\|$ is the amount of information of $\underline{Y}$, that resides in the S_M. Furthermore, complementary orthogonal projection matrix is defined as [72]

$$\underline{Q}_M = (\underline{I} - \underline{P}_M)$$

Again by definition, Q_M is an orthonormal matrix and $\underline{Q}_M = \underline{Q}_M{}^2$.

$$\underline{Y}_{Q_M} = \underline{Q}_M\underline{Y} \tag{3}$$

Equation (3) gives the residual of $\underline{Y}$ which sits outside the S_M. Therefore, norm of $\underline{Y}_{Q_M} \in \mathcal{R}^N$, $\|\underline{Y}_{Q_M}\|$ gives us the novelty of the (M+*1*)th anchor. $\|\underline{Y}_{Q_M}\|$ is

$$\|\underline{Y}_{Q_M}\| = \sqrt{\sum_{i=1}^{N} y_i{}^2} \tag{4}$$

Even though our description considered the introduction of a new anchor, it can be used in VCS to:

1. Identify a good subset of anchors that preserves or improves routability (Section 3.7 - *C*),
2. Determine a terminating criterion for introducing anchors to a given network, and
3. Identify good anchor locations.

3.3 Reducing Dimensionality of VCS

Determining the optimal number of anchors and their placement for a given network is a critical problem for VCR. Different selections of M anchors typically result in different routabilities. Therefore, even if the coordinate generation is done in a central station, trying out different sets of anchors to identify the one with the best routability is an exhaustive procedure. Hence, having a method to reduce the dimension in such a way that the routability is unaffected due to this reduction is pragmatic in energy-limited WSN applications. If this is done at a central location, one can start with a large set of anchors and reduce dimensionality to meet the required criteria.

We begin with the Singular Value Decomposition of $\underline{X}$ which is a $N \times M$ coordinate matrix with $N >> M$.

$$\underline{X} = \underline{U}.\underline{S}.\underline{V}^T \tag{5}$$

where, $\underline{U}, \underline{S}$ and $\underline{V}$ are $N \times N, N \times M,$ and $M \times M$ matrices respectively. $\underline{U}$ and $\underline{V}$ are unitary matrices, i.e., $\underline{U}^T\underline{U} = I_{N\times N}$ and $\underline{V}^T\underline{V} = I_{M\times M}$

$\underline{V}^T$ is a basis of our data set $\underline{X}$. Then, $\underline{U}.\underline{S}$ gives the coordinates of the data $\underline{X}$ under the new basis $\underline{V}$.

$$\underline{\tilde{X}}_{SVD} = \underline{U}.\underline{S} \tag{6}$$

$\underline{\tilde{X}}_{SVD}$ is the projection of $\underline{X}$ on to $\underline{V}$, i.e., $\underline{X}.\underline{V}$. Note that the dimensions of $\underline{U}.\underline{S}$ is $N \times M$. Therefore each node still has an M-length coordinate vector.

In order to reduce the dimensions, we use the fact that $\underline{S}$ is a diagonal matrix where diagonal elements are non-negative singular values arranged in descending order. Coordinates from SVD in (6) can be rewritten as

$$\underline{\tilde{X}}_{SVD}{}^{(i)} = \underline{S}(i,i).\underline{U}^{(i)}\ ; i = 1,\dots,M \tag{7}$$

where $\underline{\tilde{X}}_{SVD}{}^{(i)}$ and $\underline{U}^{(i)}$ are the ith column of $\underline{\tilde{X}}_{SVD}$ and $\underline{U}$ respectively. Basically $\underline{U}.\underline{S}$ is each column of $\underline{U}$ weighted by the corresponding diagonal element of $\underline{S}$, i.e., $\underline{S}(i,i)$, the singular values as in (7). Therefore singular elements decide which ordinate has a significant contribution. We reduce the dimensionality by ignoring the less significant singular values of $\underline{S}$

$$\underline{\tilde{X}}_{SVD,R} = \underline{Z}_{N\times R} = \underline{U}_{N\times R}.\underline{S}_{R\times R} \tag{8}$$

$$\underline{Z}^{(i)} = \underline{S}(i,i).\underline{U}^{(i)}\ ; i = 1,\dots,R \text{ where } R \le M \tag{9}$$

where $\underline{Z}^{(i)}$ and $\underline{U}^{(i)}$ are the ith column of $\underline{Z}$ and $\underline{U}$ respectively. $\underline{\tilde{X}}_{SVD,R}$ is the new set of coordinates of the nodes and coordinate length is $R < M$.

SVD compresses the original data set in an optimal way, so it cannot improve over original values. This leads us to a new problem. What should be the new dimension R so that routability is not degraded? We address this issue in the next section.

3.4 Dimension Size Selecting Criteria

Before evaluating the error due to reduction of the dimension from M to R, let us zero-pad $\underline{Z}_{N\times R}$ so that it is of the same size as $\underline{\tilde{X}}_{SVD}$. Let us call zero-padded $\underline{Z}$, $\underline{Z}_P$. Then the distance between $\underline{\tilde{X}}_{SVD}$ and $\underline{Z}_P$ will be same as the distance between $\underline{\tilde{X}}_{SVD}$ and $\underline{Z}$, which is evaluated using Frobenius norm [120] as:

$$\left\|\underline{\tilde{X}}_{SVD} - \underline{Z}_P\right\|^2_F = \sum_{\forall i,j}(\underline{\tilde{X}}_{SVD}(i,j) - \underline{Z}_P(i,j))^2 = \left\|\underline{U}.\underline{S} - (\underline{U}_{N\times R}.\underline{S}_{R\times R})_P\right\|^2_F$$

$$= \left\|\underline{U}.\underline{S} - (\underline{U}.\underline{S}_P)\right\|^2_F$$

where $\underline{S}_P$ is zero-padded $\underline{S}$. $\underline{U}$ is an orthonormal matrix. Orthonormal matrices induce rotations. Since Frobenius norm [120] is invariant for rotations,

$$\left\|\underline{\tilde{X}}_{SVD} - \underline{Z}_P\right\|^2_F = \left\|\begin{bmatrix} 0 & & \cdots & & 0 \\ & \ddots & & & \\ \vdots & & 0 & & \vdots \\ & & & \sigma_{R+1} & \\ & & & \ddots & \\ 0 & & \cdots & & \sigma_M \end{bmatrix}\right\|^2_F \tag{10}$$

the difference between the full SVD coordinates and dimension reduced coordinates are simply given by

$$\left\|\underline{\tilde{X}}_{SVD} - \underline{Z}_P\right\|_F = \sqrt{\sum_{i=R+1}^{M} \sigma_i^2} \tag{11}$$

By defining a threshold value for the information loss in the transformation, we can get a value for the dimension $R$of the new coordinate system.

3.5 Reducing Dimensionality Based on Anchors' VC

$\underline{V}$ in (5) is evaluated based on $\underline{X}$, which is a $N \times M$ matrix that consists of VCs of the entire network. With sensor networks, it is crucial to reduce communication and computation overheads involved. This section presents a process to generate the transformation matrix $\underline{V}$ with a much smaller subset of data of $\underline{X}$, X_A, the $M \times M$ matrix corresponding to the coordinate set of only the M anchors. As M<<N, computation overhead is reduced significantly.

Let the SVD of $\underline{X}_A$be

$$\underline{X}_A = \underline{U}_A \cdot \underline{S}_A \cdot \underline{V}_A{}^T \tag{12}$$

$\underline{V_A}$is a basis for $\mathcal{R}^M$. Note that $\underline{V_A}$ has the same size as $\underline{V}$ in (5). Following the same procedure as earlier

$$\underline{\acute{X}}_{SVD,(i)} = (\underline{X})_{(i)}.\underline{V_A} \tag{13}$$

$\underline{\acute{X}}_{SVD,(i)}$is the SVD-based coordinate of the *i*th node, and $(\underline{X})_{(i)}$is the *i*th row of $\underline{X}$, i.e., VC of *i*th node. Each node can evaluate its SVD coordinate locally with the knowledge of $\underline{V_A}$. In order to reduce the dimension, each node can estimate R following the same procedure explained in Section 3.4, based on $\underline{S_A}$.

$$\underline{\acute{X}}_{SVD,R(i)} = (\underline{X})_{(i)}.(\underline{V_A})_{M\times R} \tag{14}$$

Moreover, the novelty analysis explained in Section 3.2 can be used to identify novelty of each anchor just using anchors' VC set $\underline{X_A}$. Let us define $(\underline{X_A})_{(i)}(\in \mathcal{R}^M)$ as the coordinate vector of *i*th anchor. Then projection matrix for the *i*th anchor can be written as

$$\underline{P}_{M,A_i} = \underline{B}\,(\underline{B}^T\underline{B})^{-1}\underline{B}^T \tag{15}$$

where $\underline{B}$ is $\underline{X_A}\backslash(\underline{X_A})_{(i)}$. Hence complementary orthogonal projection matrix is $\underline{Q}_{M,A_i} = (\underline{I} - \underline{P}_{M,A_i})$. Projection of $(\underline{X_A})_{(i)}$ on to the subspace *M-1* and complementary projection of $(\underline{X_A})_{(i)}$ can be found following the same procedure explained in Section 3.2. Hence the novelty of each anchor on the remaining subspace can be estimated.

3.6 Implementation in Sensor Networks

Algorithms in sensor networks are in general implemented in a distributed manner. However, certain algorithms that are required to be executed only once per network may be implemented offline and results distributed to the nodes. Next we discuss the offline and online realization of the method.

A. *Offline Realization*

As the following examples indicate, a centralized implementation does not diminish the value of the proposed methods for sensor networks.

Ex. 1: For manual sensor node deployment, e.g., when a sensor network is deployed in a building, the reduced VCs can be pre-computed and each node preconfigured with its VCs. VCR will simplify routing and achieve very high routability in spite of geographical voids.

Ex. 2: Consider a sensor network where the nodes are deployed randomly. Each node sends its neighbors information to a central station or a mobile base that traverses the region so that the adjacency matrix [121] of the network can be formed at the central station with the complexity of $O(N^2)$. Reduced coordinates are computed using the algorithms in Sections 3.2–3.4 implemented at the base station. Since adjacency matrix is available, reduced coordinates of cardinality R can be sent back to each node with an operation of complexity of $O(N^2)$. Alternatively, the mobile station traverses the region distributing the coordinates to the nodes. Centralized implementation avoids multiple flooding in the network involved in traditional anchor generation phase [22][23].

B. *Online Realization*

A distributed implementation of the above may be achieved as follows. The anchor-based VC generation is first carried out the traditional way, i.e., via flooding [4]. One of the anchors collects the set of anchors' coordinates and generates $\underline{V_A}$. It estimates R and sends first R columns of $\underline{V_A}$ to the rest of the nodes in the network with organized flooding mechanism, which requires $O(N)$ messages. Each node i can now generate $\underline{\acute{X}}_{SVD,R(i)}$ (14) locally by simply multiplying its own coordinate by $(\underline{V_A})_{M\times R}$.

3.7 Performance Analysis

The performance of the proposed methods is evaluated next for three network topologies representative of a variety of networks: Figure 3.1 a) is a 496-node circular-shaped network with three holes in the middle of the network, Figure 3.1 b) is a uniformly distributed 30-by-30 node grid with 100 missing nodes, and Figure 3.1 c) is an odd-shaped network with 550 nodes. The communication range of a node in all three networks is unity. MATLAB® 2008b is used for our simulations. VCSs are generated purely based on the adjacency matrix [121]. We used random anchor placement, but the same methodology is valid for any other anchor selection method. For consistency, we use pure GF for routing; hence the results can be expected to hold for other routing algorithms, e.g., an improved GF scheme with backtracking. Routing is considered successful if the packet is routed to the exact destination, identified by a unique node ID. GR is evaluated using Monte Carlo simulation. GR is evaluated considering random source destination pairs. If a packet is routed to the destination using GF, then that packet is counted as routed. The average of all those routed packets to total packets is defined as the Greedy Ratio, i.e.,

$$\text{Greedy Ratio\%} = \frac{\text{\# timespacketsreach thedestination}}{\text{Total \# packetsoriginated}} \times 100\% \qquad (16)$$

A. Novelty Value in Anchor Placement

Novelty of anchors was evaluated using (4) by introducing one anchor at a time. In all three networks (see Figure 3.1), anchors introduced after about 15th anchor do not contain significantly novel information. Furthermore, this method can be used to select a better anchor location for *(M*+1*)*th anchor by defining a novelty threshold for an anchor. If the novelty of $(M+1)$th anchor is less than the threshold, we can change the position of the anchor until we find a ‘good’ anchor.

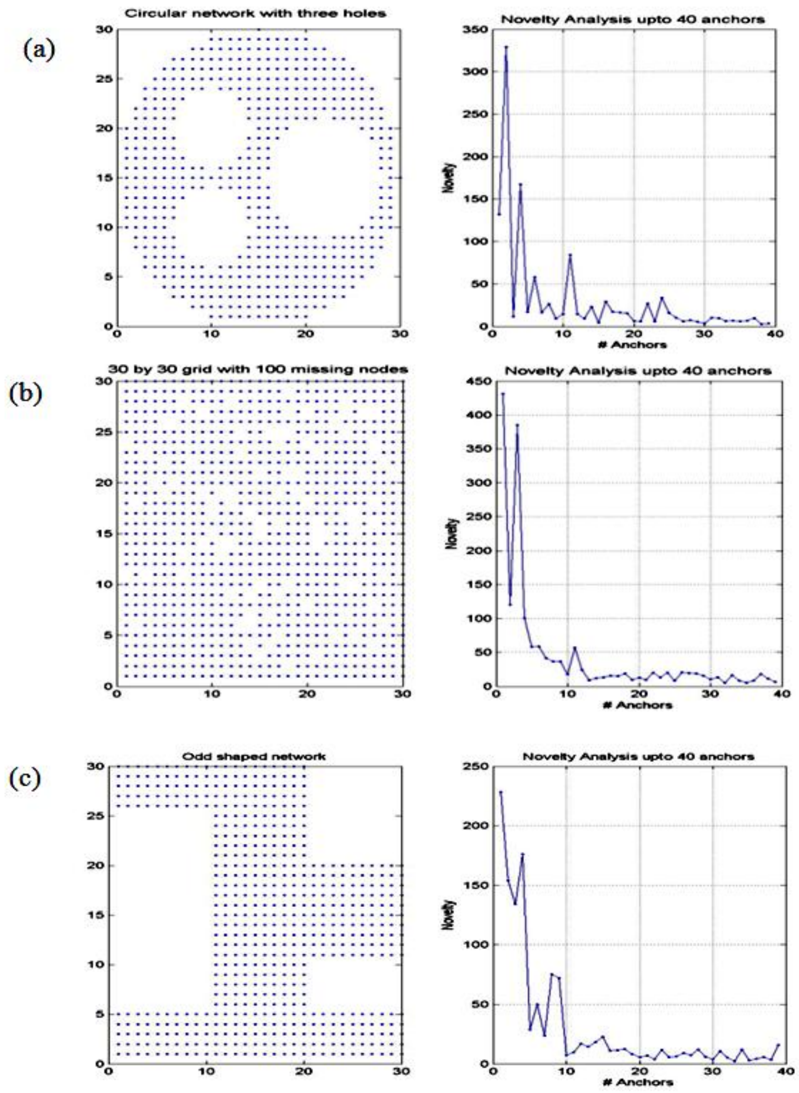


Figure 3.1. Novelty measurement as # anchors increases up to 40 for a)Circular shaped network with three holes in the middle, b) Grid based network with 100 random missing nodes, and c) Odd shaped network.

B. Dimensionality Selection Criteria and Performance of Reduced Coordinate System

Performance of new coordinates with reduced dimension given by SVD was evaluated and compared with the performance of original coordinate set. A good estimate for R in (8) was obtained using (11) for centralized implementation without using a brute-force approach. Moreover, *R* for online implementation was obtained, as explained in Section 3.6. GR of the new coordinate system should be the same as or within a bearable margin compared to the original coordinate system for both implementations; hence, ultimate selection criterion of *R* is the GR difference, where GRs were obtained by Monte Carlo simulation.

We did the simulation on the three networks in Figures 3.1 a), b), and c). In Figures 3.2 a), b), and c), we have plotted, as the dimensionality of the new coordinate system varies, the following:

1. Amount of information remaining after dimension reduction given by the dimension criteria in (11) using entire VC information,
2. Singular value corresponding to each dimension using entire VC information,
3. GR difference between original coordinate system and new coordinate system with corresponding dimension,
4. Amount of information remaining after dimension reduction given by the dimension criteria in (11) using anchors' VC information, and
5. Singular value corresponding to each dimension using anchors' VC information.

It can be clearly seen from Figure 3.2 that the singular values are approximately the same as the GR difference, so we can avoid having a Monte Carlo approach to get GR difference. Moreover, centralized implementation and online implementation criteria curves are almost the same. For implementations, *R* is 10, 5 and 5, for the three networks respectively, based on Figure 3.2.

As shown in Figure 3.3, in all 10 configurations with different random anchor placements, GR were almost the same for the original coordinate system with 40 anchors as well as reduced coordinate system for *R*: 10, 5 and 5 respectively for both centralized and distributed applications.

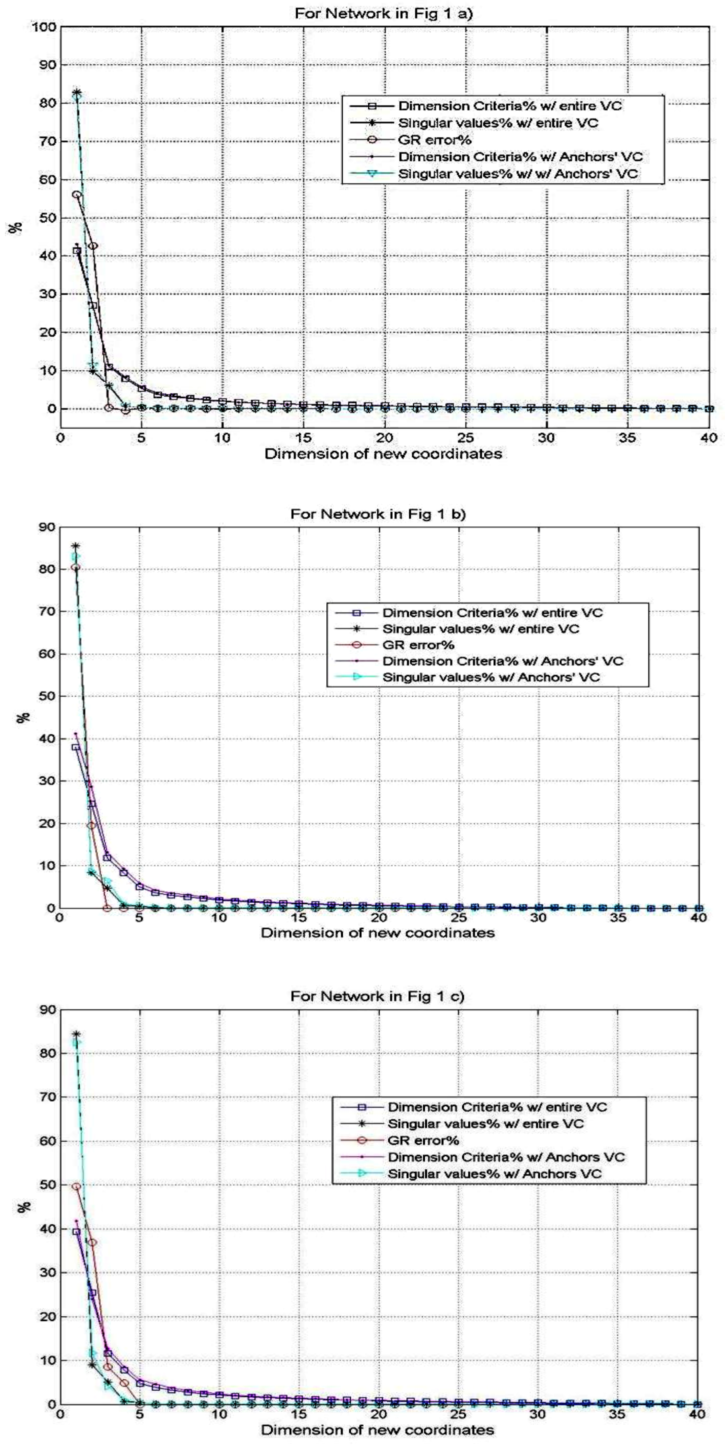


Figure 3.2. Selection of *R* for network a) Figure 3.1 a), b)Figure 3.1 b), and c) Figure 3.1 c).

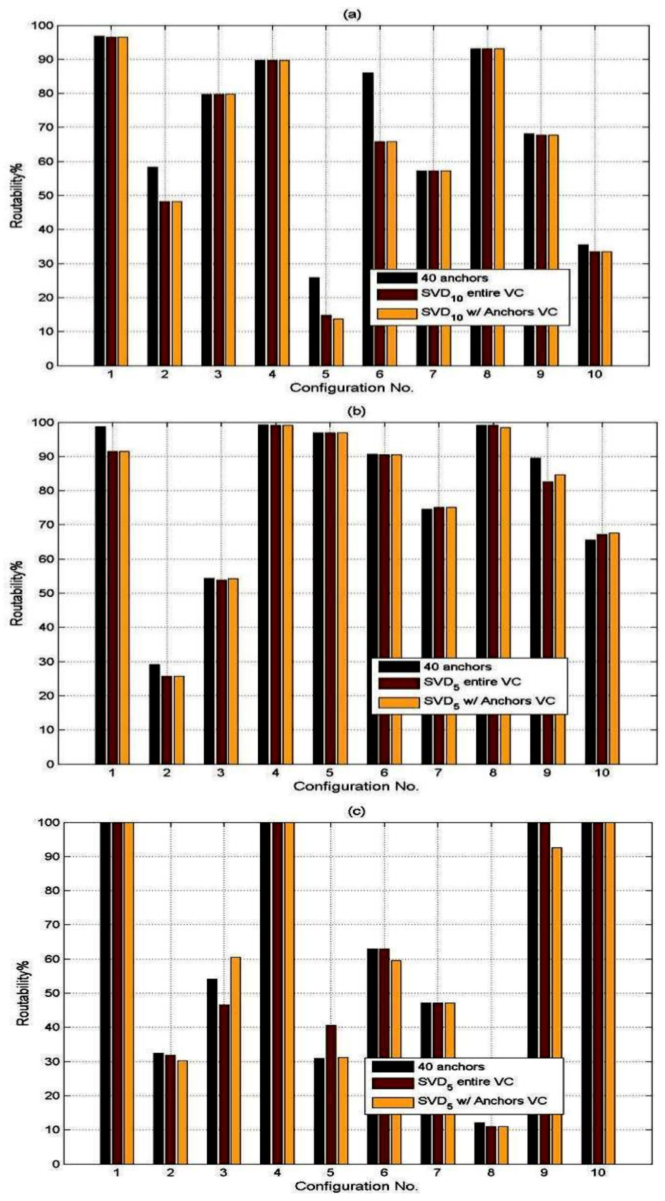


Figure 3.3. GF with entire anchor set (40 anchors) SVD based reduction with dim. 5 (SVD_5 w/ entire VC), anchor coordinate set based SVD reduction with dim. 5(SVD_5 w/ Anchors VC) a) for Figure 3.1 a), b) for Figure 3.1 b), and c) for Figure 3.1 c).

C. Novelty-Based Anchor Selection Followed by SV-Based Compression

Novelty values of anchors were used to select a good set of anchors out of the original anchor set, and that coordinate set from good anchors was further compressed using (8).

First, 40 random anchors were placed in the networks in Figures 3.1 a), b), and c). Novelty thresholds were defined based on the novelty value of *i*th anchor on the 39-anchor subspace.

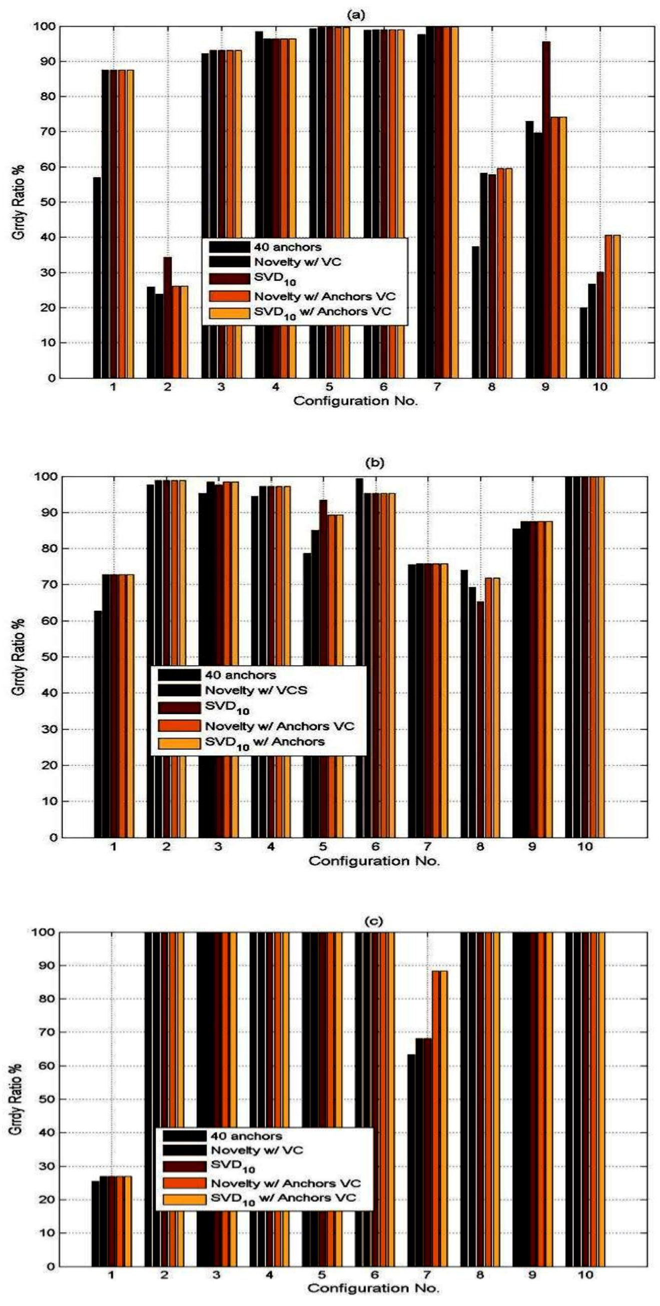


Figure 3.4. GR with original coordinates (40 anchors), coordinates selected by novelty (Novelty w/ VCS), SVD-based reduction on selected coordinate set by novelty(SVD_{10}), coordinates selected by novelty based on Anchors VC (Novelty w/ Anchors VC), and SVD-based reduction on selected coordinate set by novelty based on Anchors VC(SVD_{10} Anchors VC) in network: a) for Figure 3.1 a), b) for Figure 3.1 b), and c) for Figure 3.1 c).

In the simulation, the subset of anchors with a novelty higher than 75% of the mean novelties of anchors was selected to define the ‘good’ anchor set. Proposed methods for offline and online implementations resulted in the same decision values. The number of anchors selected by novelty method, based on entire coordinate set and just the anchors’ coordinate set, for the three topologies in 15 simulation turns with random anchor placements, are tabulated in Table 3.1. Then the coordinates based on these selected anchors was further compressed as in (8).

Table 3.1
Maximum, Minimum And Average Novelty Based Coordinate Set Size With 40 Initial Anchors, in 15 Configurations of Networks in Figure 3.1. a), b), and c)

Network	Maximum dimension selected	Minimum dimension selected	Average dimension selected
A	32	22	28
B	35	18	26
C	36	21	28

The average size of coordinate sets selected by novelty method for networks in Figures 3.1 a), b), and c) for both the implementations discussed were 28, 26, and 28 out of 40 respectively. Using SVD compression, the numbers were further reduced to R= 10, 5 and 5 respectively in online as well as in offline realizations. We have selected a general threshold for novelty just for illustrating the purpose of using the novelty information to filter good anchors. Threshold should be selected appropriately by observing the novelty plot of the anchors in each network individually. Even with the mean novelty threshold, in some of the cases selecting a subset of anchors improves the routability (see Figure 3.4). This can be explained by the fact that redundant anchors degrade the routability. Online and offline realizations give more or less the same performance.

3.8 Conclusions

In virtual coordinate-based routing in sensor networks, the higher the number of anchors, higher is the communication overhead. Although higher number of anchors reduces the probability of having identical coordinate, it does not necessarily increase the routability. Therefore, a method of identifying ‘good’ anchors or anchor locations and coordinate size reduction is essential to improve routability and energy efficiency. The first contribution of this chapter, novelty estimation, can be effectively used to find the novel information content of anchor. Different 40-anchor configurations indicate that most of the routing information can be captured by about 15 anchors. Based on the novelty value of anchors in the network, an effective set of anchors can be selected. Our results for different configurations show that the coordinate length can be reduced from 40 to 28 on average, while maintaining Greedy Routability within a narrow margin. Moreover, the proposed novelty method can be used as a tool for network partitioning based on the effectiveness of anchors in each region. The next contribution is dimension reduction based on SVD. In the example networks, it reduced the cardinality of the virtual coordinates from 40 to 5, a change by a factor 8, resulting in significant efficiencies in packet length, and implicitly in energy consumption. Basically, SVD extracts prominent features and information in the original coordinates while removing the linear dependency in the original coordinate set, resulting in reduced dimensionality. Centralized and distributed implementations of the algorithms are discussed.

CHAPTER 04

DIRECTIONAL VIRTUAL COORDINATE SYSTEMS FOR WIRELESS SENSOR NETWORKS

4.1 Introduction

Virtual Coordinate Routing (VCR) and Geographical Routing (GR) are two main classes of address-based routing schemes for WSNs. Geographical routing [4][69] relies on physical location information of nodes and directional information that can be derived from individual node locations. Obtaining location information, however, requires mechanisms like GPS, which are costly or infeasible in some applications, or localization algorithms, which are complex and error-prone as a result of their reliance on measurements such as RSSI or time delay. GR also suffers from poor routability in the presence of concave physical voids. Connectivity-based approaches provide an alternative solution to overcome weaknesses associated with location determination and geographical voids. VCR [22][23][36] uses a Virtual Coordinate System (VCS) that characterizes each node by a coordinate vector of size *M*, consisting of the shortest hop distance to each of a set of *M* anchors, which may be generated using network wide flooding [23]. The number of anchors becomes the networks' dimensionality.

Routing in virtual domain has two phases. Most of the VCR schemes [22][23][113] use Greedy Forwarding (GF) combined with a back-tracking algorithm. In GF, a packet is simply forwarded to a neighbor that is closer to the destination than the packet-holding node. Virtual Coordinates (VCs) of nodes are used for distance evaluation between nodes as well as for node identification (ID). Distance is estimated using either L^1 or L^2 norm based on VCs; such values are often unreliable estimates of the distance as the contributions due to different anchors are not orthogonal. When a closer neighbor cannot be found, i.e., the packet is at a local minima, backtracking is employed to climb out of it.

Performances of VCR and anchor placement are highly correlated. Anchors may be selected randomly [36] or by selecting nodes with specific properties, e.g., by selecting all the perimeter nodes [101]. When a message reaches a local minima, an expanding ring search is performed in [101] until a closer node is found or TTL (Time-To-Live) expires. In Virtual Coordinate assignment protocol (VCap), the coordinates are defined based on three anchors [22]. At local minima, VCap causes a packet to follow a rule called local detour. In Logical Coordinate based Routing (LCR) [23], backtracking is used when GF fails at a local minima. Aligned virtual coordinate system (AVCS) [82] re-evaluates VCs by averaging its own coordinates with neighboring coordinates to overcome local minima. In Axis-based Virtual Coordinate Assignment Protocol (ABVCap) [113], 5-tuple VC is assigned to each node corresponding to longitude, latitude, ripple, up, and down. All these VCR protocols rely mainly on Greedy Forwarding (GF), followed by a backtracking scheme to overcome local minima. Convex Subspace Routing [36], in contrast, selects dynamically changing subsets of anchors to provide convex distance surfaces for routing.

VCS has its inherent advantages and disadvantages. VCS is a connectivity-based, higher dimensional transformation of WSN, resulting in some attractive properties such as considerably high routability without any geographical information, effectiveness of connectivity information embedded in VCs, and insensitivity to physical voids and to localization errors. Physical domain to virtual domain transformation is many to one as VCs are insensitive to directions, which is one main cause of identical coordinates and local minima. If an adequate number of anchors are not appropriately deployed, it may also cause the network to suffer from identical coordinates and local minima [36], resulting in logical/virtual voids. Identification of the optimal number of anchors and proper anchor placement remains a major challenge [36].

Inadequacies associated with VCS are due to loss of directionality information and the lack of information about physical network topology. This chapter proposes, for the first time, novel transformation with which the VCS can regain its lost directionality, thus acquiring some sense of physical location, to supplement the connectivity information embedded in original VCS. No additional transmission cost is involved, as each node can evaluate the directional values with VCs available locally. Acquiring directionality provides new information, hitherto not available in VCS, facilitating a new approach for designing a broad spectrum of WSN algorithms. A technique to identify 'good' anchors alleviating the issues in VCS discussed in [36], novel routing schemes and generating Topology Preserving Maps (TPMs) with lower cost compared to the proposed scheme in [37] are among potential applications of Directional Virtual Coordinate Routing (DVCS). As an example, we illustrate a deterministic algorithm for routing in a constrained tree topology, based on new, transformed coordinates in directional virtual space. To our knowledge, no deterministic algorithms have been developed before using the VC domain.

In sum, the contribution of this research can be listed as below:

1. Novel concept of transforming directionless VCS to directional VCS

2. Properties in the new directional VCS and deterministic routing in a constrained tree network
3. Directional Virtual Coordinate Routing (DVCR) - Routing in directional VCS

The proposed routing scheme in directional virtual space, called Directional Virtual Coordinate Routing (DVCR), is compared with CSR [36] and LCR [23]. Moreover it is compared with a geographical routing scheme called Greedy Perimeter Stateless Routing (GPSR) [69] which makes greedy forwarding decisions until it fails, for example due to a geographical void, and attempts to recover by routing around the perimeter of the void. DVCR outperforms CSR and LCR with a noticeable value, achieving more or less the similar performance as GPSR.

Section 4.2 explains the new transformation of VCS to directional VCS. In Section 4.3, a deterministic routing based on directional coordinates in a simple tree is discussed. A novel routing protocol is proposed in Section 4.5. Performance evaluation is in Section 4.6. Finally, Section 4.7 concludes the contribution of this research.

4.2 Directionless Virtual Space to Directional Virtual Space Transformation

As a virtual coordinate corresponds to the distance to a particular anchor, the physical domain to virtual domain transformation is many to one. The coordinate propagates concentrically around the anchor, thus losing the directional information. Consequences of this mapping include identical coordinates and local minima encountered in routing [36]. A novel transformation of VCs to regain the directionality lost is proposed next. The notations used are summarized in Table 4.1.

Table 4.1
Summary of the Notations used in Chapter 4

Notation	Description
N	Number of network nodes
$N_i, N_d, N_s \in N$	Node i, Destination, Source
M	Number of anchors
$A_i, i = 1{:}M$	Anchor set (a subset of N)
$h_{N_iN_j}$	Minimum hop distance between node N_i and N_j
$[h_{N_iA_1}, \dots, h_{N_iA_M}]$	Node N$_i$*'s* VC
$[n_{i1} \dots n_{ij} \dots n_{iL}]$; $j = 1{:}L, L \le C_2^M$	Node N$_i$*'s* transformed VC
$D_{N_iN_j}$	Distance between N_i and N_j in transformed domain
K	Neighbors set
$N_{i,Prev}$	Node that forward the packet to current node
$N_{i,Next}$	Node that current node will forward the packet

First consider a 1D network where one can easily visualize the concept behind the transformation. Table 4.2 contains the VCS for the 1D network shown in Figure 4.1 with respect to two anchors A_1 and A_2, which are $h_{A_2A_1}$hops apart (8 hops in Figure 4.1). Note that $h_{N_iA_1}$ propagates symmetrically from the corresponding anchor, thus losing directionality. Even though $(h_{N_iA_1} - h_{N_iA_2})$ provides the sense of directionality for the region between anchors, as can be seen in Table 4.2, it remains constant outside the region bounded by anchors, thus failing to provide directional information. Conversely, $(h_{N_iA_1} + h_{N_iA_2})$ has a constant value in between the anchors, but linearly varies elsewhere. By combining those, a node N_i is characterized using $f\left(h_{N_iA_1}, h_{N_iA_2}\right)$ that is defined as,

$$f\left(h_{N_iA_1}, h_{N_iA_2}\right) = \frac{1}{2h_{A_2A_1}}\left(h_{N_iA_1} - h_{iA_2}\right)\left(h_{N_iA_1} + h_{N_iA_2}\right) \qquad (1)$$

$f\left(h_{N_iA_1}, h_{N_iA_2}\right)$, as shown in Table 4.2, maps the nodes' VC $\equiv (h_{N_iA_1}, h_{N_iA_2})$ linearly to the real axis with positive and negative values with center at the midpoint of A_1 and A_2, providing directional information in the virtual domain. The term $\frac{1}{2\, h_{A_2 A_1}}$ normalizes the distance to provide a unit difference of the ordinate between two adjacent nodes.

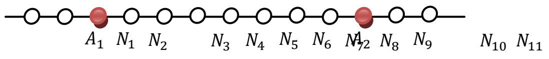


Figure 4.1. 1D network with two anchors A_1 and A_2.

Prior VC-based routing schemes such as LCR [23] encountered local maxima at anchors even for this simple linear array [36], but with this transformation, the local minima problem is overcome completely.

Table 4.2
Example VC transformation Steps in 1D Network shown in Figure 4.1

Node ID	N_1	N_2	A_1	N_3	N_4	N_5	N_6	N_7	N_8	N_9	A_2	N_{10}	N_{11}
h_{iA_1}	2	1	0	1	2	3	4	5	6	7	8	9	10
h_{iA_2}	10	9	8	7	6	5	4	3	2	1	0	1	2
$h_{iA_1} - h_{iA_2}$	-8	-8	-8	-6	-4	-2	0	2	4	6	8	8	8
$h_{iA_1} + h_{iA_2}$	12	10	8	8	8	8	8	8	8	8	8	10	12
$f(h_{iA_1}, h_{iA_2})$	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6

Furthermore, one can now view each node as a point in a vector space. Define, $\vec{\boldsymbol{u}}_{\boldsymbol{A_1 A_2}}$ as the unit vector in $A_1 A_2$ direction, which is named as a *virtual direction*. Ordinate in Figure 4.1 can be written in the form:

$$\vec{\boldsymbol{f}}\left(\boldsymbol{h}_{\boldsymbol{N_i A_1}}, \boldsymbol{h}_{\boldsymbol{N_i A_2}}\right) = f\left(h_{N_i A_1}, h_{N_i A_2}\right) \vec{\boldsymbol{u}}_{\boldsymbol{A_1 A_2}} \qquad (2)$$

Now consider a 2D sensor network. Select two coordinates of node N_i: $h_{N_i A_j}$ and $h_{N_i A_k}$, with respect to anchors, A_j and A_k. Then the magnitude of the virtual distance vector component in the $A_j A_k$ direction is given by

$$f\left(h_{N_iA_j}, h_{N_iA_k}\right) = \frac{1}{2h_{A_jA_k}}\left({h_{N_iA_j}}^2 - {h_{N_iA_k}}^2\right) \tag{3}$$

Since there are M ordinates available, C_2^M different virtual directions (though not orthogonal to each other) may be specified and can be evaluated locally at each node. Some of the properties in this directional Virtual Space are discussed next. Each transformed domain ordinate can be written in the form:

$$\vec{\boldsymbol{f}}\left(\boldsymbol{h}_{\boldsymbol{N_iA_j}}, \boldsymbol{h}_{\boldsymbol{N_iA_k}}\right) = f\left(h_{N_iA_j}, h_{N_iA_k}\right)\vec{\boldsymbol{u}}_{\boldsymbol{A_jA_k}} \tag{4}$$

where $\vec{\boldsymbol{f}}\left(\boldsymbol{h}_{\boldsymbol{N_iA_j}}, \boldsymbol{h}_{\boldsymbol{N_iA_k}}\right)$ is the vector representation of the transformed ordinate of A_iA_j and $\vec{\boldsymbol{u}}_{\boldsymbol{A_jA_k}}$ is the virtual direction obtained by A_jA_k. We can also define the virtual distance between two nodes N_p and N_q in this direction to be

$$F_{A_jA_k}(N_p, N_q) = f\left(h_{N_pA_j}, h_{N_pA_k}\right) - f\left(h_{N_qA_j}, h_{N_qA_k}\right) \tag{5}$$

Figure 4.2 shows the 2D extension of the transformation to a grid network. Transformed coordinates are given by $f(A_1, A_2)$, $f(A_3, A_4)$], providing directionality and dividing the grid into four quadrants.

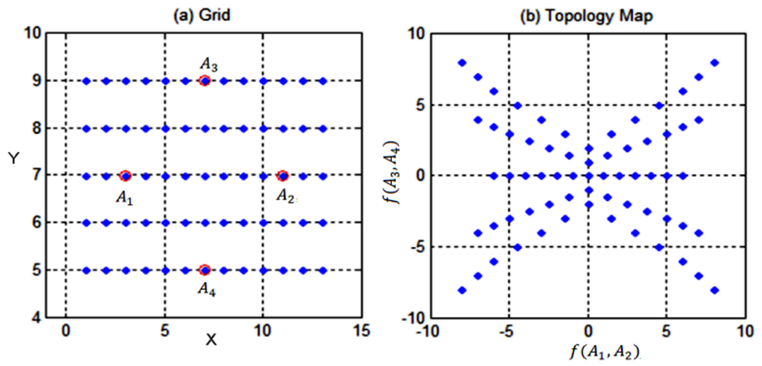


Figure 4.2. (a) Physical map of the grid and (b) Directional domain map, $f(A_1, A_2)$ Vs. $f(A_3, A_4)$, of 2D grid.

Property 1: Consider a 2D network with two anchors A_1 and A_2, which are $h_{A_1A_2}$ hops apart, the transformation $f\left(h_{N_iA_1}, h_{N_iA_2}\right) = \frac{1}{2h_{A_1A_2}}\left({h_{N_iA_1}}^2 - {h_{N_iA_2}}^2\right)$ partitions the network into two sections with positive and negative ordinates. Moreover, the nodes that are equidistance in terms of hops to A_1 and A_2 have $f\left(h_{N_iA_1}, h_{N_iA_2}\right) = 0$.

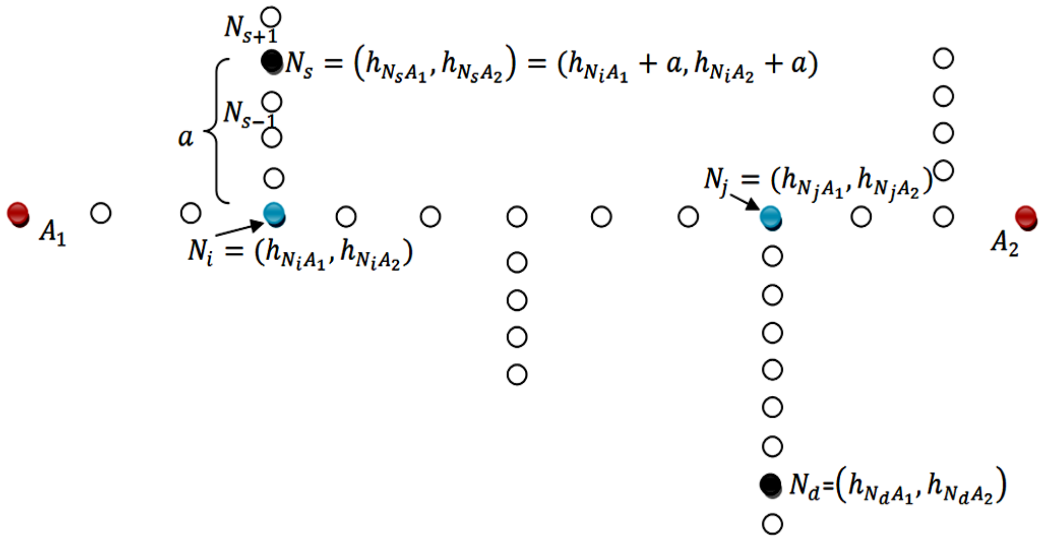
Proof: Consider the transformation in Eq. (3). The nodes with $h_{N_iA_1} > h_{N_iA_2}$ have positive transformed ordinates while nodes with $h_{N_iA_1} < h_{N_iA_2}$ have negative transformed ordinates. Nodes with $h_{N_iA_1} = h_{N_iA_2}$have zero ordinates in the transformed domain. QED

By randomly or systematically selecting more anchor pairs, the network can be partitioned based on the sign of each ordinate as in a four-quadrant Cartesian coordinate system. Partitioning networks based on the sign is demonstrated in Section 4.5. The relationship of (3) allows the derivation of distance and coordinate relationships that in turn allow for systematic, and even deterministic, methods for routing using VCs. To our knowledge, this is the first instance of use of VCs this way. We illustrate the use of the directional information using a simple example next. A routing algorithm based on DVCS is presented and evaluated in Section 4.5.

4.3 Routing in Directional Virtual Domain

In this section, we demonstrate how it is possible to exploit properties of DVCS to develop strategies to identify routing paths, which was not feasible with directionless VCS. For the example presented in this section, that of a simple tree, a DVCS-based deterministic routing protocol can be used to guarantee 100% routability. This direction-based method can be considered as a foundation for developing relationships to discover routing paths in more complex topologies.

Consider a Constrained Tree (CT) network with branches extending to both the sides off one main trunk (backbone). Assume that the maximum degree of a node is three, i.e., no two branches occur at same point, and there are no branches off branches (see Figure 4.3). CT network topologies fit well in environments such as mine-shafts and pipeline distribution systems. The traditional VCR schemes, such as LCR and CSR, cannot guarantee 100% routability in these networks. Consider the network shown in Figure 4.3, where a packet is to be sent from node $N_s \equiv (7,13)$ to $N_d \equiv (16,10)$. In this case, the packet will be forwarded to N_{s+1} in Greedy Forwarding based on VCs, whereas the correct neighbor to forward the packet is N_{s-1}.



N_{s+1}
$N_s = (h_{N_sA_1}, h_{N_sA_2}) = (h_{N_iA_1} + a, h_{N_iA_2} + a)$
N_{s-1}
a
$N_j = (h_{N_jA_1}, h_{N_jA_2})$
A_1
A_2
$N_i = (h_{N_iA_1}, h_{N_iA_2})$
$N_d = (h_{N_dA_1}, h_{N_dA_2})$

Figure 4.3. Constrained tree network with two anchors A_1 and A_2.

Property 2: In a CT, the gap between any two adjacent nodes in a branch is constant, and uniquely dependent on the junction node. Specifically, for a branch off node $N_i = (h_{N_iA_1}, h_{N_iA_2})$, the gap is given by

$$\boldsymbol{g} = (\boldsymbol{h_{N_iA_1}} - \boldsymbol{h_{N_iA_2}})/(\boldsymbol{h_{N_iA_1}} + \boldsymbol{h_{N_iA_2}}) \tag{6}$$

Proof: Junction node coordinates are unique. Consider the junction node $N_i = (h_{N_iA_1}, h_{N_iA_2})$. A positive integer x which makes $(h_{N_iA_1} + x) - (h_{N_iA_2} - x)$= $h_{N_iA_1} - h_{N_iA_2}$ does not exist. Hence the gap in each branch given by (6) is unique. QED

Property 3: In a CT, only the members of the backbone $N_i = (h_{N_iA_1}, h_{N_iA_2})$ satisfies

$$\boldsymbol{h_{N_iA_1}} + \boldsymbol{h_{N_iA_2}} = \boldsymbol{h_{A_1A_2}} \tag{7}$$

Proof: This can be proven by the characteristics of VC. QED

Property 4: In a CT, junction node $N_i = (h_{N_iA_1}, h_{N_iA_2})$ can identify the members on its branch and how many hops that each member of the branch is away from itself.

Proof: As in property 2, a gap in a branch is unique and is known by the branching node. Assume a node $N_d = \left(h_{N_dA_1}, h_{N_dA_2}\right)$ is a member of the branch from junction N_i. Then virtual ordinates of N_d satisfies $h_{N_dA_1} = h_{N_iA_1} + ga$ and $h_{N_dA_2} = h_{N_iA_2} + ga$. g can be found as in (6). Thus 'a' , the number of hops to the backbone from N_d can be calculated. Consider another junction node $N_j = (h_{N_iA_1} + x, h_{N_iA_2} - x)$ with gap in its branch g'. There does not exist a positive integer x and a', which satisfy $h_{N_dA_1} = h_{N_iA_1} + x + g'a'$ and $h_{N_dA_2} = h_{N_iA_2} - x + g'a'$. Hence N_d exists only in the branch of N_i. QED

Theorem 1: In a tree with branches on two sides off one main trunk, with no branches off branches, with node degree ≤ 3, 100% routability can be achieved with two anchors placed, one at each extreme of the trunk.

Proof: If the destination is on the same branch as the source, the source can ascertain it by evaluating the gap between itself and its neighbor, and it can forward the packet toward or away from branch node correctly. Otherwise, the routing is performed in two steps. Initially, the packet is routed to the junction node where the current node holding branch connects to the backbone. Then the packet is routed to the destination from the junction node, but the current node should find out the coordinates of the junction node. Consider Figure 4 3. Let the current node be N_s and destination be N_d. The distance between anchors, $h_{A_1A_2}$, in (7) is known and (6) is simply the difference between the current node and the neighbor. Therefore the VC of the branching node $(h_{N_iA_1}, h_{N_iA_2})$ can be found. Thus'a' , the number of hops to the backbone, can be found. If a and $(h_{N_iA_1}, h_{N_iA_2})$ are known, a packet can reach the backbone, i.e., node $\left(h_{N_iA_1}, h_{N_iA_2}\right)$, and then it can be routed to the destination. Any junction node can identify whether N_d is in its branch or not. If N_d is a member of the branch, the junction node will forward the packet to its neighbor on the branch. If N_d is not a member of its branch, the junction node will forward the closest neighbor, excluding the neighbor on the branch, to N_d. QED

A_1 and A_2 need not be at corners but we should make sure all the branches are in between A_1 and A_2. Moreover, if the number of nodes in-between A_1 and A_2 is odd, and if there is a branch at the middle point, all the nodes in that branch will have zero ordinate (see proof of Property 1); this can be avoided by assuring hop distance between A_1 and A_2 to be even when anchor A_2 is selected. Furthermore, in a tree with branches on two sides (provided branches are off one main trunk), when there is a branch off a branch, it can be treated as a sub-constrained tree network, hence the need to add exactly one additional anchor. Routing should be done in each sub- constrained tree based on corresponding anchors. This will allow us to get the number of anchors needed for any tree, i.e., any graph without cycles. A simple adjustment can be proposed if there are two branches at the same node. After generating VCS with respect to anchors A_1 and A_2, members of the backbone and branching nodes that have two branches can identify themselves. After that, junction nodes with two branches can add one more bit to the coordinate of the nodes in one of the branches to indicate whether it's the upper or lower branch. This newly added bit can be used to prevent identical coordinates in the upper and lower branches. Theorem 1 holds for the network after the small adjustment. Moreover, as observed, the same approach can be applied in a tree network with degree 3.

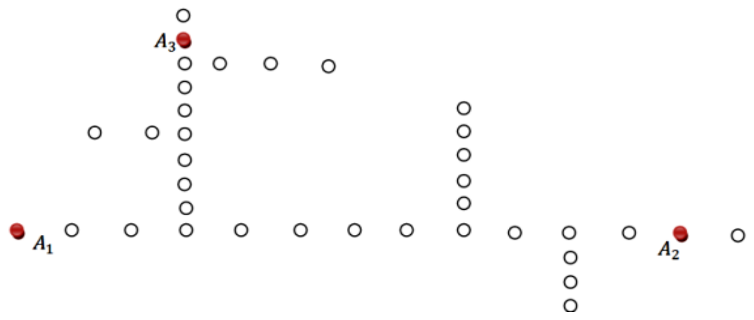


Figure 4.4. Constrained tree with branches in a branch, which can be modeled as a sub constrained tree.

4.4 Simulation Results: Partitions in Directional Virtual Space

The effectiveness of the directional domain is evaluated in five representative examples of a variety of networks, as shown in Figure 4.5: (a) spiral-shaped network with 421 nodes, (b) a grid-based network with 100 randomly missing nodes (800 nodes), (c) a 496-node, circular-shaped network with three physical voids/holes, (d) a network of 343 nodes mounted on walls of a building, and (e) odd-shaped network with 550 nodes. Communication range of a node in all five networks is unity. MATLAB® 2009b was used for the computations.

As stated in property 1, the sign of each ordinate in transformed domain is used to identify different sectors of the network. In the networks shown in Figure 4.5, three randomly selected anchors,A_1, A_2,and A_3 were placed. Then, using the transformation given in (3), the new coordinates, $\left[f\left(h_{iA_1}, h_{iA_2}\right), f\left(h_{iA_1}, h_{iA_3}\right), f\left(h_{iA_2}, h_{iA_3}\right)\right]$ were generated by each node locally. Based on the sign of each ordinate in the directional coordinate, i.e., positive/negative, different sections were colored as shown in Figure 4.5. Since three anchors' ordinates are used for transformation, cardinality in transformed domain is 3 (C_2^M). Hence the maximum possible sign combinations in the network is 2^3. As in Figure 4.5, not all the sign combinations exist but existing combinations clearly partition the network.

4.5 Directional Virtual Coordinate Routing (DVCR)

In this section, we present a novel routing scheme based on transformed coordinates. In a network with M randomly selected anchors, a node can evaluate its transformed coordinates of cardinality $P = C_2^M$ locally. Let the transformed current node coordinate be $N_i \equiv \left[n_{i1} \ldots n_{ij} \ldots n_{iP}\right]$ and that of the destination be $N_d \equiv \left[n_{d1} \ldots n_{dj} \ldots n_{dP}\right]$. L^2 distance between N_i and N_d is using transformed coordinates is

$$D_{N_i N_d} = \sqrt{\textstyle\sum_{\forall j}(n_{ij} - n_{id})^2}\,; j = 1{:}J \leq C_2^M \qquad (8)$$

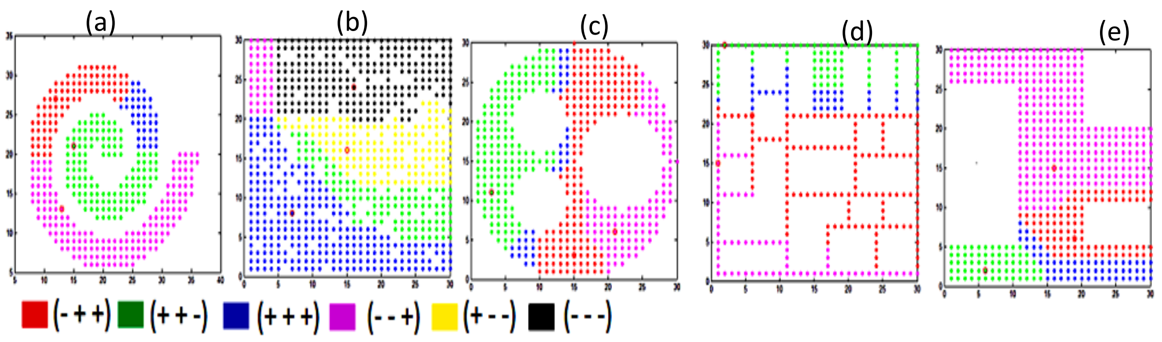


Figure 4.5. Partitions of (a) Spiral shaped network with 421 nodes, (b) A grid based network with 100 randomly missing nodes (c) A 496-node circular shaped network with three physical voids/holes, (d) A network of 343 nodes mounted on walls of a building (e) Odd shaped network with 550 nodes, based on the sign of the ordinates in transformed domain created by three randomly selected anchors A_1, A_2, A_3. Transformed domain has three ordinates generated by (A_1, A_2), (A_1, A_3), and (A_2, A_3) pairs.

The packet is forwarded to a neighbor using Greedy Forwarding (GF). To overcome the local minima, the minima node performs an approximate hop distance estimation from itself and also from neighbors as explained next, based on (9), (10) and (11). The assumption is that L^1 in transformed domain (see (5)) is a good representation of the hop distance. Hence there exists a neighbor which has lower hop distance (estimated) to destination. For this estimation, two directional ordinates are sufficient.

Define the ordinate difference set $\Delta_{A_pA_q}$ between current node N_i and all the neighbors $N_k \in K$ as

$$\Delta_{A_pA_q} = \left|\left(F_{A_pA_q}(N_i, N_k)\right)\right| \ ; \ \ N_k \in K \tag{9}$$

Since there are $|K|$ number of neighbors, size of $\Delta_{A_pA_q}$ is the same as that of $|K|$. Consider $\Delta_{A_1A_2}$ with respect to $f(A_1, A_2)$. Let α_{12} and β_{12}, be $\max(\Delta_{A_1A_2})$ and $\min(\Delta_{A_1A_2})$ respectively. Then the approximate hop distance between N_i and N_d is represented with respect to $f(A_1, A_2)$,

$$\alpha_{12}n + \beta_{12}m = |\left(F_{A_1A_2}(N_i, N_d)\right)| \tag{10}$$

Similarly α_{34} and β_{34} is obtained following the same method with respect to another ordinate $f(A_3, A_4)$. Another representation of the approximate hop distance between N_i and N_d with respect to $f(A_3, A_4)$,

$$\alpha_{34}n + \beta_{34}m = |\left(F_{A_3A_4}(N_i, N_d)\right)| \tag{11}$$

By solving (10) and (11), the approximate hop count from current node to destination, which is $+m$, can be estimated. This can be repeated for neighbors set K to get the hop distances from neighbors to destination. Packet will be greedily forwarded to the neighbor selected by this hop count approximation. Algorithm of the routing protocol can be summarized as in Figure 4.6.

4.6 Performance of Directional Virtual Coordinate Routing

The performance of proposed Directional Virtual Coordinate Routing (DVCR) scheme is evaluated next, for the five networks introduced in Figure 4.5. Performance of DVCR is compared with two virtual coordinate-based routing schemes, Logical Coordinate Routing (LCR) [23] and Convex Subspace Routing (CSR) [36], and a geographic routing scheme, Greedy Perimeter Stateless Routing (GPSR) [69]. Five randomly selected nodes, serving as anchors in order to illustrate the performance of proposed routing scheme, are independent of the anchor placement. In LCR implementation, we assumed that the entire path traversed is available at each node so that backtracking can be perfectly performed avoiding any loops; that is, the implemented case is the best case of LCR and is not achievable in practice due to the cost involved in transmitting the required information. Time-To-Live (TTL) of the packet is set to 100 hops.

```
Input: VCS
Output: Closest neighbor to destination
while ($\boldsymbol{N_d} \neq \boldsymbol{N_i}$ || $\boldsymbol{TTL} \geq \mathbf{0}$ ) % $TTL$- Time-To-Live
        $\boldsymbol{R_k} = \boldsymbol{D_{jd}}$ ; $\boldsymbol{N_j} \in \boldsymbol{K}$; $\boldsymbol{N_j} \neq \boldsymbol{N_{i,Prev}}, \boldsymbol{N_{i,Next}}$ % Calculate the transformed domain distance
from Neighbors set K to destination excluding
        $\boldsymbol{N_{i,Next}}$ and $\boldsymbol{N_{i,Prev}}$
        $\boldsymbol{d_{jd}} = \boldsymbol{D_{jd}}$ %Current distance to desination

  if $\boldsymbol{Min}(\boldsymbol{R_k}) == \mathbf{0}$
        if $\boldsymbol{N_d} == \boldsymbol{N_i}$
            ROUTED
        else % if identical coordinates
            Find $\boldsymbol{n}+\boldsymbol{m}$ for $\boldsymbol{N_i}$ and $\boldsymbol{N_j} \in \boldsymbol{K}$; $\boldsymbol{N_j} \neq \boldsymbol{N_{i,Prev}}, \boldsymbol{N_{i,Next}}$ and pick the closest neighbor.
If no closer neighbor exists routing fails
        end
  elseif $\boldsymbol{Min}(\boldsymbol{R_k}) \leq \boldsymbol{d_{id}}$
         $\boldsymbol{N_{i,Prev}}$ = $\boldsymbol{N_i}$
         $\boldsymbol{N_i}$ = $\underset{\boldsymbol{j}}{\boldsymbol{argmin}} \boldsymbol{R_k}(\boldsymbol{j})$
         $\boldsymbol{N_{i,Next}}$ = $\underset{\boldsymbol{j}}{\boldsymbol{argmin}} \boldsymbol{R_k}(\boldsymbol{j})$
   elseif $\boldsymbol{Min}(\boldsymbol{R_k}) > \boldsymbol{d_{id}}$ %Local minima
            Find $\boldsymbol{n}+\boldsymbol{m}$ for $\boldsymbol{N_i}$ and $\boldsymbol{N_j} \in \boldsymbol{K}$; $\boldsymbol{N_j} \neq \boldsymbol{N_{i,Prev}}, \boldsymbol{N_{i,Next}}$ and pick the closest neighbor.
If no closer neighbor exists routing fails
   end
end
```

Figure 4.6. Pseudo code of DVCR algorithm.

Average routability, average path length that packets traversed, and average energy consumption per successfully delivered packet are used as the performance metrics. Average routability evaluation considers all source-destination pairs, i.e., each node generated a set of (N-1) messages, with one message for each of the remaining node as destination.

$$\text{Average routability} = \mathrm{R_{AVG}}\% = \frac{\text{Total \# of packet that reached the destination}}{\text{Total number of packet generated}}\% \tag{12}$$

$$\text{Average path length} = \mathrm{H_{AVG}} = \frac{\text{Cumilative number of hops that each packet traversed}}{\text{Total number of packet generated}} \tag{13}$$

Note that the average path length calculation includes the path lengths for unrouted messages as well.

In order to have a fair estimation of the energy consumption, average energy consumption per successfully routed packet was defined as

Average energy consumption per successful packet delivery

$$= E_{AVG} = \frac{E_{\propto} \times H_{AVG} \times Packet\ size}{R_{AVG}\%} \tag{14}$$

where $E_{\propto}$ is average energy per byte. For all the routing schemes, a fixed packet length of 12 bytes was assumed, with 4 bytes each for destination ID, current node ID and VC/Physical coordinates. The random anchor placement performance was averaged over five random anchor configurations.

Performance comparison in terms of $\mathrm{R_{AVG}}\%$, $\mathrm{H_{AVG}}$ and $\mathrm{E_{AVG}}$ are as shown in Figure 4.7. With random anchor placement, the proposed scheme DVCR outperforms CSR and LCR with $\mathrm{H_{AVG}}$ to shortest distance path length ($\mathrm{H_s}$) ratio close to unity.

Also it outperforms GPSR in spiral and grid with missing nodes and achieves almost the same performance in the rest of the networks. Even though DVCR achieves a higher $R_{AVG}\%$, E_{AVG} (see Figure 4.7 (c)) is less than that of GPSR while very close to E_{AVG} of CSR and LCR. It is important to note that GPSR relies on accurate location information, achievable via expensive hardware such as GPS, or localization schemes subject to significant complexity and estimation errors. The importance of directionality information was illustrated by the performance of DVCR and anchor selection mechanism. More importantly, the required number of anchors is 5, which is a significantly lower number compared to the anchors used in other literature, to achieve $R_{AVG}\%$ over 95%. And the proposed scheme does not require any costly anchor placement, though proper anchor placement increases the routability of DVCR, as observed.

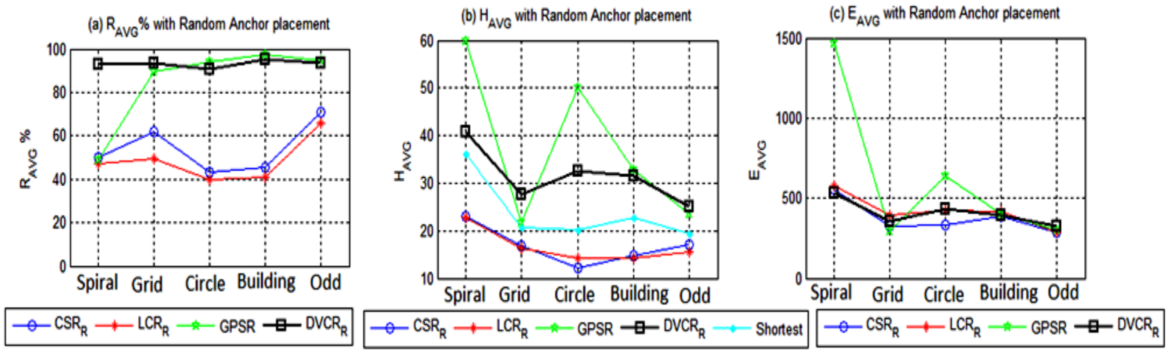


Figure 4.7. (a) $R_{AVG}\%$ (b) H_{AVG}(c) E_{AVG} of CSR, LCR, GPSR and DVCR with 5 randomly placed anchors in Spiral, 30 by 30 node Grid with 800 nodes, Circle with 3 holes, building and odd networks.

4.7 Conclusions

A simple and novel transformation is proposed for virtual coordinates that, for the first time, allow VCS to recover directionality lost during the coordinate generation, thereby significantly increasing the effectiveness of virtual coordinate systems in routing. The issues such as identical coordinates and local minima are mainly caused by the loss of directionality in virtual coordinate system generation mitigated in the directional domain. Regained directions are called virtual directions. Network partitioning and routing in special cases of tree networks are some of the properties discussed.

Directional space contains the inherent connectivity information while sense of directions of the node arrangement, which provides a good environment for routing. The proposed routing scheme, Directional Virtual Coordinate Routing (DVCR), outperforms Convex Subspace Routing (CSR) and Logical Coordinate Routing (LCR) with 38.9% and 44.6% average increment in average routability over five network types respectively, in 1.35 average path length to shortest path length ratio with five randomly selected anchors, which is less than 1.5% of nodes.

Effective anchor placement strategy and topology map generation by selecting nearly orthogonal virtual directions are under investigation.

CHAPTER 05

TOPOLOGY PRESERVING MAPS – EXTRACTING LAYOUT MAPS OF WIRELESS SENSOR NETWORKS FROM VIRTUAL COORDINATES

5.1 Introduction

Virtual coordinates provide an economical alternative to geographical coordinates for routing [4] and self-organization of large-scale Wireless Sensor Networks (WSNs). Geographical coordinate-based protocols such as Geographical Routing (GR) require physical location of nodes, which may be obtained by GPS or a localization algorithm. Use of GPS is infeasible or too costly for many applications, while localization using analog measurements such as signal strength and time delay is difficult and prone to errors [91][93]. Signal strength is susceptible to noise, fading, and interferences due to multipath and other devices. The need for accurate power control and signal strength measurements contributes to increased hardware complexity, as well as cost. Routing is carried out using directional information derived from geographic coordinates, and hence concave physical voids in the network degrade the performance of GR schemes. Virtual Coordinate Systems (VCS) characterize each node by a coordinate vector consisting of the shortest path hop distances to a set of anchors [22][23][101]. These anchors are a set of ordinary sensor nodes with no additional capabilities. Coordinates can be obtained using a controlled/organized flooding mechanism [79] initiated by the anchors. VCS is a higher dimensional abstraction of a partial connectivity map of sensors. It has several properties, such as ease of generation, and facilitating connectivity-based routing without the need for geographical information [22][23][34][36][91] that make it attractive for large-scale or resource-starved WSNs. The number of anchors becomes the network's dimensionality in the virtual coordinate space. As the network's connectivity information is embedded in Virtual Coordinates (VCs), the physical voids are transparent in Virtual Space (VS). However, VCs lose the directional information related to node positions. The number of anchors required and their placement for a given network play a crucial role in the performance of VC-based routing. However, identification of the optimal number of anchors and proper anchor placement remain major challenges. Under-deployment of anchors causes identical node coordinates, while their over-deployment and improper placement worsen the local minima problem, resulting in logical voids [23].

Many disadvantages associated with VCS in comparison to geographical coordinate systems are due to the lack of information about physical network topology and layout. As each virtual ordinate propagates radially away from the corresponding anchor, the directional information of a node with respect to the anchor is lost. Thus physical layout information such as physical voids, relative physical direction information of sensor nodes with respect to X-Y positions, and even explicit connectivity information among pairs of nodes are absent in a VCS description. The above information can be revealed if the physical topology map is available. With both, partial connectivity information that is embedded in VCs and position or direction information as in geographical coordinates can be used to overcome the disadvantages in each other's domains. However, physical topological information has to be achieved without inheriting the disadvantages associated with obtaining physical location information or localization.

Obtaining a topology map resembling the physical layout topology of a network from the set of VCs that is based only on hop distances to a small set of anchors has not been possible up to now. In this research, we present the theoretical basis and techniques to obtain Topology Preserving Maps (TPMs) that contain the topology of a network and physical features, including its geographical voids, boundary profiles and relative Cartesian directional information. TPMs overcome many of the disadvantages of VCS compared to geographical coordinate systems but without inheriting its disadvantages, whilst preserving all the advantages of connectivity-based VCs.

Topology preserving maps preserve the neighborhood information. In addition, they are rotated and/or distorted versions of the real physical node maps to account for connectivity information inherent in VCs. Therefore, the topological coordinates provided by the proposed method are a good substitute for geographical coordinates for many applications that depend on connectivity and location information. In fact, the topological coordinates, in conjunction with VCs from which it is derived, have been demonstrated to be better than geographical coordinates for routing due to significantly enhanced routing performance [39]. Boundary node identification for proper anchor placements [22][23][101], backbone identification [126], and recognition of geographic voids are among other examples that can significantly benefit from TPMs. The results presented here demonstrate the ability to determine and visualize the structural characteristics of large-scale WSNs in both 2D and 3D. The ability to do such visualization without the need for analog measurement capability at nodes will be invaluable for future large-scale nanosensor networks [1][6] as well. Even though we focus on WSN context here, the technique is applicable to a broader class of networks.

This research presents a novel technique for obtaining TPMs of 2D and 3D networks from the hop distances to a small set of nodes. Next, Section 5.2 reviews the background. After presenting the theoretical foundation for obtaining TPMs in Section 5.3, we also refine the method to reduce its complexity. A performance evaluation metric for topology maps is presented in Section 5.4. In Section 5.5, we discuss the results of three alternatives for TPM generation, with different computational and communication complexities. Section 5.6 addresses implementation issues. Finally, Section 5.7 discussed the future work and concludes our work.

5.2 Background

We briefly review the related work on use of geographical and virtual coordinates, and localization techniques for generating geographic coordinates and maps for which proposed TPMs are a competitive, economical replacement. The term TPMs has been used in contexts outside sensor networking, such as multi-dimensional data organization. Though some of them are not directly applicable to WSNs, we review the most relevant ones to place the proposed scheme in context.

A. Geographic Routing (GR) vs. Virtual Coordinate Routing (VCR)

In Geographic Routing, the physical location of nodes is used for node addressing as well as for routing. A packet is forwarded in the direction of the destination, and thus GR gets disrupted by geographical voids. Concave voids are especially difficult to overcome. Greedy

Perimeter Stateless Routing (GPSR) [69] makes greedy forwarding decisions until it fails, e.g., due to a geographical void, and attempts to recover by routing around the perimeter of the void. Greedy Other Adaptive Face Routing (GOAFR) [68] is a geometric ad-hoc algorithm combining greedy forwarding and face routing to overcome the local minima issue. Greedy Path Vector Face Routing with Path Vector Exchange GPVFR/PVEX [77] is similar to [68] but it requires the network's planar graph.

VC-based schemes, where each node is characterized by a coordinate vector corresponding to hop distances to a set of anchors, uses a distance measure in VCS to identity the node for packet forwarding. VCR scheme in [101], for example, uses all the perimeter nodes as anchors. When a packet reaches a local minima, an expanding ring search is performed until a closer node is found or TTL expires. In Virtual Coordinate assignment protocol (VCap), the coordinates are defined based on hop distances [22]. At local minima, VCR causes a packet to follow a rule called "local detour". In Logical Coordinate-based Routing (LCR) [23], backtracking is used when greedy forwarding fails at a local minimum. Aligned virtual coordinate system (AVCS) [82] re-evaluates VCs by averaging a node's own coordinate with neighboring coordinates in an attempt to overcome local minima. Convex Subspace Routing [36] overcomes the local minima by using a subset of anchors for routing and by dynamically changing the subset to provide convex distance surfaces for routing. In Axis-Based Virtual Coordinate Assignment Protocol (ABVCap) [113], each node is assigned a 5-tuple VCs corresponding to longitude, latitude, ripple, up, and down. Existing VCR protocols rely mainly on Greedy Forwarding (GF), followed by a backtracking scheme to overcome the local minima issue. Geo-Logical Routing (GLR) [39] is a novel routing scheme that combines the advantages of VCS and TPM proposed in this research to overcome disadvantages of each others' domain, thus impressively outperforming existing VCR schemes as well as GPSR which requires physical coordinates.

B. Localization

We focus on relative localization techniques, as global localization is realizable through relative localization and the actual positions of a subset of nodes or physical anchors. Centralized and distributed algorithms are available for relative localization. Distributed algorithms use Received Signal Strength Indication (RSSI), Radio Hop Count, Time Difference of Arrival, and Angle of Arrival for relative localization. RSSI uses signal strength to estimate the distance between nodes while Radio Hop Count uses hop distance. The latter uses a probabilistic correction equation to approximate the hop distance to real distance [14][115]. Disadvantages of RSSI measurement include sensitivity to terrain [93] and large variations due to fading and interference. The relationship between RSSI and distance is very difficult to predict indoors as well as in complex outdoor environments due to absorption and reflection of signals and propagation characteristics over different terrains. No robust and scalable algorithms are available for localization of nodes deployed on surfaces of complex 3D structures. An RSSI measurement-based distributed algorithm using triangulation for localization of 2D and 3D WSNs is proposed in [127].

Centralized algorithms for localization of 2D networks include Semidefinite Programming (SDP) and MDS-MAP [14][102]. Former algorithm develops geometric constraints between nodes, represents them as linear matrix inequalities (LMIs), and then simply solves for the intersection of the constraints. Unfortunately, not all geometric constraints can be expressed as LMIs that preclude the algorithm’s use in practice. MDS-MAP is Multi-Dimensional Scaling (MDS) based on connectivity information.

The localization scheme in [76] first identifies the boundary nodes of the network, and then, using several rounds of network flooding by boundary nodes, selects a subset of nodes as landmarks. Next, Delaunay triangles are generated based on Voronoi cells formed with landmarks, which again requires network flooding by landmarks. Finally, the network layout is discovered based on the landmarks’ locations. The landmark population has to be dense for the Delaunay triangles to be rigid, thus increasing the communication cost. Moreover, boundary nodes need to be identified accurately without physical information, and an incremental algorithm is required to combine the Delaunay triangles.

Factors that contribute to errors in localization include inaccuracies in distance estimate, the position calculation, and the localization algorithm [91]. How the localization error propagates and accumulates in a network is illustrated in [91] in terms of geographic distribution of the error, correlation, mean error, and probability distribution of the error. This study shows that routability of Geographic Routing (GEAR) falls significantly and the percentage of deliveries to wrong destinations increases as the percentage error in localization increases.

As both VCS and topology maps are generated based on hop distance, they are not affected by fading or signal strengths. Further, they do not rely on analog measurements such as RSSI or time delay, and thus do not have cumulative errors that affect the performance, as networks scale.

C. *Other Topology Preserving Maps*

The TPMs discussed in this research deviates from the localization maps. The relative localization schemes expect the relative distances to be accurate. Thus, given the absolute position of a subset of nodes, global localization is realizable. In contrast, in topology maps, what is important is the topology preservation, not the physical distances. The derived topology should be homeomorphic (topologically isomorphic) to the physical layout of the sensor network; that is, between two topological spaces, there has to be a continuous inverse function. In our case, it is a mapping, which preserves the topological properties of the physical network topology.

In the context of analysis of high-dimensional data, unsupervised learning algorithms have been proposed that use eigen-value decomposition for obtaining a lower dimensional embedding of the data. Here we discuss four such schemes: Multi-Dimensional Scaling (MDS), Local Linear Embedding (LLE), Isomap, and Laplacian Eigenmaps (LE) [16]. None of these methods is designed for, nor is suitable for, resource-starved WSNs.

Multi-Dimensional Scaling (MDS) [102][109] is a commonly used statistical technique in information visualization for exploring similarities or dissimilarities in higher dimensional data from the complete distance matrix (similarity matrix) $D,$ which is defined as the matrix of all the pair-wise distances between points/nodes. $D = [d_{ij}]_{N\times N}$, where N is the number of nodes in the network and d_{ij} is the distance from node i to node j with $d_{ij} = d_{ji}, \geq 0$ and $d_{jj} = 0$. In general d_{ij} can be any distance metric, but there is a possibility for the algorithm to fail if d_{ij} is not the Euclidean distance. Generating D based on hop distances requires all the nodes in a WSN to serve as anchors, an extremely expensive proposition that calculates and stores information about the distances between each pair of nodes. If such information were available at each node, 100% routing could be achieved just by following the ordinate corresponding to the destination, i.e., without the need for the topology map. MDS is therefore not practical or applicable for generating TPMs of WSN. Our novel method, based on Singular Value Decomposition (SVD), generates topology maps of 2D and 3D networks, using a set of M anchors, where $M \ll N, N$ being the number of nodes.

Isomaps [115] are an extension of MDS to geodesic distance-based topology map generation. Again, the geodesic distances are actual distances among nodes, which require expensive error prone distance estimators such as RSSI or Time of Arrival (TOA). Furthermore, if a node has the information of entire network, 100% routability is achievable without the need for a topology map. Moreover, LLE and LE both use an iterative approach to preserve the neighborhood distances, the realization of which is infeasible in an energy-limited WSNs.

All four schemes rely highly on physical distances between all the possible pairs of nodes, and thus require localization approaches. Accuracy of both central and distributed implementations of localization is highly sensitive to channel fading and signal to noise ratio (SNR).

5.3 Topology Preserving Maps for 2D and 3D WSNs

A novel technique for obtaining a representation that closely resembles the physical layout of a sensor network from its VC set is presented next. The objective is to characterize each node with a (x, y) coordinate pair, or (x, y, z) in case of 3D WSNs, that results in a TPM that is homeomorphic to the network's physical layout, and preserves information about node connectivity, physical layout, and physical voids.

Subsection A develops the technique by starting with the VCs of all the nodes to obtain a TPM. Subsection B discusses the extension of TPMs to 3D networks. A significantly more efficient version of the technique that uses information of only a small subset of nodes to evaluate the transformation matrix is presented in Subsection C. Finally, Subsection D proposes a method of calculating node's Cartesian coordinates with lower computational complexity. Notations used in the text are summarized in Table 5.1.

Table 5.1
Notations Used In Chapter 5

Notation	Description
N	Set of nodes
N=|N |	Number of network nodes
n_i ∈N	Node i
A ⊂N	Set of anchor nodes
M =|A | $(M << N)$	Number of anchors
A_i ϵA , $i = 1{:}\,M$	i^{th}anchor
$h_{n_i n_j}$	Minimum hop distance between nodes n_i, n_j
$P_{N\times M}$	Virtual coordinate matrix of the entire network
$P_{(i)} = [h_{n_i A_1}, \dots, h_{n_i A_M}]$	VCs of Node n_i
$Q_{R\times M}$; $R << N$	VCs of a subset of nodes
$P_{SVD}{}^{(i)}$	i^{th} principle component of P
$[X_T, Y_T]$, $[X_T, Y_T, Z_T]$	Topological coordinate matrix of a 2-D network Topological coordinate matrix of a 3-D network
$[X_T, Y_T]_{(i)} = [X_{T,i}, Y_{T,i}]$ $[X_T, Y_T, Z_T]_{(i)} = [X_{T,i}, Y_{T,i}, Z_{T,i}]$	Topological coordinates of node n_i of a 2-D network Topological coordinates of node n_i of a 3-D network
$[X_p, Y_p]$, $[X_p, Y_p, Z_p]$	Physical coordinates of a 2-D network Physical coordinates of a 3-D network

A. 2D topology preserving maps from VCs

Consider a 2D sensor network with N nodes and M anchors. Thus, each node is characterized by a VC vector of length M. Let P be the $N \times M$ matrix containing the VCs of all the nodes, e.g., the ith row corresponds to the M-long VC vector of the ith node, and jth column corresponds to the virtual ordinate of all the nodes in the network with respect to jth anchor. Therefore,

$$P = [h_{n_iA_j}] = \begin{bmatrix} h_{n_1A_1} & h_{n_1A_2} & \cdots & h_{n_1A_M} \\ h_{n_2A_1} & h_{n_2A_2} & \cdots & h_{n_2A_M} \\ & & \cdots & \\ h_{n_iA_1} & h_{n_iA_2} & \cdots & h_{n_iA_M} \\ & & \cdots & \\ h_{n_NA_1} & h_{n_NA_2} & \cdots & h_{n_NA_M} \end{bmatrix}$$

where $h_{n_iA_j}$ is the hop distance from node n_i to anchor A_j. For sensor network applications, it is generally desirable to have only a small subset of nodes as anchors, i.e., $M \ll N$. The 2D network has an M-dimensional representation under the VC transformation. The main goal thus is to extract the 2D representation of the network from this M-D space.

Singular Value Decomposition (SVD) [72] of P is denoted as

$$P = U.S.V^T \qquad (1)$$

where U, S and V are $N \times N$, $N \times M$, and $M \times M$ matrices respectively. U and V are unitary matrices, i.e., $U^TU = I_{N\times N}$ and $V^TV = I_{M\times M}$. SVD extracts and packages the salient characteristics of the dataset P providing an optimal basis for P. Moreover V is an optimal basis of P^T, i.e., V spans R^M.

Let us consider the Principle Components (PCs) of P

$$P_{SVD} = U \cdot S \qquad (2)$$

P_{SVD} is a $N \times M$ matrix that describes each node with a new set of M-length coordinate vectors. It gives the coordinates for the data in P under the new basis V. As S is a diagonal matrix with diagonal elements being the singular values of P arranged in their descending order, elements in S provide unequal weights on columns of U. Using the unitary property of V, it is also the projection of P on to V [72], i.e.,

$$P_{SVD} = P \cdot V \tag{3}$$

The columns of P_{SVD}, i.e., the PC values of the VC set are arranged in the descending order of information about the original coordinate set. The 1st PC captures the highest variance of the data set, and each succeeding component has the highest variance possible under the constraint that it be orthogonal to the preceding components. 1st PC is considered indispensable in most SVD based techniques, as it contains the most important and discriminating information. However, in case of VCs, discarding 1st PC makes it possible to recover the layout of the network. This is due to 1st PC inheriting the resultant convex shape associated with the dominant concentrically increasing property of individual VCs. Appendix A shows how the convex nature of individual VCs causes the convexity of 1st PC.

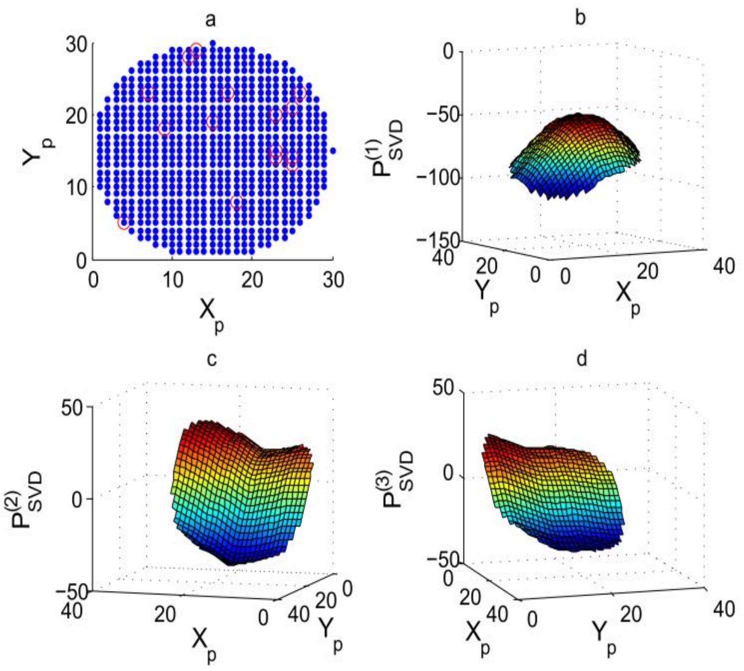


Figure 5.1. a) Circular network of 707 nodes with 15 anchors; b) - d) First three PCs $P^{(1)}_{SVD}$, $P^{(2)}_{SVD}$,and $P^{(3)}_{SVD}$ plotted against the phisical positions. Randomly selected anchors are marked in red cirlcles.

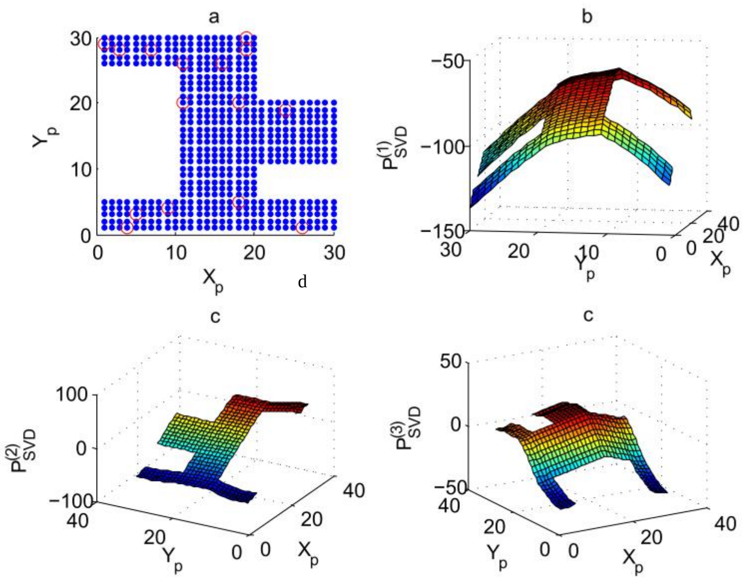


Figure 5.2. a) Odd shaped network with 550 nodes with 15 anchors; b) - d) First three PCs $P_{SVD}^{(1)}$, $P_{SVD}^{(2)}$,and $P_{SVD}^{(3)}$ plotted against the phisical positions. Randomly selected anchors are marked in red cirlcles.

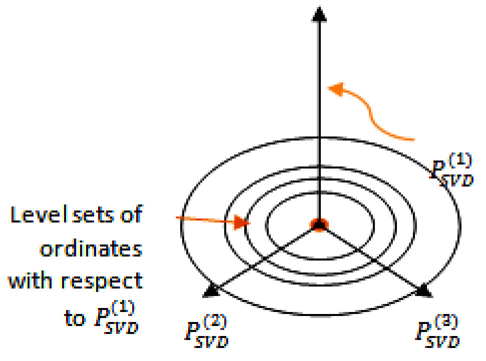


Figure 5.3. Nature of principal component directions derived from virtual coordinates.

Figures 1 and 2 show for two different networks, the physical layout and the first three PCs, i.e., columns of P_{SVD} given by (3), plotted against the corresponding physical positions of the nodes. The initial triplet of SVD coordinates are the dominating coordinates, while the rest, as observed in [37], are similar to Fourier basis vectors with far less significant amplitudes.

The set of VCs have the connectivity information embedded in it, though it has no directional information. All the nodes that are h hops away from the jth anchor have h as the jth ordinate. Each ordinate propagates as a concentric circle centered at the corresponding anchor, while the angular information is completely lost. Thus we can expect the most significant ordinate based on SVD, i.e., first column of P_{SVD} to be a radially distributed vector, which will not provide information to distinctly identify different nodes (see Figure 5.1 (b)) and Figure 5.2 (b)). As SVD provides an orthonormal basis, 2nd and 3rd ordinates are orthogonal to 1st ordinate while being perpendicular to each other as illustrated in Figure 5.3. This leads to the fact that the second and third columns of P_{SVD} provide a set of two-dimensional Cartesian coordinates for node positions, unencumbered by the dominant radial information in VCs, which was captured by the first column. Thus instead of the M coordinates of a row of P_{SVD} to characterize a node, the second and third columns can be used as Cartesian coordinates, i.e.,

$$[X_T, Y_T] = [{P_{SVD}}^{(2)}, {P_{SVD}}^{(3)}] \tag{4}$$

where ${P_{SVD}}^{(j)}$ is the jth column of P_{SVD}. $[X_T, Y_T]$ is the Cartesian coordinate matrix of the entire node set, i.e., its ith row, $[X_T, Y_T]_{(i)}$, is used as the Cartesian coordinates of ith node. By construction of SVD coordinates, the second and third basis vectors form an orthogonal Cartesian plane for the network, and we assert that corresponding ordinates give us an approximate set of Cartesian coordinates for the TPM. The most fascinating fact is that these Cartesian coordinates are estimated without having any kind of physical directional or positioning information beyond the radial information (hop distance) with respect to the anchors. Thus obtained, the TPM reflects the original topological characteristics of the network. One can even identify features such as physical voids that were not apparent in the VC-based description.

We now illustrate the procedure using as an example the T-shaped network of 10 nodes shown in Figure 5.4 (a). Physical coordinates $[\mathrm{X_P}, \mathrm{Y_P}]$ and virtual coordinate matrix P with respect to anchors A, C, E, and J are given in Table 5.2. SVD evaluation of P as in (1) provides V as,

$$V = \begin{pmatrix} -0.55 & -0.34 & -0.71 & -0.30 \\ -0.34 & -0.25 & 0.00 & 0.91 \\ -0.55 & -0.34 & 0.71 & -0.30 \\ -0.54 & 0.84 & 0.00 & 0.03 \end{pmatrix}$$

P_{SVD} can now be evaluated using (3), and thus topological coordinates of nodes are given by (4). $[{P_{SVD}}^{(2)}, {P_{SVD}}^{(3)}]$ is tabulated in Table 5.2 and plotted in Figure 5.4 (b).

B. 3D topology preserving map from VCs

Sensor networks may be deployed within 3D volumes, 3D surfaces, or a combination of those. Here we consider sensors deployed on a 3D surface, which may even wrap around,

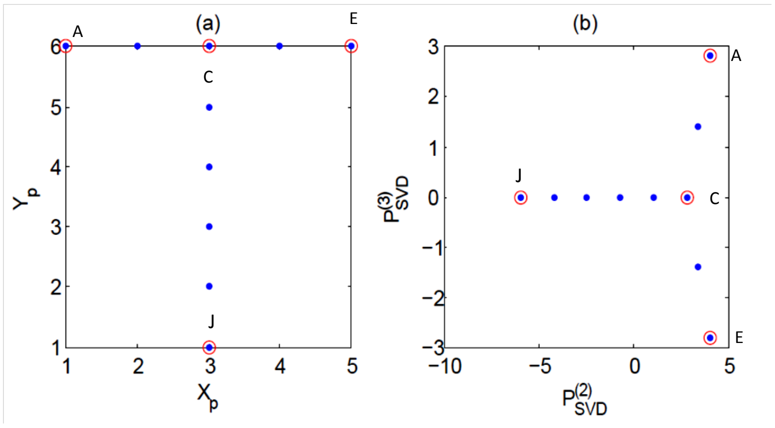


Figure 5.4. a) Physical map of a T-shaped example network; b) Topology map of the network in a).

Table 5.2
Physical coordinates, Virtual Coordinates and Topological coordinates for the network in Figure 5.4(a)

ID	X_P	Y_P	P				P_{SVD}			
			A	C	E	J	$P_{SVD}^{(1)}$	$P_{SVD}^{(2)}$	$P_{SVD}^{(3)}$	$P_{SVD}^{(4)}$
A	1	6	0	2	4	7	-6.62	4.06	2.83	0.82
B	2	6	1	1	3	6	-5.74	3.46	1.41	-0.12
C	3	6	2	0	2	5	-4.87	2.87	0.00	-1.05
D	4	6	3	1	1	6	-5.74	3.46	-1.41	-0.12
E	5	6	4	2	0	7	-6.62	4.06	-2.83	0.82
F	3	5	3	1	3	4	-5.76	1.10	0.00	-0.76
G	3	4	4	2	4	3	-6.66	-0.66	0.00	-0.47
H	3	3	5	3	5	2	-7.55	-2.43	0.00	-0.19
I	3	2	6	4	6	1	-8.45	-4.19	0.00	0.10
J	3	1	7	5	7	0	-9.35	-5.96	0.00	0.39

thus affecting virtual coordinate propagation in complex ways. Consider the uniform cylindrical surface shown in Figure 5.5 (a) on which 900 nodes are deployed. Figures 5.6 (a)-(d) show the plots of the first four PCs for each node in the network. They are denoted by ${P_{SVD}}^{(1)}, {P_{SVD}}^{(2)}, {P_{SVD}}^{(3)}$, and ${P_{SVD}}^{(4)}$ respectively. As SVD provides an orthonormal basis, 2nd, 3rd, and 4th PCs are orthogonal to 1st ordinate while being perpendicular to each other. Fascinatingly, the salient feature of the VCS, i.e., the radial propagation of coordinates is captured by the 1st PC as observed in 2D case. Thus removing it from further

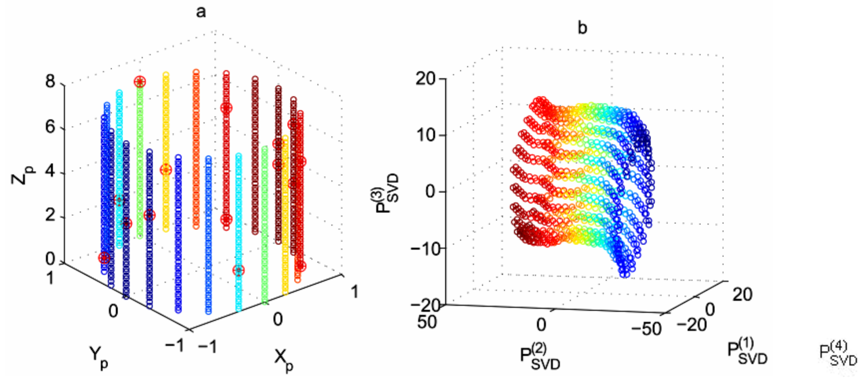


Figure 5.5. a) A network on a cylindrical surface (900 nodes) Randomly selected 20 anchors are marked in red cirlcles; b) topology map of a)

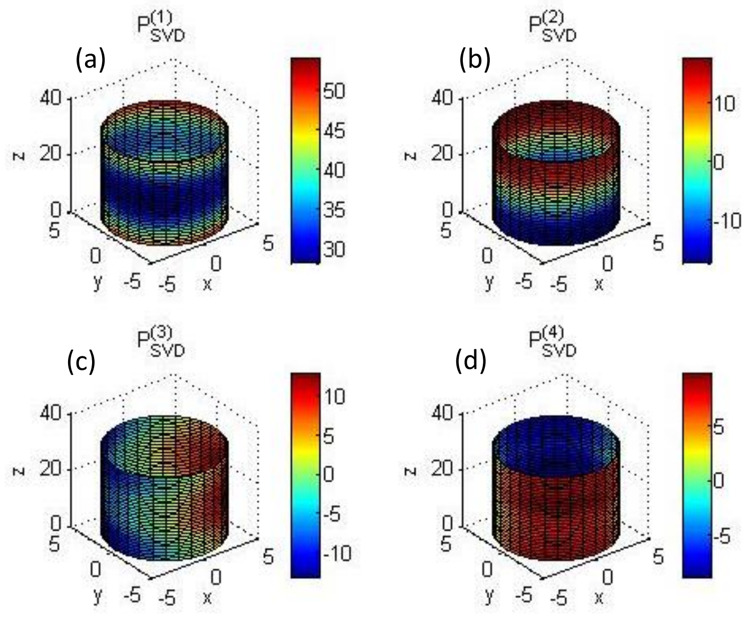


Figure 5.6. First four PCs a) $P_{SVD}^{(1)}$; b) $P_{SVD}^{(2)}$; c) $P_{SVD}^{(3)}$; and d) $P_{SVD}^{(4)}$ of a cylindrical network plotted as a color map on the surface of the network.

Table 5.3
Computational Complexity and Memory Usage Comparison ($N \gg M$)

Method	**Full SVD implementation with *P***	**EVD method of estimating *V* of *P***
# **Computations**	$(4N^2M + 8M^2 + 9M^3)$ [58]	*Upper bounded by* $(4M^2N +$ $8M3$ [58]
Memory usage	$(M \times M) + (N \times N) + (N \times M)$	*Upper bounded by* $(M \times M) + (1 \times M)$

consideration allows us to uncover linear patterns embedded in the VC set. As seen in Figure 5.6 (b), the second PC varies along the height of the cylinder, thus it can be used to obtain the Z coordinate for the topology map. More interestingly, 3rd and 4th PCs, which can be taken as X and Y coordinates, directionally distribute in such a way that they are orthogonal to each other while being normal to 2nd PC. The resulting TPM is illustrated in Figure 5.5 (b). TPM generation on 3D surfaces can thus be done by ignoring the first PC and by taking 2nd, 3rd, and 4th columns of P_{SVD} to provide a set of three-dimensional (3D) Cartesian coordinates. Therefore, topological coordinates of node n_{i} can be written as

$$[X_T, Y_T, Z_T]_{(i)} = [{P_{SVD}}^{(2)}, {P_{SVD}}^{(3)}, {P_{SVD}}^{(4)}]_{(i)} = [P_{(i)}.V^{(2)},\ P_{(i)}.V^{(3)}, P_{(i)}.V^{(4)}] \tag{5}$$

where $[{P_{SVD}}^{(j)}]_{(i)}$ is jth PCs of node n_i. Note that the above result holds for 3D volumes as well. The first PC in that case will propagate radially outward from the center of the volume, as opposed to from the center of the area in case of 2D networks.

C. *Generation of Cartesian coordinate set using VCs of a subset of nodes*

Cartesian coordinates for 2D TPM are obtained by multiplying the node's VC by V as in (3) and (4) (and as in (5) for 3D TPMs). V is based on P, the $N \times M$ matrix that consists of VCs of all the nodes. In sensor networks, it is crucial to reduce communication and computation overheads. This section presents a process to generate the transformation matrix V with only a small subset of rows of P, thus significantly reducing the computation overhead.

Let Q be the sub-matrix of P corresponding to an appropriately selected set of R nodes (rows). Let the SVD of Q be

$$Q = U_Q \cdot S_Q \cdot {V_Q}^T \tag{6}$$

Q is $R \times M$, where M is the number of anchors. U_Q, S_Q and V_Q are $R \times R$, $R \times M$ and $M \times M$ matrices respectively. If Q is selected appropriately, V_Q can serve as a substitute, or at a minimum a good approximation, for V for TPM generation. Note that V_Q has the same size as V in (1) and is also unitary. Following the same procedure as earlier, we use

$$P_{SVD} = P \cdot V_Q \tag{7}$$

The Cartesian coordinates for TPMs of 2D and 3D networks can be written as

$$[X_T, Y_T] = [{P_{SVD}}^{(2)}, {P_{SVD}}^{(3)}]$$

$$[X_T, Y_T, Z_T] = [{P_{SVD}}^{(2)}, {P_{SVD}}^{(3)}, {P_{SVD}}^{(4)}] \quad (8)$$

respectively.

While there are many possible ways to select the subset of nodes, we use the following two simple options in this research:

1. Use the set of M anchor nodes ($Q = Q_A$)
2. Use a set of R randomly selected nodes ($Q = Q_R$)

As $N >> M, R$, significant savings in overhead can be achieved, and results presented later demonstrate that the impact on accuracy is negligible.

V is a basis of R^M. V_Q is also a basis for R^M even though it is based on subset of coordinates. Therefore, we can write

$$V = V_Q \cdot \Phi$$

where Φ is a rotation matrix. If the selected subset of coordinates is a good representation of the entire P, similar TPMs can be generated, as demonstrated in Section 5.5, with a significantly low amount of computational, memory, and communication complexities.

D. *A computationally efficient implementation*

Computational power and memory available at a sensor node is limited. Conventional SVD calculation of $P_{N\times M}, (N \gg M)$, which involves computing U, S, and V, has approximately $(4N^2M + 8NM^2 + 9M^3)$ operations [58]. Also the memory requirement is approximately the sizes of V, U and S that is $(M \times M + N \times N + N \times M)$. In this section, we present a technique for further enhancing the efficiency of the computation necessary for 2D and 3D TPM generation. Note that U is a byproduct of SVD and is not necessary for topology map computation. The Eigen-value decomposition (EVD) based approach [72] to evaluate matrix

V not only allows us to implement the TPM generation in a distributed manner, but also completely avoids generating matrix U, thus reducing the computational complexity and memory requirement compared to those for SVD. From (1), (3), and (4), ${P_{SVD}}^{(j)}$, the jth column of P_{SVD} is given by

$$ {P_{SVD}}^{(j)} = [h_{n_iA_1}, \dots, h_{n_iA_M}] \cdot V^{(j)};\ i = 1{:}\,N \tag{9} $$

$[h_{n_iA_1}, \dots, h_{n_iA_M}]$ is the coordinate vector of the node i. Also $V^{(j)}$ is the jth basis vector/column of V. $j = 2{,}3$ for 2D networks, while $j = 2{,}3{,}4$ for 3D networks. Thus, $V^{(2)}$ and $V^{(3)}$ are sufficient to evaluate 2D Cartesian coordinates $[X_{T,i}, Y_{T,i}]$ of node i. 3D networks require $V^{(2)}$, $V^{(3)}$ and $V^{(4)}$. Define C as

$$ \begin{gathered} C = P^T.P = V.S^2.V^T \\ C.V = V.S^2 \end{gathered} \tag{10} $$

C is an $M \times M$ symmetric matrix. This is an eigenvalue problem [72]. Therefore, let us solve,

$$ C.v = \lambda.v. \tag{11} $$

v is an eigenvector of C that is a column of V. Eigen values λ can be found by solving

$$ |C - \lambda.I| = 0 \tag{12} $$

The eigenvectors, corresponding to second and third largest eigenvalues, provides the second and third columns of V. Now $[X_{T,i}, Y_{T,i}]$ ($[X_{T,i}, Y_{T,i}, Z_{T,i}]$ for 3D case) can be evaluated locally without calculating the entire V matrix. Also $U_{N\times N}$ is not evaluated at all, which reduces the memory consumption significantly. Therefore the memory consumption is upper bounded by $(M \times M) + (1 \times M)$. Number of computations required for this method of calculating V is upper bounded by $(4M^2N + 8M^3)$[58], which are the computations associated with the calculation of entire V and S. Since $N \gg M$, this method is significantly less complex compared to the full SVD implementation (see Table 5.3). For example, if the number of anchors in the network is set to $M \leq 0.01N$, which is reasonable based on our experience, the upper bound of computations required with this method is only 0.99% of the computations required for a full SVD-based calculation with (3) and (4), indicating a significant reduction in complexity.

5.4 A Metric for Evaluating 2D Topology Preservation

Evaluating the degree of topology preservation of the sensor node maps generated is essential for investigating the effectiveness of the proposed scheme. While visual inspection can provide preliminary evidence of its effectiveness, a formal metric is needed for quantifying the accuracy. A quantitative parameter to express the error provides a framework to compare and improve different mapping techniques. An effective metric should be able to capture and quantify the failures to preserve the topology of the real node map and the neighborhoods. Such a metric is not currently available. Here we develop a metric that can be used for this purpose.

A method based on the coloring of nodes is used in [98] to show whether a neighborhood has been altered in the topology map. In [98] and [115], error is quantified as the difference of the positions in the actual physical map and the topology map, and the residual variance, respectively. The focus of our research is TPMs based on hop distances. The requirement is that the map from calculated $[X_T, Y_T]$ set is homeomorphic to the physical layout and preserves information about node connectivity, physical layout, and physical voids. Thus the actual physical distance is not of significance, and the metrics in [98] and [115] are not appropriate.

Consider as an example, a 1D network with six nodes numbered 1 to 6, as in Figure 5.7 (a). Figures 7 (b) and (c) show two derived maps that need to be evaluated. If all the nodes are in same order as in initial topology then the Topology Preservation Error must be 0%. Node 3 in Figure 5.7 (b) has flipped two node positions. The error metric should identify the number of out-of-order nodes as well as the degree of the error/node flips (one node and two node positions respectively for Figure 5.7 (b)).

Consider a 1D network with N nodes and define an indicator function $I_{i,j}$ where

$$I_{i,j} = \begin{cases} 1 & i \text{ and } j \text{ are out of order compared to original placement} \\ 0 & i \text{ and } j \text{ are in same order as original placement or } i = j \end{cases}$$

$i, j = 1 \text{ to } N$

Then, number of out of order pairs = $\sum_{\text{all i,j}} (\mathrm{I}_{\text{i,j}})$

The total number of possible pairs in an N node network is P_2^N. We define the following metric:

$$\text{Topology Preservation Error } = \mathrm{E}_{\mathrm{TP}} = \frac{\sum_{\text{all i,j}}(\mathrm{I}_{\text{i,j}})}{\mathrm{P}_2^{\mathrm{N}}} \tag{13}$$

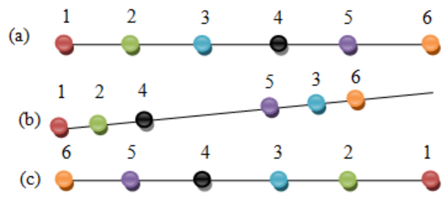


Figure 5.7. (a) A network, (b) a topology map of (a) with a node flip, and (c) a topology map of (a) with 180^0 rotation.

For the network in Figure 5.7 (b), N = 6 and

$$\mathrm{E_{TP}} = (\mathrm{I_{3,4}} + \mathrm{I_{3,5}} + \mathrm{I_{4,3}} + \mathrm{I_{5,3}})/\mathrm{P}_2^6 \times 100\% = (\mathrm{I_{3,4}} + \mathrm{I_{3,5}})/\mathrm{C}_2^6 \times 100\% = 13.3\%$$

Nodes 1 and 2 are in the right position compared to the rest while node 3 is shifted by two positions. Moreover, nodes 4 and 5 flipped their positions by one. Therefore, there are four total node flips, and $\mathrm{E_{TP}}$is 13.3%. A TPM is invariant to rotations. Thus, for Figure 5.7 (c), where nodes are just reversed, $\mathrm{E_{TP}}$ has to be zero. To handle such cases, the two lines being compared need to be adjusted for any rotations.

To extend $\mathrm{E_{TP}}$ equation to 2D topologies, we evaluate the 2D topology by considering all contiguous line segments in two orthogonal directions (say $\vec{H}$ and $\vec{V}$) of the physical map.

Let us assume there are α lines in $\vec{H}$ direction and β lines in $\vec{V}$ direction in the network, then

$$\text{Topology Preservation Error in } \vec{\mathrm{H}} \text{ direction } \% = \mathrm{E}_{\mathrm{TP}|\vec{\mathrm{H}}} = \frac{\sum_\alpha \sum_{\text{all i,j}}(\mathrm{I_{i,j}})}{\sum_\alpha \mathrm{P}_2^{\mathrm{N_h}}} \qquad (14)$$

i, j are nodes in each horizontal line and each line has N_h nodes. Similarly, error in vertical direction is evaluated as

$$\text{Vertical neighborhood preservation error } \% = \mathrm{E}_{\mathrm{TP}|\vec{\mathrm{V}}} = \frac{\sum_\beta \sum_{\text{all i,j}}(\mathrm{I_{i,j}})}{\sum_\beta \mathrm{P}_2^{\mathrm{N_v}}} \qquad (15)$$

i, j are nodes in each vertical line and each has N_v nodes. The overall Topology Preservation Error, $\mathrm{E_{TP}}$, can be defined as:

$$\mathrm{E_{TP}} = \frac{\sum_\beta \sum_{\text{all i,j}}(\mathrm{I_{i,j}}) + \sum_\alpha \sum_{\text{all i,j}}(\mathrm{I_{i,j}})}{\sum_\alpha \mathrm{P}_2^{\mathrm{N_h}} + \sum_\beta \mathrm{P}_2^{\mathrm{N_v}}} \qquad (16)$$

5.5 Results

The performance of the proposed TPM generation is evaluated next using three 2D examples and two 3D examples representative of a variety of networks. MATLAB® 2009b was used for the computations.

A. TPMs of 2D networks

Figures identified as (a) in Figures 5.8-10 show the physical maps of the three 2D networks considered: an odd-shaped network with 550 nodes (Figure 5.8 (a)), a 496-node circular shaped network with three physical voids/holes (Figure 5.9 (a)), and a network of 343 nodes on walls of a building (Figure 5.10 (a)). The communication range of a node in all three networks is unity. Detailed specifications of these networks are available at [21]. Topology maps are generated based on methods summarized in Table 5.4.

Unless otherwise indicated, the results shown correspond to fifteen randomly placed anchors in each of the networks. Building network in Figure 5.10(a) has just three anchors. Figures 5.8-10 (b) show TPMs constructed based on (4), using entire VC set of each network. Therefore TPMs in Figures 5.8-10 (b) use input data matrices of sizes 550×15, 496×15 and 343×3 respectively (Case 1, Table 5.4). Figures 5.8-10 (c) are the topology maps created using only the anchors' coordinate set, that is, using (7) and (8) based on the input data matrices Q_A of size 15×15, 15x15, and 3×3 respectively (Case 2, Table 5.4). Topology maps in Figures 5.8-10 (d) are created based on coordinates of 10 randomly selected nodes; the corresponding sizes of Q_R are 10×15, 10×15 and 10×3 respectively (Case 3, Table 5.4). For the purpose of comparison, Figures 5.8-10 (e) consider all the nodes in the network to be anchors, corresponding to P of sizes 550×550, 496×496, and 343×343

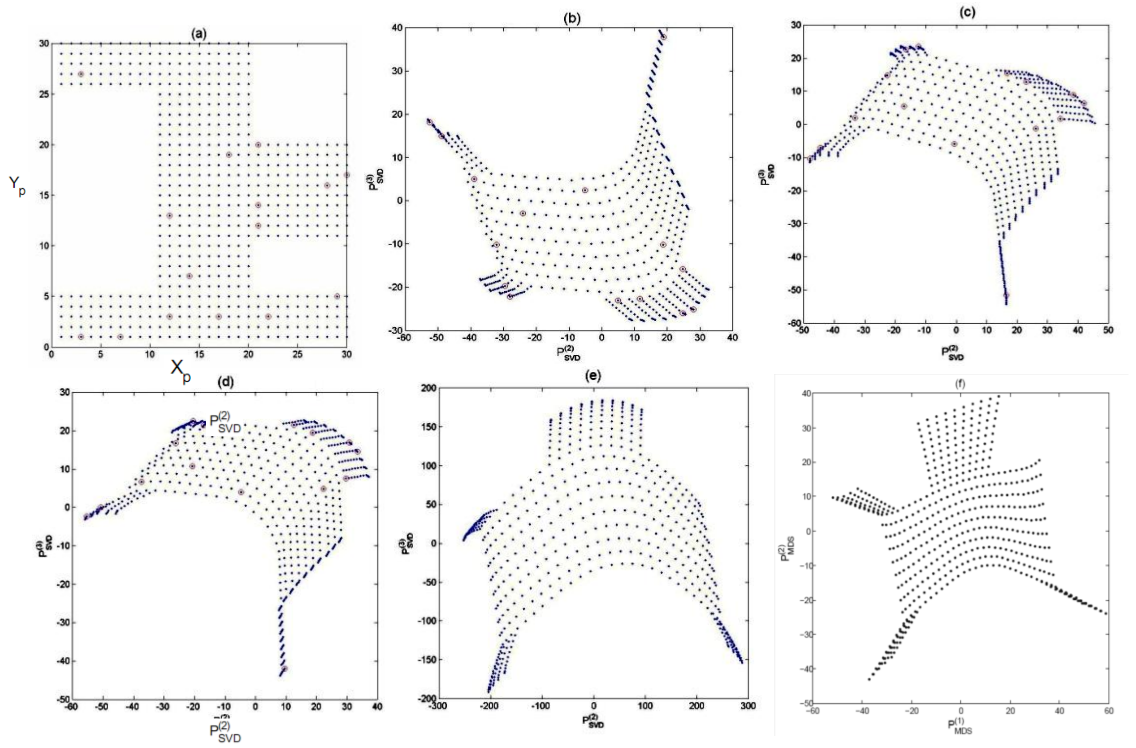


Figure 5.8. a) Odd shaped network with 550 nodes and 10 random anchors; $[X_T, Y_T]$ is generated based on b) Case 1: entire VC set, c) Case 2: anchors' coordinate set, d) Case 3: randomly selected nodes' coordinate set, and e) Case 4: coordinate set with all the nodes are anchors, f) MDS

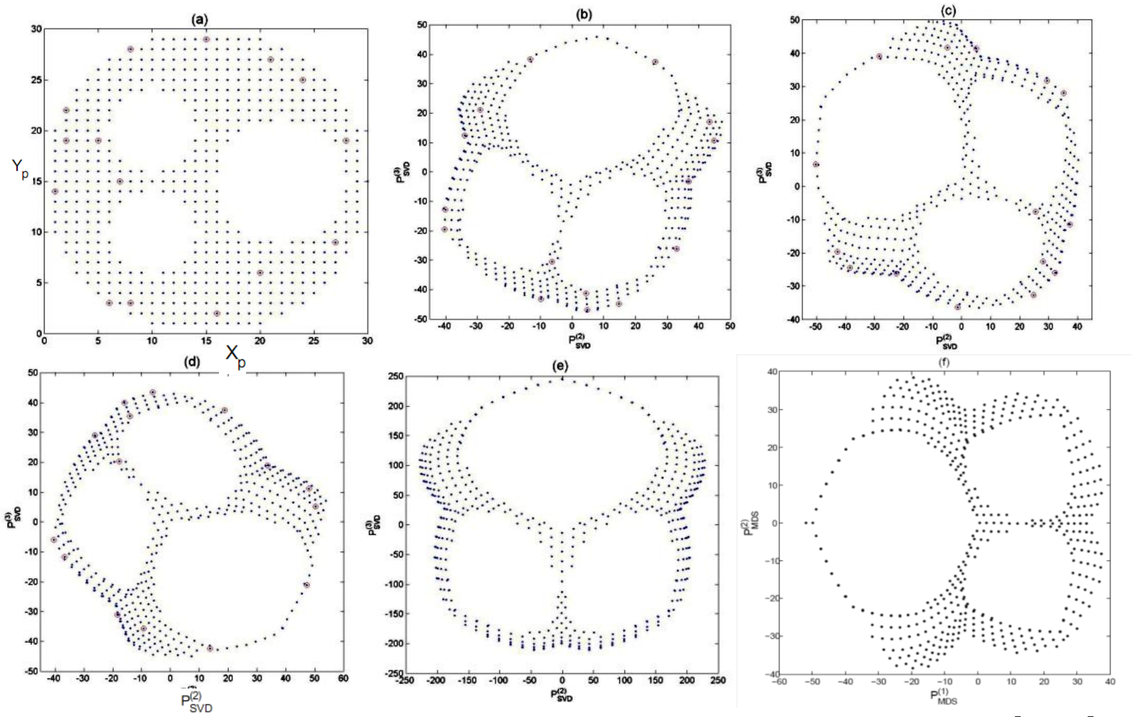


Figure 5.9. a) Circular network with 3 physical voids with 496 nodes and 10 random anchors; $[X_T, Y_T]$ is generated based on b) Case 1- entire VC set, c) Case 2: anchors' VC set d) Case 3: randomly selected nodes' VC set and e) Case 4: VC set with all the nodes are anchors f) MDS

Table 5.4
Four Different Topology Map Generation Approaches for WSNs of N nodes and M Anchors

Case	**Description**	**Size of input data matrix**
1	$[X_T, Y_T]$ from P	$N \times M$
2	$[X_T, Y_T]$ from Q_A	$M \times M$
3	$[X_T, Y_T]$ from Q_R	$R \times M$
4	$[X_T, Y_T]$ from $Q_A\|_{\|A\|=M=N}$	$N \times N$

Table 5.5
ETP For Topology Maps In Figure 5.8-10.

	E_{TP} (%)			
Figure	**Case 1**	**Case 2**	**Case 3**	**Case 4**
Figure 5.8 (a)	1.6777	1.5894	1.5011	1.4570
Figure 5.9 (a)	0.3605	1.0698	0.4884	0
Figure 5.10 (a)	0.1315	0.1315	0.1315	0.0376

respectively for the three networks (Case 4, Table 5.4). Case 3 is more efficient in terms of memory consumption and computational complexity. Finally, we compare our results with those of MDS-MAP method proposed in [102] shown in Figures 5.8-10 (f). For MDS, data from the complete distance matrix D, which is defined as the matrix of all the pair-wise distances between points/nodes, is required. $D = [d_{ij}]_{N\times N}$, where N is the number of nodes in the network and d_{ij} is the distance from node i to node j. As proposed in [102], d_{ij} can be either geodesic distance or hop distance between i and j. For this comparison, we use hop distances to generate MDS-MAP, thus VCS requires all the nodes to be anchors. A TPM for the circular network of Figure 5.1 (a) can be found in [37].

Figures 8-10 clearly demonstrate the effectiveness of the proposed method in generating TPMs. Starting just with VCs, without explicit knowledge of geographical information, the generated topology maps have captured significant features, such as the physical voids and boundaries of the original network. A key observation we can draw from Figures 5.8-10 is that the constructed topology maps are nonlinearly scaled and rotated compared to the

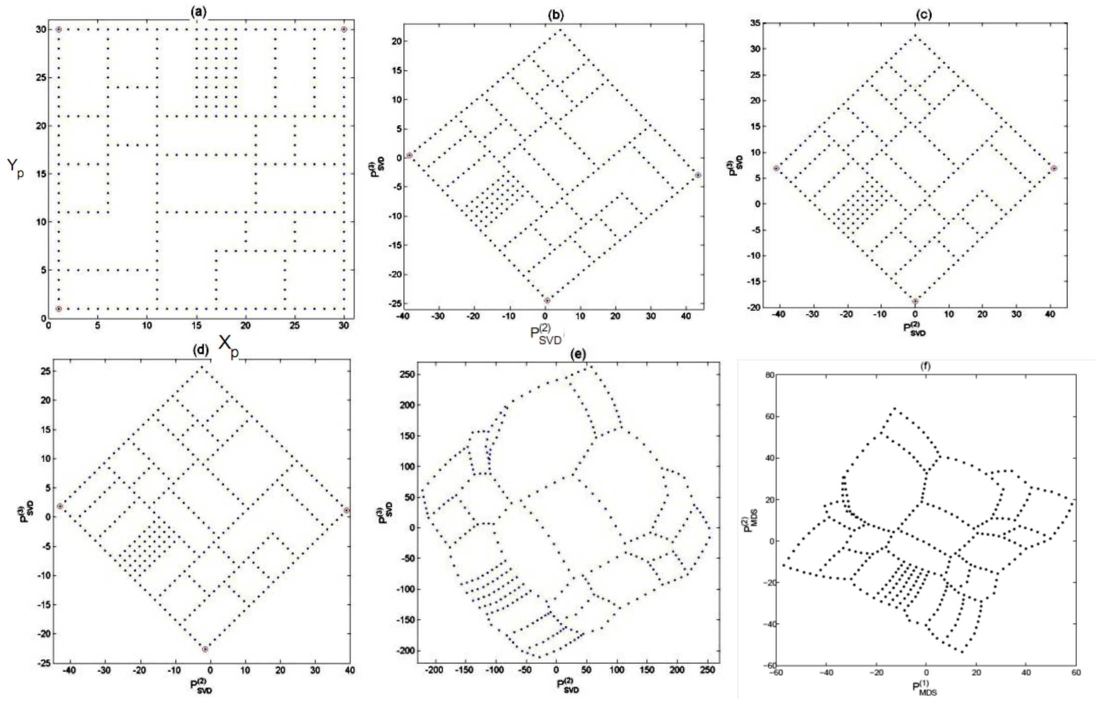


Figure 5.10. a) Network in a building with 343 nodes and 3 anchors; $[X_T, Y_T]$ is generated based on b) Case 1- entire VC set, c) Case 2- anchors' coordinate set d) Case 3- randomly selected nodes' coordinate set and e) Case 4- coordinate set with all the nodes are anchors f) MDS

actual network map. Yet, the original and constructed maps are topologically isomorphic. In contrast to previous cases, the topology maps of Figure 5.10 (b), (c), and (d) are simply rotated and linear scaled versions of the original. In this network, we used only three anchors that were manually selected. The physical voids present in Figure 5.10 (a) are well preserved. Even though the map in Figure 5.10 (e) was obtained using all the nodes as anchors, its shape is deformed compared to Figure 5.10 (b)-(d); however, in terms of neighborhood preservation, Figure 5.10 (e) is better. For example, one of the L-shaped rooms in the building network (Figure 5.10 (a)) is distorted in the topology maps of Figures 5.10 (b)-(d). In Figure 5.10(e) the L-shape is deformed, but the neighborhood of that L-shaped room is preserved. Case 4 is presented here only for the purpose of comparison. If all nodes are anchors, a very expensive proposition for WSNs, the need for TPMs does not arise for many applications, such as routing. Obtaining MDS-MAPs shown in Figure 5.8-10 (f) require the hop distances from each node to every other node. A major disadvantage is that it is not feasible to implement MDS in a distributed manner due to the extremely high communication cost associated with generating the distance matrix, which consists of the distance between every pair of nodes. In fact, if such information is available at each node, it can be used to achieve 100% routability without the need to generate TPMs.

Moreover, from topology maps in Figure 5.10, we can draw the valuable conclusion that good anchor placement can significantly reduce the number of anchors required for topology map generation. It is topology-preserving to a very high degree as intended. It can be clearly seen that Figure 5.10 (b)-(d) is very close to the original map, indicating that an appropriately placed small number of anchors can produce very accurate topology maps. This reveals the possibility of obtaining even physically representative layout maps with the appropriate selection of anchor nodes for a certain class of networks. Furthermore, our later research in [86] demonstrates that TPMs can be obtained even under large communication ranges.

E_{TP} (in (16)) for the different topology maps is presented in Table 5.5. Note that the error in all the cases is less than 2%. The best performance in terms of E_{TP} was achieved when all the nodes were selected as anchors for the networks in Figure 5.8-10. Case 4 (Table 5.5) acts as a lower bound for the E_{TP} for each network.

Even though SVD-based TPM generation started with a VC set where there is no directionality information, the resultant topology map has directional information that can be used for routing in many ways. For example, to avoid logical voids in VC routing, organized random routing and geographic routing in virtual space may be used [39]. Moreover, as we discussed in Section 5.2, there are other VCSs [82][113], which are derivatives of hop distance based VCS used here. Results of applying the proposed TPM generation method to two such systems, ABVCap [113] and Aligned VCS [82], are presented in Appendix B.

B. TPMs of 3D networks

In this section we present the 3D TPMs generated using the proposed scheme. Two example networks deployed on 3D are considered as shown in Figure 5.11 (a) and Figure 5.12 (a):

a. T-joint (3D surface network): A pipeline structure joining two perpendicular cylinders in a T-joint. There is a hole in one of the cylinders (see Figure 5.11 (a)). Each cylinder has a unit radius and a height of 7 units. It is covered with 1642 nodes, each with a communication range of 0.4. 50 randomly selected nodes (i.e., 3% of the nodes) served as anchors.

b. 3D volume network: It consists of a solid sphere of radius 4 with a cylindrical hole, mounted on two perpendicularly crossed cylinders with height 10 and radius 2 (see Figure 5.12 (a)). The entire volume is filled with 3827 nodes, each with a communication range of 0.5. 50 randomly selected nodes (i.e., less than 1.5% of the nodes) served as anchors.

TPMs of the corresponding physical topologies are shown in Figure 5.11 and 12. The results clearly demonstrate the effectiveness of the TPM generation for sensor networks deployed on 3D surfaces and in 3D volumes. Moreover, it indicates that the maps can be obtained using a very small number of random nodes serving as anchors.

5.6 Realizations, Applications and Extensions

The major contribution of this research is the technique described and evaluated above for the generation of TPMs. Subsection A briefly addresses the realization details of the TPM algorithm in a static WSN. Routing is a crucial operation in WSNs. Subsection B discusses how WSN routing can benefit from TPMs. Subsection C discusses the impact of network dynamics on TPMs.

A. Off-network and In-network realization of TPM

First, let us consider the case where the TPM computation is done at a central node. There are many scenarios where a centralized implementation is feasible or even preferable. In a sensor network where the nodes are randomly deployed (e.g., dropped from a plane), it may be necessary and useful for the command center to obtain a map of the sensor node deployment indicating geographic voids, boundaries, etc. In this case, each node may send information about its neighbors to a base or a central station. The adjacency matrix of the network is formed based on the nodes connectivity information, which can be gathered with the worst-case complexity of $O(N^2)$ where N is the number of nodes in the network. Then the procedures explained in Section 5.3 can be used to generate an effective and accurate TPM, since there is no computational or memory limitations at the base station. Moreover, if necessary, the map can be broadcast back to the individual nodes, together with the transformation matrix (V or V_Q), an operation of worst-case complexity of $O(N^2)$. Note that redistributing 2nd and 3rd columns of V or V_Q is sufficient for a node to calculate its topological coordinate. Generating coordinates at a central station avoids multiple flooding in the network [22][23].

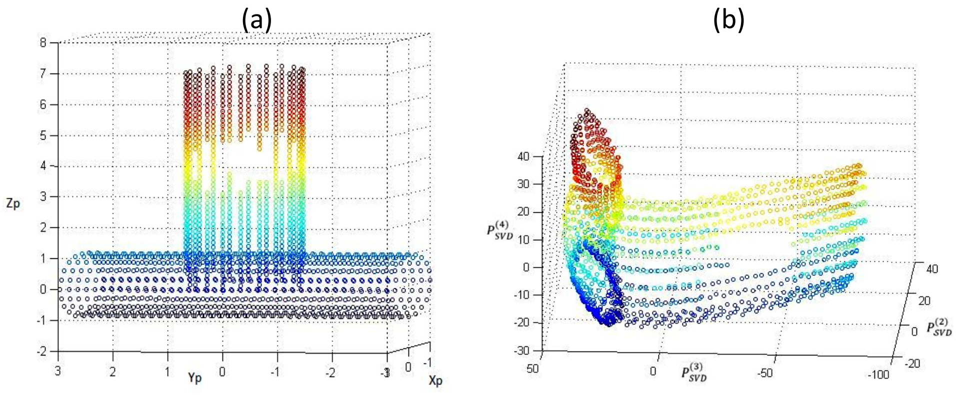


Figure 5.11. 3-D Surface network, consisting of two perpendicular cylinders (T joint) (1642 nodes, 50 randomly selected anchors): a) Physical layout; b) TPM.

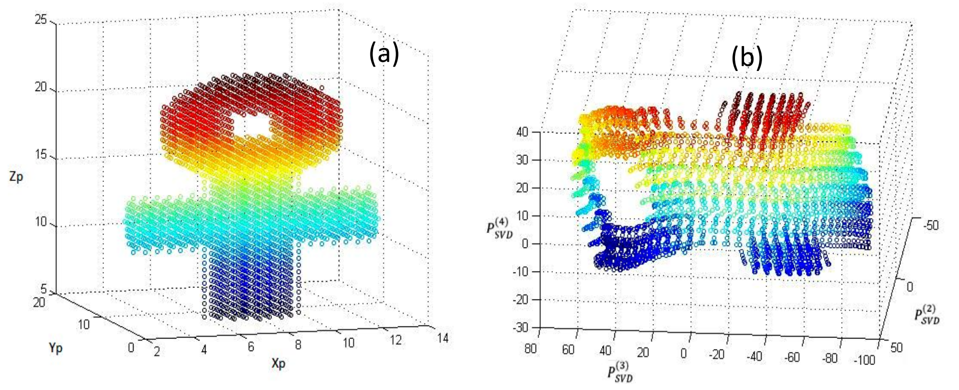


Figure 5.12. 3-D Volume network, consisting of a sphere standing on two crossed cylinders. Sphere has a hole in it. (3827 nodes and 50 randomly selected anchors): a) Physical layout; b) TPM.

Table 5.6
Performance Comparison of GLR, LCR, CSR and GPSR with 10 anchors [39]

	Avg. routability%	
Routing scheme	**Circle with voids (Figure 5.10 a)**	**Building network(Figure 5.9 a)**
GLR	94.6	89.3
LCR	56.5	49.7
CSR	87.3	75.4
GPSR	93.8	97.4

A distributed implementation of the above may be achieved as follows. The anchor-based VC generation is first carried out the traditional way, via flooding [AJ10]. Following that, the anchors broadcast their coordinates, which requires $O(MN)$ messages. Since the sub-matrix

($Q=Q_A$) of all the anchors' coordinates is now available at each node, i, every node can generate V_Q (using (7)) and compute its own $[x_{T,i}, y_{T,i}]$ locally by simply multiplying its own coordinates by 2nd and 3rd columns of V_Q.

B. TPM-based routing

In static WSNs, the VC generation needs to be done less frequently or perhaps only once during initialization. Therefore, topological coordinates also need not be updated frequently. Thus, the cost incurred in calculating Cartesian coordinates may be more than compensated by efficiency gains in terms of performance during long-term operation. For example, as illustrated in [39], Geo-Logical Routing (GLR) scheme that uses both VCS and TPM to overcome disadvantages in each other's domains outperforms the physical information-based routing scheme, Greedy Perimeter Stateless Routing (GPSR)[69]. Table 5.6 summarizes the performance of GLR, the geographic coordinate-based scheme GPSR, and two VCS-based routing schemes, namely Convex Subspace Routing (CSR) and Logical Coordinate Routing (LCR). Routability is evaluated over all possible source-destination address pairs. Additional details of GLR algorithm is available in [39].

C. TPM for dynamic networks

Network dynamics that cause changes in connectivity among nodes pose a challenge for VC-based approaches, as VC values depend on the connectivity of the network. Examples of such conditions include node failures, the introduction of new nodes, and change in connectivity due to mobile nodes. TPMs presented here capture the physical layout information of the network, i.e., the topological coordinates corresponds to the physical position of a node, albeit on a somewhat distorted layout. When a node (or even an anchor) fails, the already calculated topology coordinates (TPCs) of a node still remain valid for the topology map. Thus, any algorithm relying on TPCs can continue to function even though the underlying VCs may no longer be valid. This can be considered as an advantage of using the TPCs instead of the VCs, as VCs have to be regenerated to accommodate the change in connectivity.

Introduction of new nodes or mobility of nodes that cause major changes in network topology can render prior TPM inaccurate, thus requiring its re-computation. If the change in the connectivity pattern is completely localized, it may be possible to estimate the TCs of a new node based on some localized computations involving its immediate static neighbors. Evolution and evaluation of TPMs for such environments is under investigation.

5.7 Discussion

This section addresses the convexity of first principle component of an anchor-based VCS and the applicability of proposed TPM generation scheme for other existing virtual coordinate systems.

A. Convexity of the first principle component

Being the distance to the corresponding anchor from a node, by definition each VC radially increases around the corresponding anchor. Due to the fact that 1st principle component (PC) captures the salient dominant features of the dataset, its magnitude variation over the network is always convex; due to the possibility of having a positive or negative sign, the actual shape of 1st PC variation is either convex or concave.

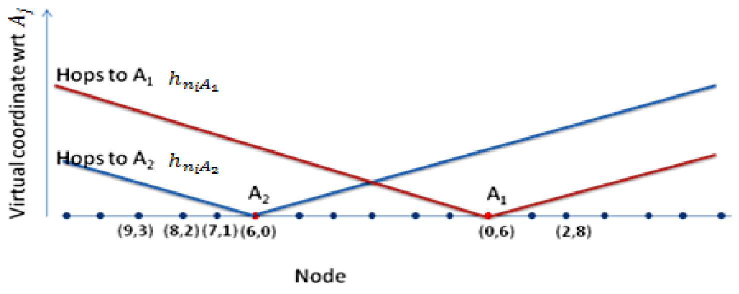


Figure 5.13: VCS corresponding to two anchors in a 1D network.

We demonstrate the convexity of magnitude first on a simple 1D network, and then extend it to a 2D full grid. Let the VCS with respect to M anchors of a 1D network, as illustrated in Figure 5.13, be $P = \left[P^{(1)}\, P^{(2)} \dots\, P^{(M)}\right] = [h_{n_iA_1}, \dots, h_{n_iA_M}]$. By definition, each virtual coordinate with respect to anchor A_j: $P^{(j)} = [h_{n_1A_j}, \dots, h_{n_NA_j}]^T$ is a convex function with respect to the node position n_i.

The 1st PC can be written as

$$P_{SVD}{}^{(1)} = [P^{(1)}\, P^{(2)} \dots\, P^{(M)}].V^{(1)} = \begin{bmatrix} h_{n_1A_1} & h_{n_1A_2} & \dots & h_{n_1A_M} \\ h_{n_2A_1} & h_{n_2A_2} & \dots & h_{n_2A_M} \\ & & \dots & \\ h_{n_iA_1} & h_{n_iA_2} & \dots & h_{n_iA_M} \\ & & \dots & \\ h_{n_NA_1} & h_{n_NA_2} & \dots & h_{n_NA_M} \end{bmatrix} \cdot [v_{11}, v_{12}, \dots, v_{1M}]^T$$

(17)

$PV^{(1)}$ is a linear combination of the set of convex functions $[P^{(1)}\, P^{(2)} \dots\, P^{(M)}]$. Reference [92] proves that the direction of 1st PC, i.e., $\mathrm{V}^{(1)}$ goes through the centroid of the data points. Since $[\mathrm{h}_{\mathrm{n_iA_1}}, \dots, \mathrm{h}_{\mathrm{n_iA_M}}]$ lie in the 1st orthant of the multidimensional space all the time, its centroid is also in the 1st orthant. Hence $\mathrm{V}^{(1)}$ is a unit vector with either all positive coefficients or all negative coefficients. Without loss of generality, one can say that (17) is the addition of M convex functions and thus 1st PC is also a convex function.

A similar argument can be made for the 2D full grid, since virtual coordinates with respect to anchor $\mathrm{A_i}$ is a 2D convex surface. Moreover, all the ordinates lie in the 1st orthant. Hence for a 2D grid, $\mathrm{V}^{(1)}$ is a unit vector with either all positive or all negative coefficients resulting in a sum of convex functions, which is convex. Therefore, 1st PC is convex for a 2D full grid as well.

B. TPM in from other VCS

The use of proposed TPM generation technique with other virtual coordinate systems derived from hop distances is demonstrated next using two such schemes, ABVCap [113] and Aligned VCS [82].

Axis-Based Virtual Coordinate Assignment protocol (ABVCap) [113] characterizes each node by a 5-tuple consisting of longitude, latitude, ripple, up, and down. These entries are specified relative to virtual lines identified in the network as follows. Initially, three anchors $(\mathrm{X}, \mathrm{Y}, \mathrm{Z})$ are selected based on VCap anchor selection. A fourth anchor, Z', is selected such that it is furthest away from Z and equidistance from X and Y. Based on hop distances to these four anchors, i.e., VCS of $(\mathrm{X}, \mathrm{Y}, \mathrm{Z}, \mathrm{Z}')$, [113] proposes a scheme to generate a new coordinate system with directionality. Generation of 5-tuple (longitude, latitude, ripple, up, down) involves several additional network floodings. Figure 5.14 (a) shows an example network used in [113] with ABVCap-VCS. One notable property of ABVCap-VCS is that some nodes have more than one VC tuple assigned to them. Either one of the tuples has to be selected for each node, which introduces unnecessary complexity to identify the proper tuple for topology map generation, or multiple positions in TPM will be assigned to the same node based on different coordinate tuples. The TPM shown in Figure 5.14 (b) is generated using our scheme based on VC tuples identified in bold in Figure 5.14 (a). Figure 5.14 (c) indicates multiple positions created for node 12 due to its multiple coordinates in ABVCap. As ABVCap-based VCS does not have concentrically increasing property, 1st PC and 2nd PC provide the TPM. While this demonstrates the applicability of TPM for ABVCap, we note that essentially the same information can be obtained simply by applying the method to a simple VCS without having to undergo overhead required to generate ABVCap.

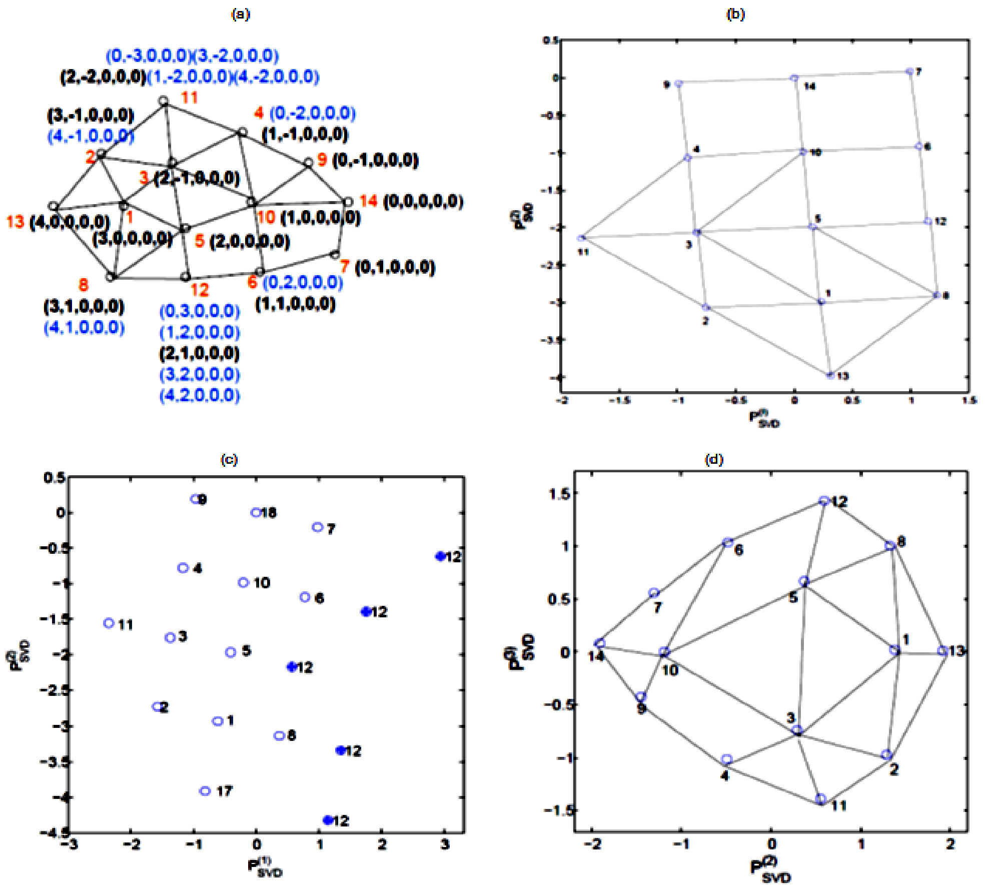


Figure 5.14: a) An example network with its ABVCap VCS [113]; b) TPM only with ABVCap coordinates in bold black, ;c) TPM with all possible ABVCap coordinates of node 12; and d) TPM from Aligned Virtual Coordinates.

Aligned VCS [82] proposes a modification for VCS to alleviate the local minima problem simply by replacing the VCs of each node with the average of the node's and its neighbors' VCs. Thus we have used the VCS w.r.t (X, Y, Z, Z') of Figure 5.B.1 (a) to evaluate aligned VCS as in [82]. The TPM from the corresponding aligned VCS is shown in Figure 5.B.1 (d). Since aligned VCs are also radial in nature, the radial component can be removed using the 1st PC, and the 2nd and 3rd PCs provide the Cartesian coordinates. These results indicate the applicability of the proposed TPM generation technique to other virtual coordinate systems as well.

5.8 Conclusions

We presented a novel and a fundamental technique for generating topology preserving maps from virtual coordinates for 2D and 3D (both surface and volume) WSNs. As demonstrated, the transformation matrix for converting the virtual (logical) coordinates to a set of topological Cartesian coordinates can be obtained using the virtual coordinates of a very small set of nodes. Results show that a remarkable 2D topology preservation error (E_{TP}) $\leq 2\%$ is achievable with a small number of anchors.

TPMs may also be used in lieu of physical maps for many applications and WSN protocols [39]. While there are certain applications for which the exact sensor location is necessary, for others that do not need such information, TPM presents a robust, accurate, and scalable alternative for physical map generation or localization. Sensor network applications of topology preserving maps are diverse and vast; examples include routing, localization, boundary node identification, and effective anchor placement. Having VCs as well as approximate Cartesian coordinates (which can now be derived from VCs) can significantly enhance the routing mechanisms.

We envision many applications of the proposed TPM extraction methodology in other types of networks as well as in multidimensional graphs, e.g., for dimension reduction, visualization, and information extraction. Methods to compensate for the distortion of the maps compared to physical maps, and techniques that use derived Cartesian coordinates and the topology map to improve self-organization and routing protocols, are also under investigation.

CHAPTER 06

PMDS: PARTIAL MULTI-DIMENSIONAL SCALING FOR SENSOR NETWORKS

6.1 Introduction

Routing plays a crucial role in data dissemination and fusion in Wireless Sensor Networks (WSNs). Though physical/geographic information-based routing [69] is popular in WSN context, the cost of acquiring physical information based on GPS is expensive and infeasible in most WSN applications. However, the same performance can be achieved under relative localization [14]. Applications such as target tracking, surveillance applications, boundary detection, etc., requires relative or global localized information of the network. Time of Arrival, Angle of Arrival, RSSI, and Triangulation are among existing relative localization techniques. Not only the energy consumption and cost associated with localization but also sensitivity to fading and SNR are among the main disadvantages of using physical information-based coordinate systems in large WSNs.

Dimension reduction schemes [16][39][109] preserve the neighborhood information. In addition, though they are rotated and/or distorted versions of the real physical node maps, most of the important physical properties such as boundaries and voids are preserved. However, most of the available dimensional reduction schemes are infeasible in WSN context due to high memory requirement, intensive computational complexity, and high communication cost involved in data acquisition.

Multidimensional Scaling (MDS) [109] is a mathematical technique that is capable of generating lower dimensional embedding of the network, given the entire Line of Sight (LoS) distance matrix among all the nodes. This lower dimensional embedding is called topology map of the network. The main disadvantage of using MDS is attaining the distance matrix, which is extremely energy-consuming in WSN context, and if such information is available at a node, the network can achieve 100% routability without generating the lower dimensional map. Moreover, storing such information at a resource-starving node, especially when the network is large, is not pragmatic.

This research proposes energy- and memory-efficient Partial MDS (PMDS) approach, which is feasible in WSN context, based on Virtual Coordinate System (VCS) [22][37]. VCS characterizes each node by vector of cardinality M, which is simply the shortest path geodesic or hop distance to subset of nodes called anchors. The main difference of PMDS compared to MDS is that the former achieves relatively localized network with the distances to 1% of the network nodes, which are randomly chosen. Furthermore, performance of PMDS is analyzed using both LoS and geodesic distances. Offline as well as inline implementations are discussed. Performance is compared with Topology Preserving Maps (TPMs) in [39].

6.2 Methodology

This section discusses the proposed algorithm Partial Multi-Dimensional Scaling (PMDS). Consider a WSN of N nodes. Subset of M nodes where $N \gg M$ are selected either uniformly at random or from the boundary [22] or are based on existing anchor selection scheme [69].

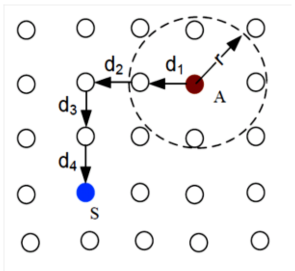


Figure 6.1. Example network with A as an anchor. S is a node which is $d_1 + d_2 + d_3 + d_4$ geodesically apart from A.

Traditional VCS is based on hop distances. However, as in this research, the concept of VCS is extended to RSSI-based shortest path geodesic distances to a set of anchors. Consider the example network in Figure 6.1. Node S is four hops away from anchor A, while the geodesic distance between A and S is $(d_1 + d_2 + d_3 + d_4)$, where d_i is the shortest geodesic gap between two nodes estimated using RSSI. The collection of these distance vectors is given by $N \times M$ matrix P. Subsection A discussed the PMDS on the entire matrix P, while Subsection B proposes partial MDS based on only anchors' distance matrix.

A. Partial MDS

Virtual Coordinate System (VCS) of all the nodes in the network w.r.t. M anchors are available and given by $P_{N\times M}$ where

$$P_{ij} = d_{ij}; i = 1,2,..N; j = 1,2,...M \tag{1}$$

d_{ij} is either geodesic or hop distance from node i to anchor j. As the first step, evaluate

$$Q_{ij} = -\frac{1}{2}d_{ij}^2; i = 1,2,..N; j = 1,2,...M \tag{2}$$

Then the matrix is 'centered' by removing column-wise and row-wise mean using centering matrices

$$C_r = I_{N\times N} - \frac{1}{N}1_{N\times 1}1_{N\times 1}^T \tag{3}$$

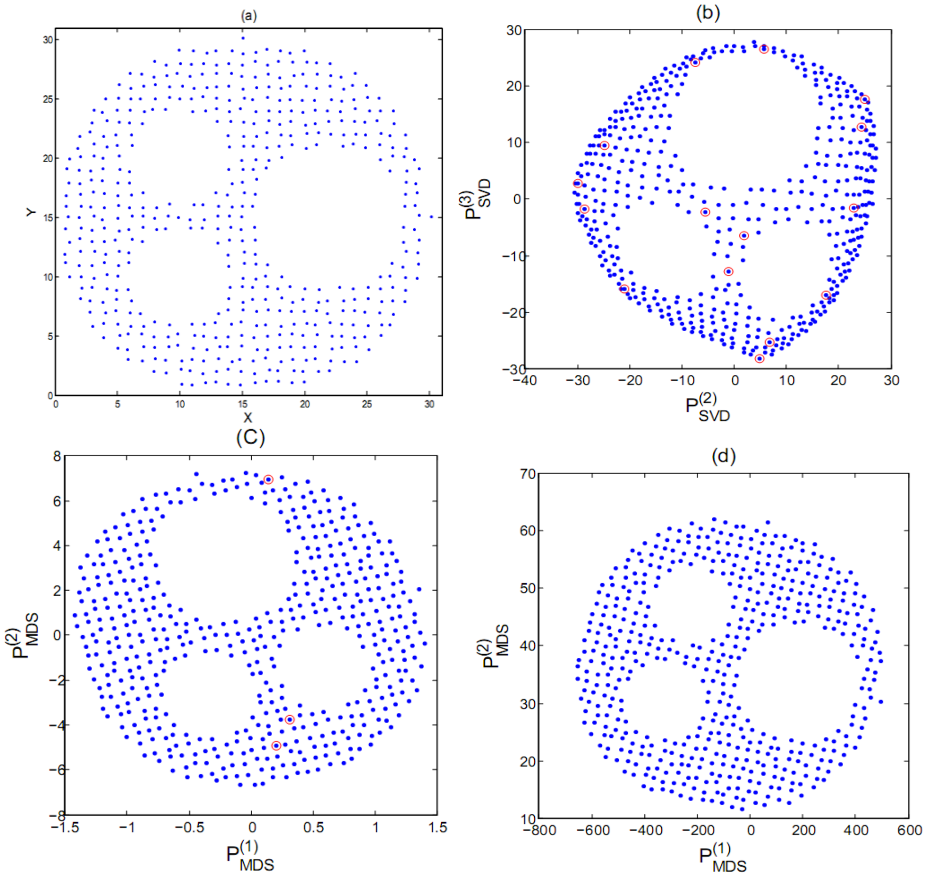


Figure 6.2. (a) 496-node circular network with three physical voids (b)TPM using SVD on RSSI based VCS with 15 anchors (c) TPM using MDS on RSSI based entire VCS with 3 anchors (d) TPM using anchors VCs only

$$C_c = I_{M\times M} - \frac{1}{M} 1_{M\times 1} 1^T_{M\times 1} \tag{4}$$

Using row- and column-wise centering matrices C_r and C_c, Q is centered.

$$R = C_r Q C_c \tag{5}$$

From the Singular Value Decomposition (SVD)

$$R = USV^T \tag{6}$$

The PMDS-based Principle Components (PCs) generate as

$$P_{MDS} = US \tag{7}$$

Note that Eq. (7) is different from the traditional MDS approach. First and second PCs are used as the physical coordinates of nodes.

$$[X_{MDS}\ Y_{MDS}] = [P_{MDS}^{(1)}\ P_{MDS}^{(2)}] \tag{8}$$

B. Partial MDS from anchors' distance matrix

Generating matrix P at a node in WSN is not pragmatic. Hence, relative coordinates generation, from the distances between anchors only, is discussed next. Consider a matrix $M \times M$ matrix P_A where

$$P_{A,ij} = d_{A_iA_j};\ i = 1,2,..M;\ j = 1,2,\dots M \tag{9}$$

$d_{A,ij}$ is the distance between anchor A_i and A_j. Following the steps from (2) to (5), centered matrix can be generated as

$$R_A = C_c Q_A C_c \tag{10}$$

where Q_A is the squared distances of P_A. Then the PCs are generated by

$$R_A = U_A S_A {V_A}^T \tag{11}$$

$$P_{MDS} = PV_A \tag{12}$$

Again, Eq. (12) is different from the conventional MDS approach. The relative physical coordinates given in (8) and be obtained and rewritten as

$$\left[X_{i,MDS}\ Y_{i,MDS}\right] = [P_i V_A^{(1)}\ P_i V_A^{(2)}] \tag{13}$$

where, P_i is the distance vector at node i, $[V_A^{(1)}\ V_A^{(2)}]$ are the first and second columns of V_A, and $\left[X_{i,MDS}\ Y_{i,MDS}\right]$ is the relative physical coordinates of node i. Computational complexity of performing MDS on $N \times N$ matrix is $O(N^3)$ while the presented MDS on anchor's distance matrix, which is $M \times M$, is $O(M^3)$.

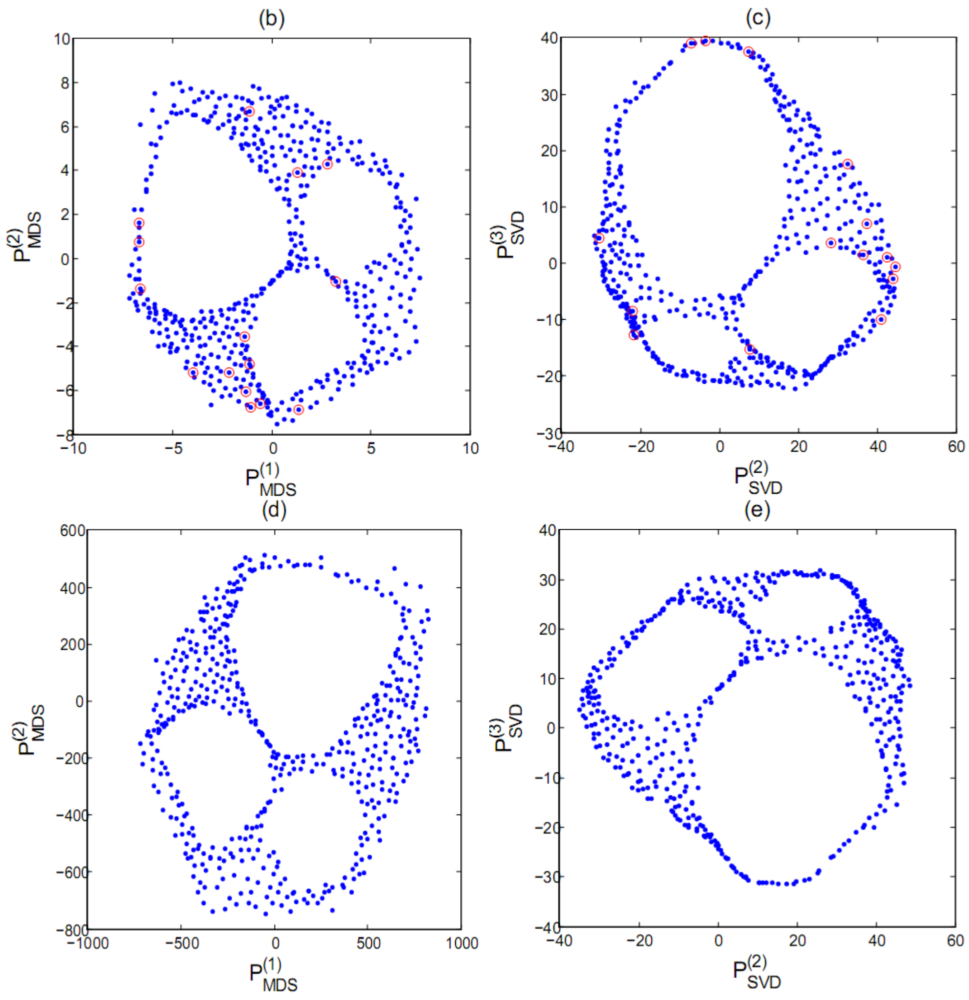


Figure 6.3. (a) 496-node circular network with three physical voids, (b)TPM using RSSI based VCS, (c) TPM using MDS, and (d) TPM using anchors VCs only

6.3 Realization of the Topology Map Generation

In this section, we consider the realization of the algorithm in wireless sensor networks.

First, consider a sensor network where the nodes are randomly deployed, and it is necessary to obtain a map of the node deployment indicating geographic voids at a central node. There are many scenarios where a centralized implementation is feasible or even preferable. In this case, each node may send information about its VC to a base or a central station. Thus, the above procedure (Section II-A) can be used to get a map of the network at a significantly lower cost compared to that of MDS approach. If necessary, the map can be broadcast back to the individual nodes, together with the transformation matrix V, an operation of transmission complexity of $O(N)$. Generating coordinates at a central station avoids multiple flooding in the network caused by distributed implementation of the same algorithm.

A distributed implementation of the above may be achieved as follows. The anchor-based distance evaluation, in terms of geodesic distance, is carried out the traditional way initially, via flooding. Following that, the anchors broadcast their coordinates, which requires $O(MN)$ messages. Now that all the anchors' coordinates array is available at each node, each node i can generate $V_A(Eq.$ (11) and (12)), and evaluate its own $[X_{i,MDS}\ Y_{i,MDS}]$ locally by simply multiplying its own coordinate by V_A.

6.4 Results

The performance of the proposed methods is evaluated using a 496-node circular-shaped network with three physical voids as shown in Figure 6.2 (a). MATLAB® 2011a was used for the computations. Communication range of a node is 1.5 units and the maximum neighborhood size of a node is 8.

Two application scenarios are discussed: (1) set of nodes with high transmission capabilities as anchors and (2) subset of network nodes as anchors, thus anchors do not have any additional capabilities (Subsection C). Moreover, the proposed PMDS is compared with SVD-based Topology Preserving Maps [39], which are briefly discussed in Subsection A. Topology Preservation Error (E_{TP}%) introduced in [39] is used as the performance evaluation metric. E_{TP}% simply evaluates the percentage number of node flips in the entire topology map compared to its physical map.

A. Topology Preserving Maps (TPMs)

TPM scheme proposed in [39] uses one hop distance-based VCS while we apply the technique on LoS and geodesic distances based VCSs. SVD of matrix *P* is evaluated without centering the matrix

$$P = U_S . S_S . V_S^T \tag{14}$$

Then the projection of P on to basis V_S is calculated

$$P_{SVD} = P \cdot V_S \tag{15}$$

Since the matrix P is not centralized, 1st principle component has captured the radial feature of the VCS. Therefore, 2nd and 3rd principle components are used as the coordinates of TPM

$$[X_T, Y_T] = [P_{SVD}{}^{(2)}, P_{SVD}{}^{(3)}] \tag{16}$$

B. Set of powerful nodes as anchors

Assume there is a set of nodes that can transmit signals to the entire network, serves as anchors, and are used only for the purpose of localization during the network setup phase. Thus, each sensor node can estimate the LoS distance from itself to the anchors based on the RSSI. Collected estimates of each node's VC can be used to get the TPM at a central station as discussed in Section III. Figure 6.2 (b) shows the performance of SVD-based TPMs, based on 15 anchors. Next, Figure 6.2 (c) is corresponding to a map constructed based on Partial MDS given in Eq. (8) using entire *P* matrix for all the nodes in the network. However, the noticeable fact is that number of anchors is just three. More interestingly, Figure 6.2 (d) indicates the inline implementation where the map is derived from just anchors distance matrix of 3×3. As given in Table 6.1, the average topology preserving error (E_{TP}) in Partial MDS maps are zero while that of SVD-based offline and inline implementations are 0.02% and 0.2% respectively.

C. Subset of network nodes as anchors

Next, we consider a subset of nodes in the network as anchors; thus, anchors do not have any additional capabilities. As in the traditional VCS generation, selected anchors flood the network. Then the rest of the nodes in the network develop geodesic distance-based VCS.

Table 6.1
Topology Preservation Error for Figure 6.2 and Figure 6.3

	$E_{TP}\%$			
	Offline		Inline	
	SVD	PMDS	SVD	PMDS
Set of powerful nodes as anchors	0.02	0.0	0.2	0.0
Subset of nodes as anchors	1.06	0.82	1.96	1.6

Fifteen anchors are randomly placed. Partial MDS- and SVD-based topology map of the offline implementation that is based on the entire VCS is as shown in Figure 6.3 (a) and (b) respectively. $E_{TP}\%$ (Topology Map Error) is estimated by averaging the error for 10 random anchor configurations. $E_{TP}\%$ for PMDS- and SVD-based offline implemented maps are 0.82% and 1.06% respectively. Figure 6.3 (c) and (d) show the TPMs of inline implementation of PMDS- and SVD-based approaches, with $E_{TP}\%$ of 1.96% and 1.6% respectively. Based on the $E_{TP}\%$ performance, we can conclude that the performance of PMDS-based TPMs is better than that of the SVD-based approach.

In-network implementation of conventional MDS is not pragmatic mainly because, if a node contains the entire network distance matrix, it can track the path to destination without requiring any complex routing scheme. Both inline and offline implementation costs are significantly higher than the proposed PMDS because, in conventional MDS, every node floods the network for distance matrix evaluation. Hence the transmission complexity is $O(N^2)$. In contrast, proposed PMDS transmission complexity for distance matrix evaluation alone is $\mathrm{O(MN)}$. Since the distance among all the nodes is required for MDS, the memory requirement is $O(N^2)$ while that for PMDS is $O(MN)$.

6.5 Conclusion

Partial Multi-Dimensional Scaling (PMDS) is a pragmatic and Wireless Sensor Network (WSN) friendly approach of achieving topology-preserving maps, without requiring physical information. In contrast to MDS, PMDS uses geodesic distances to a subset of nodes in the network. Thus, computational, memory, and communication complexity PMDS are significantly lower than those of MDS, especially when it comes to large WSNs. Performance comparison based on topology preservation error, a metric that captures the number of node flips due to lower dimensional embedding, shows that maps generated using PMDS are better than that of maps generated using the existing scheme, Topology Preserving Maps.

CHAPTER 07

ANCHOR SELECTION AND TOPOLOGY PRESERVING MAPS IN WSNS – A DIRECTIONAL VIRTUAL COORDINATE BASED APPROACH

7.1 Introduction

Virtual Coordinate Systems (VCS) provide a compelling alternative for structuring/organizing Wireless Sensor Networks (WSNs) without the need for location information. In virtual domain, each node is represented by a M dimensional vector of virtual coordinates, consisting of shortest path hop distances to M nodes, named anchors. They are different from anchors used in physical position-based localization schemes in that anchors for VCS are not localized and have the same capabilities as any other node in the network. A major advantage of connectivity information-based VCS over physical position-based systems is that they completely avoid the cost, complexity, and uncertainties associated with node localization using GPS or distance estimations. GPS-equipped nodes are expensive, are not energy efficient, and are not feasible in some applications [14]. The alternative is to use analog measurements, such as Receiver Signal Strength Indicator (RSSI) or Time of Arrival (TOA) [14], to estimate distances to other nodes and thereby obtain node positions. These analog measurements encounter uncertainties and complexities due to multipath fading, scattering, interference, and poor line-of-sight, which are difficult to overcome in many indoor and outdoor environments. VCS thus outcompetes geographical location-based schemes in large WSNs by facilitating the use of nodes with simpler hardware. VCS also possess the advantage of having connectivity information implicitly embedded in node coordinates generated based on hop distances.

VCS is a WSN-friendly self-organization strategy that provides routing schemes with performance, comparable to alternatives that require costly localization [4][22][23][36][41]. VC-based routing (VCR) schemes generally use greedy forwarding, in which a packet is forwarded to the neighbor closest to the destination. The distance between two nodes for this purpose is typically calculated using L^2 norm in VC space. Early VCR schemes such as LCR [23] and CSR [36] differ in the way they deal with local minima or logical voids in this space and the way backtracking is done.

As each Virtual Coordinate (VC) propagates in the network concentrically from the corresponding anchor, VC systems appear to lose the sense of (cardinal) directionality that is present in geographic coordinate-based systems. Two recent developments go a long way toward overcoming this hurdle by recovering or regenerating directional information lost in VCs. The first is the methodology for generating Topology Preserving Maps (TPMs) for WSNs using VCS [38]. TPMs and the associated Topology Coordinates provide a powerful alternative for physical maps and location information. TPMs preserve the neighborhood information, boundaries, voids, and even the general shape of networks, but they have neither the exact coordinates nor the distances of the physical layout. TPM and VCS are combined in a routing scheme called Geo-Logical Routing [39], which generally outperforms even the exact physical location-based routing scheme, Greedy Perimeter Stateless Routing (GPSR) [69]. The second development is the VC to Directional Virtual Coordinate (DVC) transformation we proposed in [41], which embeds virtual directions within a network. The routability is significantly enhanced by using such directional information in conjunction with VCS [41].

The number of anchors and their placement, i.e., the anchor selection, is the key factor affecting the performance of any VCS-based algorithm, including those for routing [23][36] and TPM generation [37]. Under-placement of anchors results in identical coordinates for different nodes. In general, the routability increases and the likelihood of identical coordinates decreases with the increase of number of anchors. But, as [36] indicates, over-placement of anchors may degrade routing performance. Not only that, the higher the number of anchors, the higher the VC generation cost and the higher the length of address field in packets, causing higher energy consumption.

Determining the optimal number of anchors for a network, however, is not trivial. Further complication is due to the fact that the number of anchors and their optimal placement are highly correlated. Adding the (n+1)st anchor to the optimal placement of *n* anchors does not in general result in the optimal placement of (n+1) anchors. A ‘good’ set of anchors provide high routability, while improper placement of the same number of anchors may cause excessive routing failures due to local minima [23][36][41][37] and nodes with identical coordinates [22][36]. When the node currently holding the packet is unable to find a neighbor that is closer than itself to the destination, the packet is said to be at a local minima. How internal anchors cause local minima that impact routing is addressed in [36], which explains the importance of placing the anchors on the boundary. Existing VC-based Routing (VCR) schemes use a costly backtracking phase to overcome such local minima, yet packet delivery is not guaranteed. Schemes in [22][23][101] attempt to place anchors as far apart as possible, or on the boundary. Those schemes are expensive to implement, and do not guarantee to achieve the stated goal. Not all boundary nodes are ‘good’ anchors, either. Discovering boundary nodes without any localization is not straightforward [128]. Boundary detection for the placement of anchors also increases the cost of VCS generation as it involves multiple network floodings [22][23][101]. To manage this complexity, many existing schemes determine apriori, the number of anchors to be placed, e.g., a certain percentage of the nodes, and then attempt to place them on boundary or space them far apart [22][23][128]. Thus determination of the number of anchors together with their placement remains a major challenge. No holistic solutions that determine both the number of anchors and their placement simultaneously based on the network topology exist.

The first contribution of this research is a simple, efficient, and also a very effective scheme for anchor placement. The proposed Extreme Node Search (ENS) algorithm simultaneously determines both the number of anchors and their locations. The anchor nodes selected in general are extreme nodes of the network. Nodes at corners of a network and nodes that are furthest apart are examples of extreme nodes. ENS is a completely distributed scheme that allows a node to self-identify whether it is an anchor or not. It exploits the directionality introduced to VCS by the Directional Virtual Coordinate Systems (DVCS) approach [41]. The quality of anchors selected is evaluated based on improvement of routability achieved with Greedy Forwarding (GF) and existing VCR schemes: Directional Virtual Coordinate Routing (DVCR) [41], Convex Subspace Routing (CSR) [36], and Logical Coordinate Routing (LCR) [23]. The performance of these schemes with anchors selected by ENS is also compared with Greedy Perimeter Stateless Routing (GPSR) [69], a geographical coordinates-based scheme that requires the knowledge of nodes’ geographic coordinates. Results presented for several representative networks demonstrate that ENS anchor selection not only improves the routing performance of existing VCS-based routing, but also improves the quality of TPMs generated based on [37]. In the process of developing and providing insight to the anchor placement, we also derive additional properties, including the notion of angle between two vectors in DVC space. If two vectors have ~ 90^0 angle between them, they provide a good set of Cartesian axes which can also be used to get a good network map. This is the second contribution of this research. The proposed TPM technique is computationally less complex than that in [37], which is based on singular value decomposition. The notion of specifying cardinal directions and use of angles, introduced in this research for the first time, is a radical change from the traditional VC system approaches. This, we believe, will significantly enhance the ability to exploit VCs for networking.

Section 7.2 discusses work related to anchor placement and VCS-based routing. Then Section 7.3 derives two basic properties of DVCS for a full grid. ENS-based on DVCS is discussed in Section 7.4, followed by performance evaluation in Section 7.5. Section 7.6 discusses existing TPM generation technique for WSNs. Section 7.7 proposes and evaluates TPM based on DVCS. Finally, Section 7.8 concludes the research.

7.2 Anchor Selection and Routing in Virtual Coordinate Systems

Though good anchor placement is critical for VCS performance, only a few schemes are available for discovering good anchor locations. Anchor placement scheme in [101] proposes placing them on the WSN boundary, assuming that the boundary nodes are known. But most existing boundary identification schemes require physical localization, which is expensive. Virtual Coordinate assignment protocol (VCap) [22] proposes generating a virtual

Table 7.1
Notations Used In Chapter 7

Notation	**Description**
N	Number of network nodes
n_i	Node i
M	Number of anchors ($M << N$)
$A_i, i = 1{:}M$	Set of Anchors
$h_{n_i n_j}$	Minimum hop distance between nodes n_i, n_j
$P_{(i)} = [h_{n_i A_1}, \dots, h_{n_i A_M}]$	Node n_i's VC
$P_{D(i)}$	Node n_i's Directional VC
$P_D^{(i)}$	ith coordinate of DVCS
$\vec{u}_{A_i A_j}$	Unit vector of the directional coordinate w.r.t. anchor A_i and A_j
Φ_{ij}	Angle between $P_D^{(i)}$ and $P_D^{(j)}$
$K_h(n_i)$	Set of nodes in n_i's h-hop neighborhood
g_{xy}	Displacement between node n_x and n_y in DVCS

topology based on a total of three furthest apart for the entire network. In general, multiple nodes flood the network when self-election of anchors is aimed at optimization of prior coordinates. Moreover, this results in a large number of nodes with identical coordinates. Logical Coordinate-based Routing (LCR) [23] combines an anchor placement strategy together with GF. Again, the anchor placement scheme attempts to select the furthest apart nodes, but the number is not restricted to three. A backtracking (BT) algorithm is used when a packet reaches local minima. The mode of data packets, GF/BT, is kept in the packet header while their recently visited nodes are kept in a distributed manner at intermediate nodes. Strategies for evaluating anchors and VCS dimensionality reduction are presented in [38].

Single anchor-based VCS called Spanning Path Virtual Coordinate System (SPVCS) in [84] uses a depth-first search algorithm starting from the root node for coordinate generation. Best performance is achieved when the anchor or the root node is placed at the center, as it provides a balanced spanning tree. Convex Subspace Routing (CSR) algorithm [36] moves dynamically among different subspaces of VCS to avoid local minima. Directional Virtual Coordinate Routing (DVCR) [41] is a routing scheme based on Directional VCS (DVCS) in which a node uses DVCS-based greedy forwarding when possible. At a local minima, an approximation for hop distance based on two directional coordinates is used for next node selection.

7.3 Properties of Directional VC Space

Consider a 2D network with N nodes and M anchors. VCS of the network characterizes the node n_i by $[h_{n_iA_1}, \dots, h_{n_iA_M}]$, where $h_{n_iA_j}(\geq 0)$ is the shortest path hop distance from node n_i to anchor A_j. The notations used is summarized in Table 7.1.

Each coordinate of DVCS is obtained using a pair of anchors in VCS. The pair of VCs of node n_i with respect to A_j and A_k, $[h_{n_iA_j}, h_{n_iA_k}]$ is transformed to the corresponding directional virtual coordinate (DVC) of node n_i as [41]:

$$f\left(h_{n_iA_j}, h_{n_iA_k}\right) = \left(h_{n_iA_j}{}^2 - h_{n_iA_k}{}^2\right)/2h_{A_jA_k} \tag{1}$$

This transformation basically introduces directionality to the space, i.e., $f\left(h_{n_iA_j}, h_{n_iA_k}\right)$ spans negative and positive values, with zero at the midpoint between the two anchors. This restores the directionality lost in original VCs, which corresponds to positive values propagating radially away from each anchor. Therefore, a vector notation can now be introduced. Let $\vec{\boldsymbol{u}}_{\boldsymbol{A_jA_k}}$ be the unit vector in virtual direction from A_j to A_k in the virtual domain. The vector component of node n_i in DVCS in the direction $\{A_j, A_k\}$, i.e., from anchor A_j to A_k, is [41]

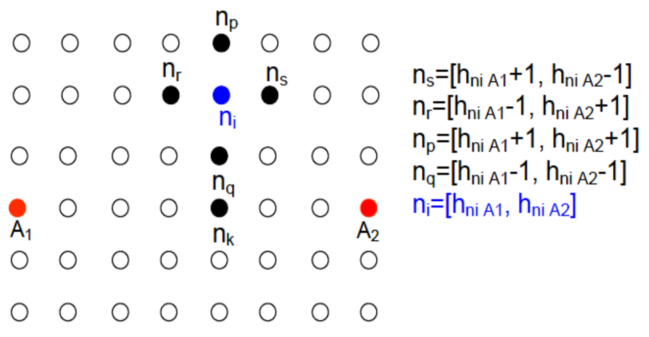
$$\vec{\boldsymbol{f}}\left(\boldsymbol{h}_{\boldsymbol{n_iA_j}}, \boldsymbol{h}_{\boldsymbol{n_iA_k}}\right) = f\left(h_{n_iA_j}, h_{n_iA_k}\right)\vec{\boldsymbol{u}}_{\boldsymbol{A_jA_k}} \tag{2}$$

$f\left(h_{n_iA_j}, h_{n_iA_k}\right)$ is the magnitude in the direction of $\vec{\boldsymbol{u}}_{\boldsymbol{A_jA_k}}$.

The displacement from nodes n_x to n_y (with respect to the coordinate defined by A_j and A_k), denoted by g_{xy}, is therefore

$$g_{xy} = -g_{yx} = f\left(h_{n_yA_j}, h_{n_yA_k}\right) - f\left(h_{n_xA_j}, h_{n_xA_k}\right) \tag{3}$$

Two properties of DVCS are derived next for a full rectangular grid network as shown in Figure 7.1 in which each node communicates with four neighbors. This simple case provides insight into the directional coordinates and helps explain the principles behind the anchor placement algorithm. Even though the assumptions are not compatible with general networks, with odd shapes, voids, etc., the results still can be used as a first order approximation to analyze such networks.



n_p
n_r
n_s
n_i
n_q
n_k
A_1
A_2
n_s=[$h_{ni\,A1}$+1, $h_{ni\,A2}$-1]
n_r=[$h_{ni\,A1}$-1, $h_{ni\,A2}$+1]
n_p=[$h_{ni\,A1}$+1, $h_{ni\,A2}$+1]
n_q=[$h_{ni\,A1}$-1, $h_{ni\,A2}$-1]
n_i=[$h_{ni\,A1}$, $h_{ni\,A2}$]

Figure 7.1. VCs of n_i and its neighbors on a grid with two anchors A_1 and A_2.

Property 1: In a full rectangular grid, when two anchors are placed furthest apart on a connected line L, the displacement between any two adjacent nodes on a line parallel to L (line L_h) is a constant that is greater than unity for each line. Moreover neighbors on a line perpendicular to L (line L_v) have a constant displacement that is less than unity for each line.

Proof: Consider nodes on line L_h passing through nodes n_r, n_i and n_s in Figure 7.1. The four neighbors of node n_i are n_p, n_q, n_r, and n_s. Latter two neighbors, with DVCs $(h_{n_iA_1}+$ *hniA2hniA1−hniA2−2/2hA1A2* and *hniA1+hniA2 (hniA1−hniA2+2) /2hA1A2* respectively, are on a line parallel to the line through anchors. As the sum of the two VCs, $h_{n_iA_1}+h_{n_iA_2}$, of any node on the line is fixed, the displacements g_{is} and g_{ri} evaluated using (4) are the same:

$$g_{is}=g_{ri}=\left(h_{n_iA_1}+h_{n_iA_2}\right)/h_{A_1A_2}>1 \tag{4}$$

Now consider the other pair of neighbors, n_p and n_q, on line L_v, with DVCs $(h_{n_iA_1}-h_{n_iA_2})(h_{n_iA_1}+h_{n_iA_2}+2)/2h_{A_1A_2}$ and $(h_{n_iA_1}-h_{n_iA_2})(h_{n_iA_1}+h_{n_iA_2}-2)/2h_{A_1A_2}$ respectively. The displacements g_{ip} and g_{qi} (from (2)) are the same:

$$g_{ip}=g_{qi}=\frac{\left(h_{n_iA_1}-h_{n_iA_2}\right)}{h_{A_1A_2}}=\frac{\left(h_{n_kA_1}-h_{n_kA_2}\right)}{h_{n_kA_1}+h_{n_kA_2}}=<1 \tag{5}$$

where, n_k is the node at the crossing point for a given vertical line on which n_i lies and the line through the anchors as shown in Figure 7.1. QED.

Next we define extreme node and property 2, which will be used to identify ‘good’ anchors.

Definition: A node is an extreme node if it is at a local minimum/maximum within its h-hop neighborhoods in its DVC system generated w.r.t two given anchors.

Note that minima occur due to negative values in DVCs.

Property 2: In a full rectangular grid, an internal node cannot be an extreme node.

Proof: Let any two anchors be A_j and A_k. Without loss of generality, consider the transformation at n_i and its 1-hop neighborhood ($K_1(n_i)$) given by (1). If the node n_i is a middle node, then there exist neighbor nodes with VCs $(h_{n_iA_j} \pm a, h_{n_iA_k} \pm a)$, where a is 1 or 0. Thus, directional coordinate of any of the neighbors can be expressed as

$$\left(h_{n_iA_j}{}^2 + a \pm 2ah_{n_iA_j} - h_{n_iA_k}{}^2 - a \pm 2ah_{n_iA_k}\right)/2h_{A_jA_k}$$

$$= f\left(h_{n_iA_j}, h_{n_iA_k}\right) \pm \left(2ah_{n_iA_j} + 2ah_{n_iA_k}\right)/2h_{A_jA_k}$$

Thus n_i's coordinate is neither a minimum nor maximum. In contrast, if n_i is an extreme node the only possibility for the neighboring VCs is $(h_{n_iA_j} - a, h_{n_iA_k} - a)$. Again a is 0 or 1. Neighbors' coordinates from (1),

$$\left(h_{n_iA_j}{}^2 + a - 2ah_{n_iA_j} - h_{n_iA_k}{}^2 - a + 2ah_{n_iA_k}\right)/2h_{A_jA_k}$$

$$= f\left(h_{n_iA_j}, h_{n_iA_k}\right) - \left(2ah_{n_iA_j} - 2ah_{n_iA_k}\right)/2h_{A_jA_k}$$

If $h_{n_iA_j} > h_{n_iA_k}$, where a is 1, $n_i$is at a maximum while if $h_{n_iA_j} < h_{n_iA_k}$ then n_i is at a minimum. When a is zero, i.e., under identical coordinate situation, both nodes with identical coordinates $(h_{n_iA_j}, h_{n_iA_k})$ are at minima/maxima. QED.

Now consider the circular network with three physical voids in Figure 7.2. Let $\{L_h\}$ be the sets of lines that are parallel to line L joining the anchors, i.e., those in $\vec{\boldsymbol{u}}_{\boldsymbol{AB}}$ direction, and $\{L_v\}$ be the set of lines that are perpendicular to L, i.e., those perpendicular to $\vec{\boldsymbol{u}}_{\boldsymbol{AB}}$ direction.

We identify $\{L_h\}$ based on the gap between neighbors given by (4) of Property 1. $\{L_v\}$ lines are identified using (5). Figure 7.2 clearly illustrates that nodes can identify the $\{L_h\}$ and $\{L_v\}$ lines they are on with a high degree of accuracy. Only 65 out of 496 nodes cannot identify their $[L_h, L_v]$ position accurately. This illustrates that the results for full rectangular grid can serve as a good first order approximation. Gray squares are the 65 discordant nodes not compatible with the first order approximation.

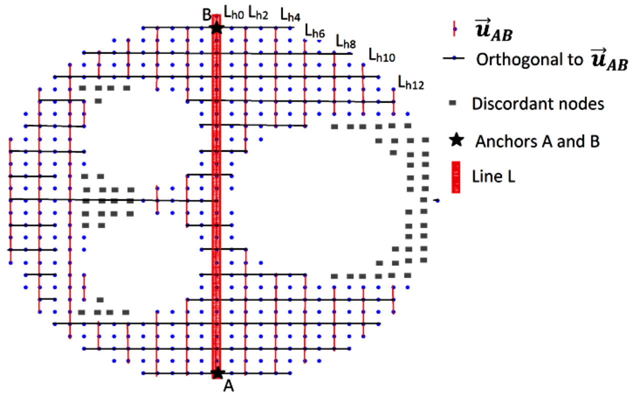


Figure 7.2. $\{L_h\}$ and $\{L_v\}$ lines of a circular network with three physical voids where anchors are at A and B. Only the even numbered lines in the two directions are shown for clarity.

7.4 Extreme Node Search (ENS) - An Anchor Selection Algorithm

Anchor placement has a critical impact on the performance of any VCS-based routing algorithm. Determining the optimal number of anchors and their placement are open problems. The fact that the two problems are interdependent makes their solution even more challenging. Optimum number of anchors can be defined as the minimum set of anchors that provide unique VCs for each node and facilitate 100% success in routing using shortest length paths. In this section we propose a simple but effective anchor placement scheme, Extreme Node Search (ENS), that attempts to assign extreme nodes, such as furthest apart and corner nodes of the network as anchors. Results below show that the selected extreme nodes are almost always situated far apart if not furthest apart.

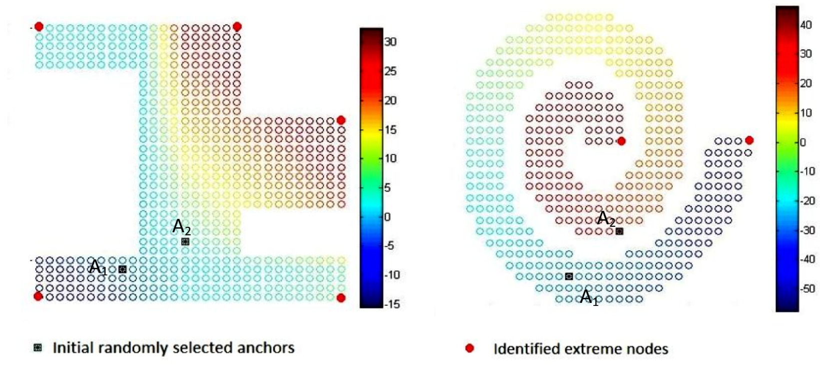


Figure 7.3. Scatter plot of DVCS coordinate for two sample networks to identify extreme nodes for anchors. Initial anchor pair is (A1,A2), color corresponds to DVCS value, and highlighted nodes are the identified extreme nodes.

```
Inputs: Neighbors set of $n_i$; $n_j \in K_h(n_i)$ ,
Output: ENS anchor candidate
Step 1 - Two random nodes initiate floodings to generate a VCS
Step 2 - Each nodes locally generates its DVCS using (1)
Step 3 - Each node checks whether it is a local minimum/maximum in h-hop neighborhood:
  If $f(h_{iA_1}, h_{iA_2}) < f(h_{jA_1}, h_{ij})$ ; $\forall\, n_j \in K_h(n_i)$
        $n_i$ is an anchor
  End
OR
  If $f(h_{iA_1}, h_{iA_2}) > f(h_{jA_1}, h_{ij})$ ; $\forall\, n_j \in K_h(n_i)$
        $n_i$ is an anchor
  End
Step 4 – Selected anchor nodes generate the VCS
```

Figure 7.4. ENS anchor selection algorithm at a node.

The strategy uses a pair of anchors, which may be randomly selected, to generate a DVC at each node. Figure 7.3 indicates the scatter plot of DVC by the initial pair of randomly selected anchors for two example networks, which can be used to identify maxima/minima, i.e., the extreme points of the network.

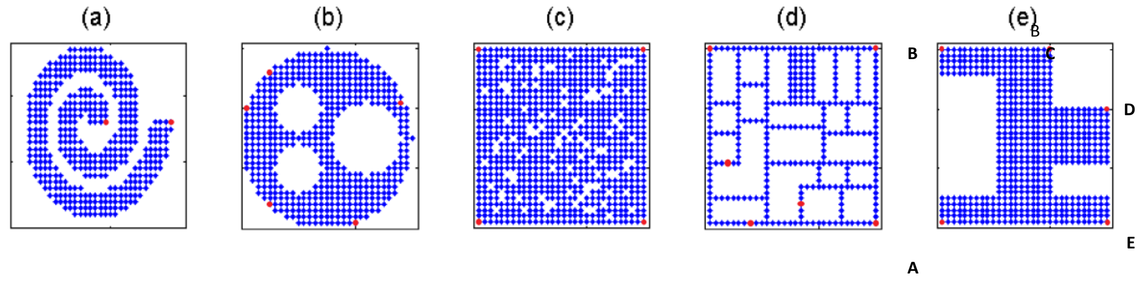


Figure 7.5. (a) Spiral shaped network with 421 nodes, (b) a 496-node circular shaped network with three physical voids/holes, (c) A grid based network with 100 randomly missing nodes, (d) a network of 343 nodes mounted on walls of a building, and (e) odd shaped network with 550 nodes.

ENS consists of three steps. First, two randomly selected nodes (A_1 and A_2 in Figure 7.3) flood the network, thus characterizing each node by two VCs. Then each node evaluates the corresponding DVC using (1). Third, each node evaluates whether it is a local minima/maxima in DVCs, within its h-hop neighborhood, $K_h(\boldsymbol{n}_i)$. h is an important parameter that we can use to restrict the identified anchors. If it is a local maxima or minima in the neighborhood, the node decides to become an anchor of the network. Now a new VCS is generated using these new anchor nodes. ENS is summarized in Figure 7.4.

Initially selected anchors can also supplement the selected anchors to be used for routing, if necessary. Another fact to notice is that ENS determines the anchor locations as well as the number of anchors for a given network.

7.5 Effectiveness of ENS Anchors

In this section, the performance and the effectiveness of the anchor placement is investigated. Five examples, shown in Figure 7.5, represent a variety of network topologies: (a) spiral-shaped network with 421 nodes; (b) a 496-node, circular-shaped network with three physical voids/holes [37]; (c) a 30x30 grid network with 100 randomly missing nodes (800 nodes); (d) a network of 343 nodes mounted on walls of a building [38]; and (e) an odd-shaped network with 550 nodes [38] were used for performance evaluation. Communication range of a node in all five networks is unity. MATLAB® 2010b was used for the computations. We use a 4-hop neighborhood, i.e., $h = 4$, for selecting extreme nodes.

The nodes colored in red in Figure 7.5 are those identified by the algorithm as the extreme nodes, i.e., local minima/maxima in a 4-hop neighborhood. Thus, they become the anchors for VCS generation. For all the networks except for those in Figures 7.5 (b) and (d), these very same nodes are identified with a high probability by the scheme irrespective of the initial selection of the two random anchors. The five red nodes in Figures 7.5 (b) and (d) are for a particular selection of the initial two nodes. This can be attributed to the fact that Figure 7.5 (b) has a circular boundary, resulting in many possible candidates for extreme nodes. For Figures 7.5 (b) and (d), the number of anchors selected varies between 3-8 and 3-10 respectively for different initial anchor selections.

A. Improvement in Greedy Forwarding

Greedy Forwarding (GF) is the underlying mechanism on which existing VC routing schemes [22]-[41] are based. The percentage of packets successfully routed to the destination is called the Greedy Ratio (GR). A good anchor placement is one that enables high GR; thus, it is a good measure to evaluate the effectiveness of an anchor placement strategy. GR is evaluated for anchors selected by the ENS and is compared with that for several other strategies in Figure 7.6. Other anchor placement schemes include: (a) anchors selected anywhere in the network uniformly at random and (b) anchors selected from the boundary nodes (again uniformly at random, similar to that in [22][23]) for the network in Figure 7.5(e). The tolerance bars in Figure 7.6 indicate the maximum and minimum variation of GR for different random anchor configurations for anchor placement schemes in (a) and (b) above. GR increases from 46.9% to 84.9% as the number of randomly selected anchors from anywhere in the network varies from 5 to 50. As the number of boundary anchors increases from 5 to 50, GR increases from 54.5% to 90.7%. GR for the five anchors selected by ENS, i.e., as in Figure 7.5 (e), is 93.3%, achieving higher routing performance with just five anchors. Note that the horizontal axis of Figure 7.5 has no significance for the ENS as it comes up automatically with maximum five anchors. None of the other schemes can achieve a similar performance even with 50 anchors. We have included results only for network in Figure 7.5 (e) due to limited space. Results for other network configurations also demonstrate a similar improvement.

To evaluate whether the anchor placement provided by ENS can be improved upon, next we add random nodes from the boundary as additional anchors beyond the five already selected. GR, which is 93.3% for the selected five anchors, increased gradually to 95.5% when the total number of anchors is 50 anchors (five selected anchors from ENS and 45 randomly selected boundary nodes). Thus additional 45 anchors provide just 2% increment in routability. This indicates that anchors from ENS are able to identify the most efficient set of anchors while guaranteeing high routability. Finally GR is evaluated using ENS selected anchors in DVCS-based GF. In other words, greedy decision is based on L^2 distance estimation using entire DVCS. As opposed to VCS, DVCS-based GF achieves 99.3% GR under anchors from ENS. Also, DVCS-based GF outperforms VCS-based GF under any anchor placement scheme which is an indication of the effectiveness of DVCS.

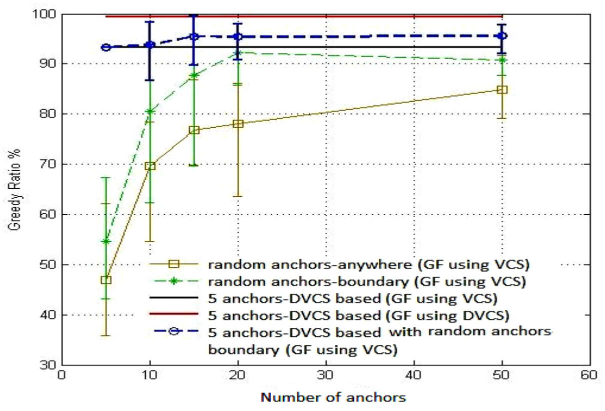


Figure 7.6. Comparison between, greedy ratio of randomly selected anchors from the boundary/anywhere in the network and that of five anchors selected from ENS for the network in Figure 7.5(e).

In evaluating any anchor placement scheme, it is extremely important to consider the impact on communication and computation complexity associated with the number of anchors. For M anchors, the VC generation complexity, i.e., the number of packets needed to generate the VCS, is $O(MN)$. The length of address field is $O(M)$. Not only does the ENS select anchors in the range of 2-10 for the given networks in Figure 7.5, the routability performance with these few anchors is better than that with 50 anchors (randomly selected or selected on the boundary), resulting in an order of magnitude reduction in complexity while achieving better performance. Selection of anchors on the boundary normally requires apriori knowledge of boundary nodes or use of a boundary node identification scheme that significantly adds to the complexity. Proposed scheme overcomes this hurdle simply relying on the DVCS generated from two random anchors to identify the anchor placement.

Table 7.2
R_{AVG}% and H_{AVG} of DVCR, CSR, LCR and GPSR

		Five random anchors			**With ENS anchors**			
		CSR	LCR	**DVCR**	CSR	LCR	**DVCR**	**GPSR**
Spiral	R_{AVG}%	49.97	46.9	**92.7**	61.66	67.3	**95.9**	49.1
	H_{AVG}/H_s	0.64	0.64	**1.13**	0.81	0.9	**1.1**	1.65
Grid	R_{AVG}%	61.89	49.7	**93.7**	87.55	52.9	**99.0**	89.5
	H_{AVG}/H_s	0.81	0.78	**1.3**	0.95	0.7	**1.07**	1.05
Circle W holes	R_{AVG}%	43.5	39.7	**90.5**	84.02	75.5	**96.9**	93.8
	H_{AVG}/H_s	0.6	0.69	**1.6**	1.01	0.9	**1.21**	2.46
Building	R_{AVG} %	45.7	40.8	**95.4**	80.14	59.5	**98.7**	97.3
	H_{AVG}/H_s	0.65	0.62	**1.39**	1.06	0.8	**1.18**	1.43
odd NW	R_{AVG} %	70.7	66.0	**93.7**	99.92	93.0	**100**	94.4
	H_{AVG}/H_s	0.88	0.8	**1.3**	1.03	1.00	**1.00**	1.2

B. *Improvement in existing routing schemes*

Having considered the effectiveness of the proposed anchor placement strategy on GF, the next question is whether the existing VC-based routing schemes benefit by the ENS-based anchors. Thus we evaluate the performance of three existing routing schemes, namely Convex Subspace Routing (CSR) [36], Logical Coordinate Routing (LCR) [23], and Directional Virtual Coordinate Routing (DVCR) [41]. A comparison is done with ENS-based anchor selection vs. five randomly selected anchors in terms of average routability ($\mathrm{R_{AVG}}$%) and average path length ($\mathrm{H_{AVG}}$) defined as

$$\mathrm{R_{AVG}}\% = \frac{\text{Total \# of packet that reached the destination}}{\text{Total number of packet generated}}\% \quad (6)$$

$$\mathrm{H_{AVG}} = \frac{\text{Cumilative number of hops that each packet traversed}}{\text{Total number of packet generated}} \quad (7)$$

Average routability evaluation considers all source-destination pairs, i.e., each node generated a set of (N-1) messages, with one message for each of the remaining nodes as the destination. For random anchor placement, $\mathrm{R_{AVG}}$% and $\mathrm{H_{AVG}}$are averaged over 10 different anchor placement configurations. H_s is the shortest hop distance from source to destination, i.e., ideal path length. The routabilities for networks in Figure 7.5 are tabulated in Table 7.2 for random and ENS-based anchor placements. Since CSR requires a minimum of three anchors, for the spiral network, two initial randomly selected anchors were also used for routing in addition to two anchors selected by ENS. Moreover in our implementation of LCR, perfect backtracking capability is assumed by saving entire path traversed by each packet at the intermediate nodes. Thus LCR routability reported is more optimistic than that a practical implementation would yield.

Not only do the ENS-selected anchors improve the routability of the existing schemes, it also finds shorter paths (see Table 7.2). Routability improvement of CSR is on average 28.4% while that of LCR is 21%. DVCR is improved by 5%. With the anchors from ENS, DVCR outperforms geographical routing scheme, Greedy Perimeter Stateless Routing (GPSR) [69], by 13.4% averaged over all the networks. Unlike GPSR, which requires expensive localization, DVCR relies solely on logical coordinates.

C. Parameter tuning in ENS

The number of anchors chosen depends on the neighborhood size h. For instance, consider the building network in Figure 7.5 (d) where ENS identifies six anchors when a 4-hop neighborhood (h=4) is used. For an 8-hop neighborhood ($h = 8$), ENS identifies only three anchors, which are marked as A, B, and C in Figure 7.5 (d). The performance of A, B, and C is more or less the same as that with all the ENS anchors. For instance, average routability of DVCR $R_{AVG}\%$ using A, B, and C is 98.8% with 1.23 H_{AVG}/H_s. The main conclusion is 4-hop neighborhood provides a set of anchors with good performance, but there exists fewer number of anchor configuration, which provides more or less the same performance under proper choice of h.

7.6 Related Work: Topology Preserving Maps

This section briefly discusses the topology preserving map generation proposed in [37]. Consider a WSN with N nodes, with each node characterized by a vector of VCs, with distance to each of the M anchors ($N >> M$). Let P be the $N \times M$ virtual coordinate matrix of the network. Singular value decomposition of P is

$$P = U.S.V^T \tag{8}$$

where, U, and V are $N \times N$, and $M \times M$ unitary matrices respectively and S is a $N \times M$ matrix where elements $S(i,i)$ are called singular values. Each node thus can be represented by its Principal Components (PC), given by P_{SVD}:

$$P_{SVD} = P.V \tag{9}$$

The second and third columns of P_{SVD} provide a set of 2D Cartesian coordinates for node positions on a topology preserving map, i.e.,

$$[X_T, Y_T]_{(i)} = [{P_{SVD}}^{(2)}, {P_{SVD}}^{(3)}] \tag{10}$$

where ${P_{SVD}}^{(i)}$ is i^{th} column of P_{SVD}, and $[X_T, Y_T]_{(i)}$ is the topological coordinate pair of i^{th} node. Topology Preserving Error (E_{TP}) is defined as the % of nodes out of order/flipped compared to the original network. Results in Section 7.7 show that the anchors from ENS significantly improve the topology preserving maps proposed in [37] in terms of E_{TP}.

7.7 Topology Preserving Maps from Directional Virtual Coordinates

Consider the DVCS of a full rectangular grid with two anchors A_j and A_k in Figures 7.7 (a) and (b). The virtual directions of $\vec{\boldsymbol{u}}_{\boldsymbol{A_j A_k}}$ for different *j* and *k* are indicated by the arrows. As each DVC is generated based on two anchors, it is possible to transform the *M* virtual coordinates to up to C_2^M DVCs. In practice, however, only a small number of anchor pairs are needed to characterize a network in DVC domain [41].

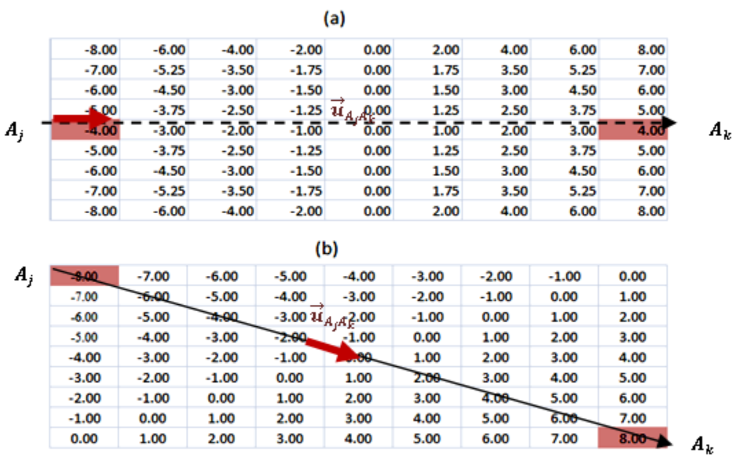
A. Angle between directional coordinates

As a DVCS contains multiple coordinates pointing in different virtual directions, it is useful to define the angle between different directions. Let P_D be the $N \times \breve{M}$ matrix containing the entire DVCS, where *N* is the number of nodes in the network and $\breve{M} \le C_2^M$ is the cardinality in DVC domain. The column $P_D^{(i)}$ corresponds to the ith coordinate. The kth row $(P_D)_{(k)}$ represents the DVC tuple of node n_k. Consider two coordinates in DVCS, $P_D^{(i)}$ and $P_D^{(j)}$ of the entire network. The angle between ith and jth coordinates is defined as

$$\Phi_{ij} = \Phi_{ji} = \cos^{-1}\left(\left(P_D^{(i)}\right)^T P_D^{(j)} / |P_D^{(i)}||P_D^{(j)}|\right) \tag{11}$$

where $|P_D^{(i)}| = \sqrt{\left(P_D^{(i)}\right)^T P_D^{(i)}}$. Consider the example network in Figure 7.8 in which eight anchors are selected in pairs to form four DVCs. The ith row of P_D is,

$[f\left(h_{n_iA_1}, h_{n_iA_2}\right), f\left(h_{n_iA_3}, h_{n_iA_4}\right), f h_{n_iA_5}, h_{n_iA_6}, f\left(h_{n_iA_7}, h_{n_iA_8}\right)]$, thus columns one through four correspond to four virtual directions $\vec{\boldsymbol{u}}_{\boldsymbol{A_1A_2}}$, $\vec{\boldsymbol{u}}_{\boldsymbol{A_3A_4}}$, $\vec{\boldsymbol{u}}_{\boldsymbol{A_5A_6}}$,and $\vec{\boldsymbol{u}}_{\boldsymbol{A_7A_8}}$. The angles between different pairs of columns of P_D from (11) are Φ_{12}, Φ_{13}, and Φ_{14} are 46.22^0, 90^0 and 133.77^0 respectively. If two of these vectors are orthogonal, they can be used to generate a 2D topology map of the network. For instance in Φ_{13}, in Figure 7.8 corresponding to vectors $\vec{\boldsymbol{u}}_{\boldsymbol{A_1A_2}}$, $\vec{\boldsymbol{u}}_{\boldsymbol{A_5A_6}}$ is 90^0 and so is that between ($\vec{\boldsymbol{u}}_{\boldsymbol{A_3A_4}}$, $\vec{\boldsymbol{u}}_{\boldsymbol{A_7A_8}}$).



(a)

-8.00	-6.00	-4.00	-2.00	0.00	2.00	4.00	6.00	8.00
-7.00	-5.25	-3.50	-1.75	0.00	1.75	3.50	5.25	7.00
-6.00	-4.50	-3.00	-1.50	0.00	1.50	3.00	4.50	6.00
-5.00	-3.75	-2.50	-1.25	0.00	1.25	2.50	3.75	5.00
-4.00	-3.00	-2.00	-1.00	0.00	1.00	2.00	3.00	4.00
-5.00	-3.75	-2.50	-1.25	0.00	1.25	2.50	3.75	5.00
-6.00	-4.50	-3.00	-1.50	0.00	1.50	3.00	4.50	6.00
-7.00	-5.25	-3.50	-1.75	0.00	1.75	3.50	5.25	7.00
-8.00	-6.00	-4.00	-2.00	0.00	2.00	4.00	6.00	8.00

(b)

-8.00	-7.00	-6.00	-5.00	-4.00	-3.00	-2.00	-1.00	0.00
-7.00	-6.00	-5.00	-4.00	-3.00	-2.00	-1.00	0.00	1.00
-6.00	-5.00	-4.00	-3.00	-2.00	-1.00	0.00	1.00	2.00
-5.00	-4.00	-3.00	-2.00	-1.00	0.00	1.00	2.00	3.00
-4.00	-3.00	-2.00	-1.00	0.00	1.00	2.00	3.00	4.00
-3.00	-2.00	-1.00	0.00	1.00	2.00	3.00	4.00	5.00
-2.00	-1.00	0.00	1.00	2.00	3.00	4.00	5.00	6.00
-1.00	0.00	1.00	2.00	3.00	4.00	5.00	6.00	7.00
0.00	1.00	2.00	3.00	4.00	5.00	6.00	7.00	8.00

Figure 7.7. DVCS coordinates with respect to anchor pairs: (a) A1, A2, and b) A3,A4, for a 9x9 rectangular grid. Arrow indicates the direction of the unit vector.

Some of the angles between directional coordinates of Figure 7.9 (a) are tabulated in Table 7.3. Anchors are selected using ENS. For example anchors A, C, and D provide three directional coordinates where the directions intersects and form a triangle. The addition of the angles $\emptyset_{AD,DC}$(94.9^0), $\emptyset_{AD,AC}$(16.7^0) ,$\emptyset_{CD,AC}$ (68.4^0) is ~180^0.

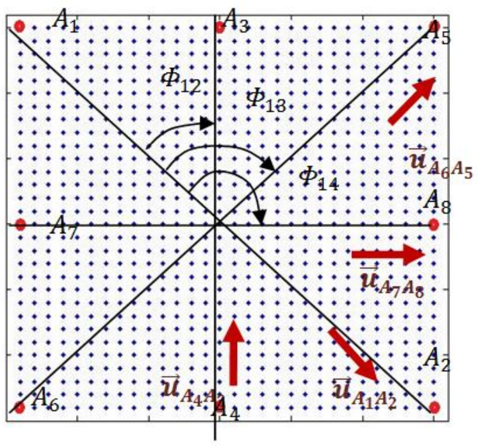


Figure 7.8. Example to illustrate the angles between different unit vectors $\vec{\boldsymbol{u}}_{\boldsymbol{A_1A_2}}$, $\vec{\boldsymbol{u}}_{\boldsymbol{A_3A_4}}$, $\vec{\boldsymbol{u}}_{\boldsymbol{A_5A_6}}$,and $\vec{\boldsymbol{u}}_{\boldsymbol{A_7A_8}}$

B. *In-Network realization of angle estimation*

Angle evaluation in (11) results in the angle between two datasets corresponding to two selected directional coordinates. If a subset of data points, which is a good representation of the entire data set, is selected for two coordinates, still the angular evaluation should hold. Thus anchors VCS, $Q_{M\times M}$ can be used for estimation of angles between two directions. First each node evaluates the DVC matrix Q_D corresponding to Q. ith column of $Q_D \equiv Q_D^{(i)}$ is a subset of data points of ith coordinate of DVCS $P_D^{(i)}$. Therefore, the approximate angle between ith and jth coordinates can be evaluated as,

$$\breve{\Phi}_{ij}=\breve{\Phi}_{ji}=\cos^{-1}\left(\left(Q_D^{(i)}\right)^T Q_D^{(j)}/|Q_D^{(i)}||Q_D^{(j)}|\right) \tag{12}$$

where $|Q_D^{(i)}| = \sqrt{\left(Q_D^{(i)}\right)^T Q_D^{(i)}}$. The error of this approximation is defined as

$$E_\phi = \sum_{\forall i,j} |\check{\phi}_{ij} - \phi_{ij}|/\phi_{ij}$$

E_ϕ for Figure 7.9 (a) is 0.08 and for Figure 7.5 (b) is 0.066, indicating that the in-network realization provides sufficiently accurate results for practical purposes. Worst case transmission complexity of informing the anchors VCS to the entire network is $O(MN)$, which is required by proposed DVCS- and SVD-based in-network realization of TPM approaches.

C. DVCS-based Topology Preserving Maps

With the definition of an angle between two DVCS, next we propose a new method for TPM generation based on the angles between different DVCs. The basic concept is that near orthogonal directions of DVCS provide a good set of axis for obtaining a topology map of the network. For example the TPM of the network in Figure 7.9 (a) is shown in Figure 7.9 (b). The map in Figure 7.9 (b) is based on coordinates $(f(h_B, h_D), f(h_A, h_C))$ which has $\emptyset_{DB,AC}$ 86^0 (Table 7.3). The resulting topology preserving error E_{TP} [37] is 1.32%. Figure 7.9 (c) is the TPM using the SVD-based scheme as in [37], but with anchors selected by ENS, while Figure 7.9 (d) is the TPM from SVD on VCS generated using randomly selected 10 anchors, exactly as proposed in [37]. E_{TP} for the former is 0% while the latter is 2.49% averaged over 10 random anchor placement configurations, which clearly indicate the effectiveness of anchor selection of ENS. TPM generated from DVCS has less E_{TP} compared to that with SVD on a VCS with randomly selected anchors.

Similarly TPM for Figure 7.10 (a) generated from near orthogonal DVCs ($\emptyset_{DB,AC} = 94.12^0$), corresponding to anchor pairs (D,B) and (A,C), is shown in Figure 7.10 (b). With an E_{TP} of 0.407%, it achieves better accuracy than the TPM generated from SVD with random anchor placement described in [38] with E_{TP} of 1.5116%, averaged over 10 random anchor placement configurations (see Figure 7.10(d)). Again, the ENS-based anchor selection improves the TPM generated with SVD-based approach, as in Figure 7.10(c), reducing E_{TP} to 0.1279%.

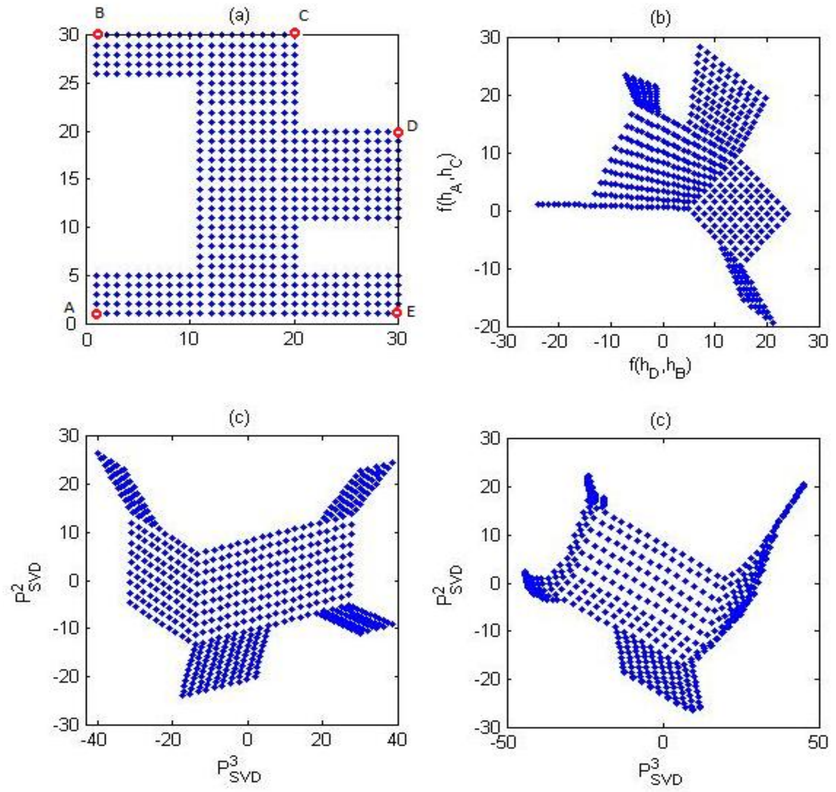


Figure 7.9. (a) Odd network. Anchors identified by ENS are indicated in red, (b) TPM using near orthogonal DVCs;, (c) SVD based TPM using the VCS generated using anchors identified by ENS, and (d) SVD based TPM using the VCS generated using 10 randomly placed anchors.

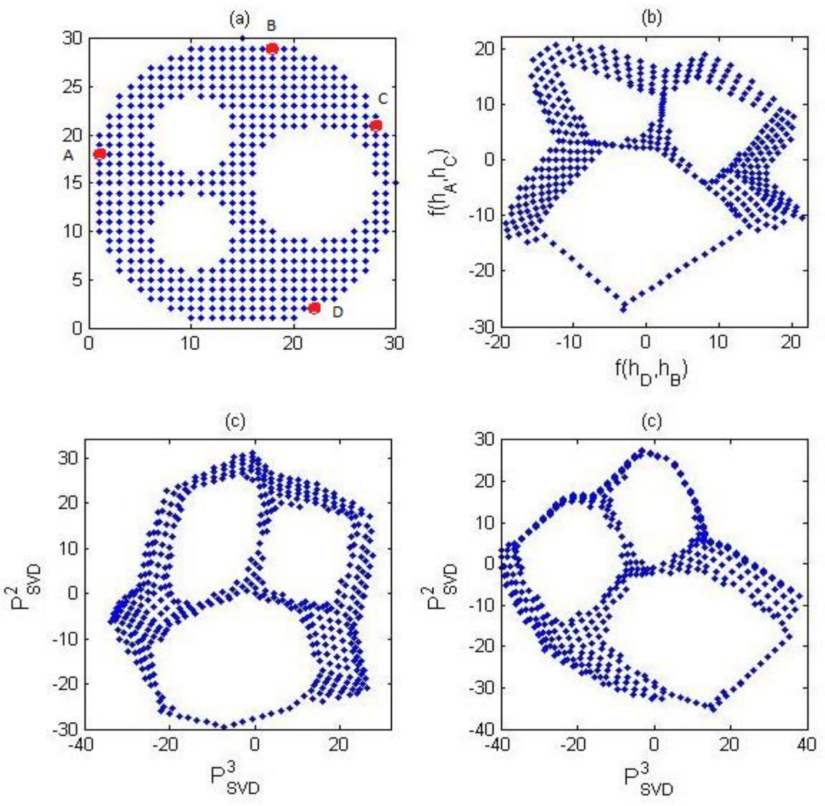


Figure 7.10. (a) Circular network with voids. Subset of anchors identified by ENS are indicated in red, (b) TPM using near orthogonal DVCs, (c) SVD based TPM using the VCS generated using anchors identified by ENS, and (d) SVD based TPM using the VCS generated using 10 randomly placed anchors.

Table 7.3
Some of the angles between coordinate-pairs in Figure 7.9(a)

Ø^0	AC	AB	AE	AD	CB	CE	CD	BE
AB	39.8	-	-	-	-	-	-	-
AE	76.2	101.8	-	-	-	-	-	-
AD	16.7	48.5	64.1	-	-	-	-	-
CB	103.3	63.5	130	109.5	-	-	-	-
CE	152.7	150.0	76.4	138.8	94.2	-	-	-
CD	111.6	121.4	54.0	94.9	106.0	49.7	-	-
BE	124.6	158.7	56.8	114	128.4	34.2	48.4	-
BD	86.0	124	41.5	75.3	161	73.5	54.7	41
ED	21.9	31.9	94.4	30.3	90.4	165	115	139

Performance of both TPM generation techniques highly depends on the anchor selection and the number of anchors. Hence, the ENS reduces the transmission, memory, and computational complexities of both TPM approaches, providing impressive E_{TP}.

7.8 Conclusion

Performance of virtual coordinate-based algorithms depends heavily on the number of anchors and their placement. We proposed a novel strategy that identifies the extreme nodes in a network as anchors, and thereby minimizes the impact due to local minima problem. Many of these extreme nodes lie on the boundary and are also spaced far apart. Performance evaluation demonstrates that ENS results in significant improvement in performance in terms of Greedy Ratio in Greedy Forwarding with an order of magnitude less anchors compared to randomly placed anchors even on the boundary. It also significantly improves the routability and path lengths of existing VC domain routing schemes, Convex Subspace Routing (CSR), Logical Coordinate Routing (LCR) and Directional Virtual Coordinate Routing (DVCR). Fascinatingly, with the proposed anchor selection, DVCR outperforms geographic routing scheme Greedy Perimeter Stateless Routing (GPSR), in spite of latter’s use of node location information. Properties of DVCS domain were investigated by introducing a vector notation. By selecting two directions that are orthogonal to each other, we demonstrated the creation of a topology map of the network. The new DVC-based approach for topology map generation involves significantly lower computation complexity compared to existing schemes.

An advantage of nodes knowing TPM and topology coordinates is that even when some nodes, including anchors, fail in the network, the topological coordinates need not be recalculated. Topological coordinates approximate the physical layout coordinates, albeit with distortions to take into account connectivity. Physical coordinates do not depend on the existence of other nodes, while traditional virtual coordinates have to be recalculated to maintain their validity. Topological coordinates, while initially calculated based on virtual coordinates, maintain a valid topological representation even when a moderate number of nodes fail. Another consequence of the work presented in this research is the ability to specify cardinal directions and use angles in VC domain. This is a radical change from the traditional VC system approaches, which we believe will significantly enhance the ability to exploit VCs for networking.

CHAPTER 08

GEO-LOGICAL ROUTING: UNIFIED VIRTUAL AND TOPOLOGICAL SPACE ROUTING

8.1 Introduction

Algorithms for self-organization and data dissemination among sensor nodes play a crucial role in performance and lifespan of large-scale Wireless Sensor Networks (WSNs). Routing techniques for data dissemination/fusion can be broadly categorized as content-based routing and address-based routing [23]. The former uses content-based attributes in the packet to define the destination set [35], while the latter uses some sort of position information, physical or virtual, to identify or reach the nodes. Physical domain schemes rely on location or physical position information [14] obtained using localization algorithms or GPS. Equipping nodes with GPS is costly and infeasible for many applications, with indoor WSNs and underwater WSN being examples. Localization based on parameters such as RSSI is error-prone and difficult to use in large-scale networks and crowded terrains. If location information is available, Geographic Routing (GR) schemes (also called position-based routing or Geometric Routing) [49][65][81] can be used. GR possesses the advantage of having directional information, but its performance is highly correlated with localization errors [101]. GR also suffers from dead-end problems, also known as local minima problems, due to physical voids. Local minima problems occur when the node currently holding the packet does not have a neighbor closer to the destination than itself to forward the packet.

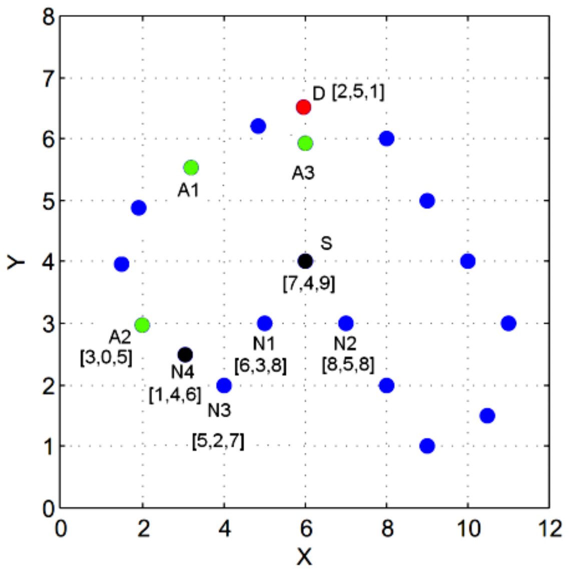


Figure 8.1. An example physical map to illustrate the effectiveness of combining physical and virtual (connectivity) information for routing.

Virtual Coordinate-based Routing (VCR) attempts to overcome the disadvantages of physical domain schemes by characterizing nodes using connectivity-based geodesic distance (in terms of hops), instead of position information and corresponding Line of Sight (LoS) distance. In VCR, a subset of nodes is selected as anchors [22][23][36][82]. These selected anchors, in contrast to anchors in localization schemes, do not have any additional features than the rest of the nodes. Each node is characterized by a VC vector consisting of minimum hop distances to each of the anchors. Like GR, most VCR schemes also use Greedy Forwarding (GF), where the packet is forwarded to the neighbor that is closest to the destination with respect to some distance function evaluated in virtual domain. Distance is typically calculated using L1 or L2 norms. Although VCs have the connectivity information embedded in the ordinates, the cardinal directionality information (north, east, X, Y, etc.) is lost. Physical voids become transparent in virtual domain. However, the local minima problem still arises in the virtual domain [36]. The problem is exacerbated due to over/under deployment and improper placement of anchors. Identification of the optimal number of anchors and proper anchor placement remain major challenges for VCR.

A. Problem definition

Many difficulties associated with Virtual Coordinate System (VCS) based schemes are attributable to lack of information about the physical network. Layout information such as physical voids and relative positions of sensor nodes with respect to X-Y directions are absent. Even though VCS is based on connectivity, explicit information on hop distances between pairs of nodes is not available and is difficult to estimate. Absence of connectivity information, on the other hand, is the cause for local minima problems in physical domain routing. Combining connectivity-based information in VCS and position or direction information in a network essentially would combine the advantages of VCR and GR schemes overcoming the disadvantages in each other's domain. Consider the simple example network in Figure 8.1, with source node S and destination D. If physical coordinates are used for LoS distance evaluation, neighbors N1 and N2 of S have higher distances to D than S. Thus, there is a local minima at node S in physical domain. Nevertheless, based on the VCS generated using anchors A1 and A2, virtual distance from N1 to D is less than that of S to D. Thus, the minima in the physical domain can overcome using VCS. Similarly, when the packet is at N4, using VCS-based GF in VC domain, N4 fails to find a closer neighbor to destination than N4 itself. In contrast, based on the physical information N4 can identify A2 as its neighbor, which is closer to D than it is. This simple example illustrates cases where local minima in one domain can be overcome by using information from the other domain. However, simply using physical coordinates to supplement virtual coordinates is not an acceptable option, as it would entail inheriting all of the disadvantages associated with obtaining physical coordinates, which the virtual coordinates were used to get rid of at the first place. With the above background, we define our problem as:

Given a network with a virtual coordinate system P wrt M anchors,

1) Generate appropriate substitution for the physical information, i.e., topological information, without using any means of physical information.
2) Merge the virtual and topological information to achieve high data dissemination performance at least as in physical coordinate routing with low energy consumption.

B. Contribution

This research proposes a family of routing schemes called Geographic-Logical Routing (GLR), which combines the topological coordinates that solely derived from VCS and VC information to achieve significantly high performance in routability.

The proposed novel approach uses topology coordinates (TCs) that are derived using existing Topology Preserving Map generation technique [37], derived using virtual coordinates instead of geographic information. This way, we are able to merge the advantages of physical and logical routing schemes without inheriting their disadvantages. No such routing scheme currently exists. Geographic-Logical Routing scheme transforms the set of VCs to a set of Cartesian coordinates (topological coordinates) [37] and switches between Geographic Routing based on topological coordinates and Logical Routing that uses virtual coordinates to achieve high routability performance. We demonstrate that the topology coordinates and maps are actually more useful than geographic coordinates and maps for WSN routing. TCs inherit the connectivity information; thus, TCs-based next node selection to forward the packet is more accurate than that of the geographic coordinates.

Switching between TCs and VCs to overcome local minima can be performed in various ways. We present a family of routing schemes, α-GLR, where a node selects either TC or VC mode for next node selection with probability α and change the mode overcoming local minima. Performance evaluation of two such schemes, 0-GLR and 1-GLR, are presented. Performance evaluation, in terms of routability, average path length, and energy consumption in various complex network shapes demonstrate that the proposed GLR schemes, 1-GLR and 0-GLR, clearly outperform two existing VCR schemes, namely Logical Coordinate Routing (LCR) [23] and Convex Subspace Routing (CSR) [36]. Moreover, GLR achieves higher routability than the geographic routing (GR) scheme, Greedy Perimeter Stateless Routing (GPSR) [69] in all but one of the networks considered, without the need for localization/GPS. Results also show that with strategic anchor placement, e.g., with Extreme Node Search (ENS) algorithm, higher routability than that with randomly selected anchors can be achieved with very few anchors.

Section 8.2 reviews related routing protocols. Characteristics of topology maps and its effectiveness in GF are discussed in Section 8.3. GLR scheme is proposed in Section 8.4. Section 8.5 evaluates the performance of GLR and Section 8.6 concludes the work.

8.2 Related Work

In this section, existing work on geographic routing and virtual coordinate routing is briefly discussed.

A. Geographic routing

Greedy Perimeter Stateless Routing (GPSR) [69] is a GR scheme that employs GF until a packet comes across a local minima. Then the algorithm recovers local minima issue by routing along the perimeter of the void using right-hand rule, where packets are routed counterclockwise along the edges of voids. There are many topologies for which the right-hand rule does not work or is less efficient than its dual, the left-hand rule. The difficulty is that the information to make the proper decision is not in generally available at the nodes. GEAR (Geographical and Energy Aware Routing) uses GF with an associated cost for each node to allow the packet to be forwarded around holes and also to distribute the routing load among the nodes. Compass routing [67] routes packets out of local minima by routing along the faces intersected by the line segments between the source and the destination. To avoid loops in face routing, a planar graph of the original network graph is required. A two-tier geographic routing protocol presented in [103] initially employs a modified version of GF where a packet is passed to the neighbor that is closest to destination, without comparing it with current distance to destination, to get out of a local minima. To avoid looping, a node that has a single neighbor marks itself as a blocked node. Blocked nodes do not participate in packet forwarding. Path Vector Face Routing in [77] is simply a GF scheme based on face information.

A method to locate and bypass holes is proposed in [49], which develops a local rule, named TENT rule, so that each node can ascertain whether it is a local minima. A distributed algorithm, BOUNDHOLE, helps packets get out of the stuck nodes. In Hole Avoiding In advance Routing protocol (HAIR) [65], the data packet attempts to avoid meeting local minima in advance. Protocols in [49] and [65] are associated with high memory requirements and transmission costs, as this identification of local minima has to be repeated for each destination. Moreover, a local minima on the way to one destination need not to be a local minima for another, thus local minima node information has to be stored separately for each destination. Reference [123] proposes Detour Routing based on Quadrant Classification (DRQC), to reduce the occurrence of the local minima problem by avoiding sending packets to the nodes which could have the local minima problem. In DRQC, each node knows the physical positions of its 2-hop neighbors, and then each node decides its state: negative for local minima problem or positive for local minima problem.

Geographic routing schemes [57] rely on knowledge of physical location of sensor nodes. Equipping nodes with GPS increases cost, energy consumption, and complexity. An alternative is to use a localization algorithm to estimate relative physical coordinates [14]. The accuracy of both central and distributed implementations of localization is highly sensitive to channel fading and signal to noise ratio (SNR). Localization errors occur in the distance estimation and the position calculation. A study of propagation of errors in localization in [91] demonstrates how routability and effectiveness of geographic routing scheme, GEAR [103], deteriorates with localization errors. As both VCS and topology coordinates used by GLR presented in this research are based on hop distances, it is not affected by fading or signal strengths. Further, they do not rely on analog measurements such as RSSI or time delay, and thus do not have cumulative errors that affect the performance, as networks scale.

B. *Virtual Coordinate Based Routing*

VC-based routing (VCR), also referred to as logical routing, has received much attention recently as it is more feasible in WSNs, insensitive to localization errors, and is capable of achieving high routability without the cost and complexity associated with localization. Some representative VC-based routing protocols are discussed next.

Scalable coordinate-based routing algorithm [101] uses a set of perimeter nodes as anchors. GF is used until a local minimum is reached, and then an expanding ring search is performed until a closer node is found or Time-To-Live (TTL) expires. In Virtual Coordinate assignment protocol (VCap) [22], a virtual space is generated for the entire network using three farthest apart anchors. An insufficient number of anchors causes the problem of nodes having identical coordinates. As a solution, a packet is delivered to a zone of nodes with identical coordinates, and then the final destination is sought using a proactive ID-based approach. Logical Coordinate based Routing (LCR) [23] uses GF followed by a backtracking scheme based on furthest apart anchor placement. It requires each node to keep the history of recent nodes that the packet visited so that it can backtrack.

Aligned Virtual Coordinate System (AVCS) [82] modifies the node's VCs by replacing it with the average of itself and neighboring nodes' coordinates as a solution to logical voids. Spanning Path Virtual Coordinate System (SPVCS) in [84] uses a single anchor, and coordinates are created based on the depth-first search algorithm starting from root node. Placing the anchor or the root node at the center gives the best performance, as it can provide a balanced spanning tree. The Axis-Based Virtual Coordinate Assignment Protocol (ABVCap) [113] is a method of constructing a VCS where each node is assigned a 5-tuple virtual coordinate corresponding to longitude, latitude, ripple, up, and down. An improved

ABVCap [113], Axis Based Virtual Coordinate Assignment Protocol (ABVCap_uni) [81] for WSNs with unidirectional links introduces an eight tuple coordinate vector with entries: longitude, latitude, ripple, up, down, ring-initiator, ring-number, and ring-order of each node. Performance evaluation of ABVCap_uni shows that delivery rate increases as the number of nodes increases and that average path length is lower than that of ABVcap. Transmission complexity increases as the new coordinates are generated in both ABVCap_uni and ABVCap. Convex Subspace Routing (CSR) protocol in [36] uses a fundamentally different approach from others by dynamically moving to different convex subspaces of VCS to avoid local minima. A triplet of anchors is used at a time to define the convex subset, which is used for GF until a local minima occurs. Then a different triplet is selected to move to a different subspace without a local minima at current location.

Performance of VCS-based algorithms is highly sensitive to a number of anchors and their placement. Extreme Node Search (ENS), a novel and effective anchor placement scheme proposed in [40], demonstrates significantly improved performance over state-of-the-art. ENS starts with two randomly placed anchors and then uses a Directional Virtual Coordinate (DVC) transformation [40], which restores the lost directionality in traditional VCS, to identify anchor candidates in a completely distributed manner. Note that DVC transformation is purely distributive and does not require any additional communication or information.

Radial propagation of anchors causes the local minima and identical coordinates. No scheme has been proposed, so far, in which advantages of GR and VCR are combined to overcome each other's weaknesses. This can be attributed to the fact that incorporating geographic coordinates incur a significant cost as discussed above in Subsection 8.2.A. The GLR routing scheme presented below overcomes this and is able to go back and forth between virtual and physical modes as the packet encounters local minima in one domain. Proposed routing scheme is compared with two VCR schemes, CSR [36], LCR [39], and the GR scheme GPSR [69] in Section 8.4.

Table 8.1
Notations Used in Chapter 8

Notation	Description
N	Number of network nodes
$N_i \in N$	Node i (current node)
N_d	Destination
M	Number of anchors
$\{A_i \mid A_i \in N \ \forall i = 1{:}M, M \ll N\}$	Anchor set
A_c	Anchor which is closest to the destination
h_{ij}	Minimum hop distance between node N_i and N_j
$P_{N\times M} = [h_{iA_j}]_{ij}$ $i = 1{:}N \ and \ j = 1{:}M$ $([P_{SVD}]_{N\times 2} = [X_T, Y_T])$	Virtual Coordinate Set (Topological Coordinate Set) of the entire network
$P^{(i)}$- ith column of P	Ordinate w.r.t. anchor A_i in VCS
$P_{(i)} = [h_{iA_1} \dots h_{iA_j} \dots h_{iA_M}]$ ith row of P	Node N_i*'s* VC
$[X_T, Y_T]_{(i)} = [X_{T,i}, Y_{T,i}] = [P_{SVD}]_{(i)}$	Node N_i*'s* 2D topological coordinate
$G_{N_iN_j}$	Distance between N_i and N_j in virtual domain
$D_{N_iN_j}$	Distance between N_i and N_j in 2D topological domain
K	Set of neighbor nodes
A_c	Anchor closest to the destination
$N_{i,Prev}$	Node that forwarded the packet to current node
$N_{i,Next}$	Node that current node forwards the packet to

8.3 Topological Coordinates from Virtual Coordinate System

Proposed Geo-Logical Routing (GLR) first extracts a set of 2D topology coordinates suitable for geographic routing from the VCs. The topology coordinates contain the information such as relative positions of nodes, shapes of boundaries and voids, etc. GLR, therefore, switches between topological and virtual coordinate spaces to overcome physical/logical voids in each domain, thus achieving enhanced routability compared to both VCR and GR schemes that it combines.

A. Topology Preserving Maps[37]

Consider a WSN with N nodes, with each node characterized by a vector of VCs, with the hop distance to M anchors ($N >> M$). VC of node N_i is therefore [$h_{iA_1}, h_{iA_2}, \dots, h_{iA_M}$], where h_{iA_j} is the minimum hop distance between node N_i and anchor A_j. Notations used are summarized in Table 8.1.

Let P be the $N \times M$ virtual coordinate matrix of the network. Singular Value Decomposition [72] of P is

$$P = U.S.V^T \qquad (1)$$

where U, S, and V are $N \times N$, $N \times M$, and $M \times M$ matrices respectively. U and V are unitary matrices, i.e., $U^T U = I_{N\times N}$ and $V^T V = I_{M\times M}$. S is a diagonal matrix with singular values as its diagonal elements arranged in descending order. SVD extracts and packages the salient characteristics of the dataset P, providing U as an optimal basis for P. Moreover V is an optimal basis of P^T.i.e. V spans $\mathcal{R}^M$. Then, $U.S$ give the coordinates for the data P under the new basis V. Elements in S provides unequal weights on columns of U. Each node thus can be represented by its Principal Components (PC), given by P_{SVD}:

$$P_{SVD} = P.V = U.S \qquad (2)$$

The set of VCs have the connectivity information embedded in it, though it loses directional information. All the nodes that are h_{A_j} hops away from the A_jth anchor have h_{A_j}

as the jth ordinate. As each ordinate propagates as concentric circles centered at the corresponding anchor, the angular information is completely lost. The first principle component, i.e., the first column of $\mathrm{P_{SVD}}$, contains this dominant radial information. As SVD provides an orthonormal basis, 2nd and 3rd ordinates are orthogonal to 1st ordinate while being perpendicular to each other. Thus, the second and third columns of $\mathrm{P_{SVD}}$ provide a set of 2D Cartesian coordinates, $[\mathrm{X_T}, \mathrm{Y_T}]$, for node positions on a topology preserving map (TPM), i.e.,

$$[X_T, Y_T]_{(i)} = [{P_{SVD}}^{(2)}, {P_{SVD}}^{(3)}]_{(i)} = [P_{(i)}.V^{(2)} \quad P_{(i)}.V^{(3)}] \tag{3}$$

where, ${P_{SVD}}^{(i)}$ is i^{th} column of P_{SVD} and $V^{(i)}$ is i^{th} column of V. Moreover $[X_T, Y_T]_{(i)}$ and $P_{(i)}$ are the topological coordinate and virtual coordinate of i^{th} node respectively. A detailed analysis of this method for obtaining TPMs is provided in [37]. In fact, instead of using the full $N \times M$ coordinate set for P, only the coordinates of the anchors ($M \times M; M \ll N$), or a small subset of nodes, ($R \times M; R \ll M$), can be used for the calculation of the topological coordinates. As illustrated in [37], it is possible to obtain a good approximation for V in (2) based only on the coordinates of the set of anchors or those of a small set of random nodes, giving rise to several pragmatic implementations in sensor networks [37]. Each node essentially needs to be provided with first three columns of V for the computation; therefore a computationally less complex approach based on eigenvalue decomposition is explained in [37].

In many ways, the TPM is a better candidate for geographic routing than the original physical map, as the former is based on actual connectivity information rather than the node position. Another advantage of topological domain over the physical domain is that in topological domain VC information is also available. A comparison between physical information and topological information in Greedy Forwarding (GF) is discussed next (Subsection B).

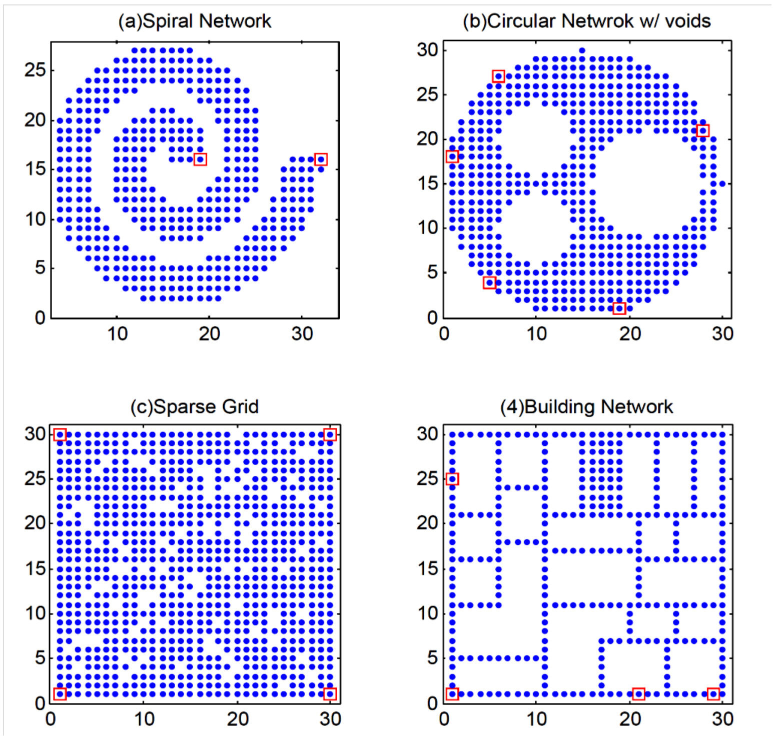


Figure 8.2. (a) Spiral shaped network of 421 nodes, (b) Circular network of 496 nodes with three voids, (c) A 30 by 30 node grid with 100 randomly missing nodes, and (d) Network in a building with 343 nodes. Red circles indicate anchors with ENS anchor placement.

B. Effectiveness of physical and topology-based Cartesian coordinates in packet forwarding

We already asserted that, in many ways, the TPM is a better candidate for geographic routing than the original physical map, as the former is based on actual connectivity information rather than the node position. A set of coordinates is good for routing if it results in accurate forwarding decisions. In other words, how accurately does the distance evaluation identify a closer neighbor (termed as 'correct' neighbor in (10)) to the destination than the current node? This can be quantitatively evaluated using

$$\mathrm{P[Selecting\ correct\ neighbor]} = \sum_{\mathrm{N_j \in N}} \sum_{\mathrm{N_i \in N}} \frac{\substack{\text{\# number of times a } \mathrm{N_j} \text{selected correct neighbor} \\ \text{to FWD the packet when destination is } \mathrm{N_i}}}{\text{Total \# nodes (N)}} \quad (10)$$

Several factors have to be taken into account to understand the significance of topological coordinates, including the fact that it is not just a substitute for physical coordinates. Topological coordinates are a significantly better option for routing and self-organization in terms of cost as well as performance. Four representative benchmark sensor-nets [21] ranging from 421 to 800 nodes shown in Figure 8.2 are used for performance evaluation. Each node has maximum four neighbors while the communication range is unity. Topology Map depends on the anchor placement. Thus random anchor placement and existing anchor placement called Extreme Node Search (ENS, briefly explained in Subsection 8.2.B) are used in the comparison. ENS identifies farthest apart corner nodes, if any, as anchors as indicated by highlighted nodes in Figure 8.2. In random anchor placement, the number of anchors, M, varied from 5 to 20.

Table 8.2
Probability of Selecting the Correct Neighbor Based on Topology Coordinates and Physical Coordinates for the Networks in Figure 8.2

	Probability of Selecting Correct Neighbor					
Network Topology	Topology Coordinates					Physical Coordinates
	M=20	M=15	M=10	M=5	ENS**	
Spiral	0.66	0.63	0.63	0.65	0.68	0.43
Circle with voids	0.64	0.61	0.62	0.52	0.60	0.54
Sparse Grid	0.64	0.64	0.6	0.56	0.63	0.61
Building Network	0.84	0.80	0.78	0.72	0.84	0.83

** See Figure 8.2, ENS anchors are circled in red

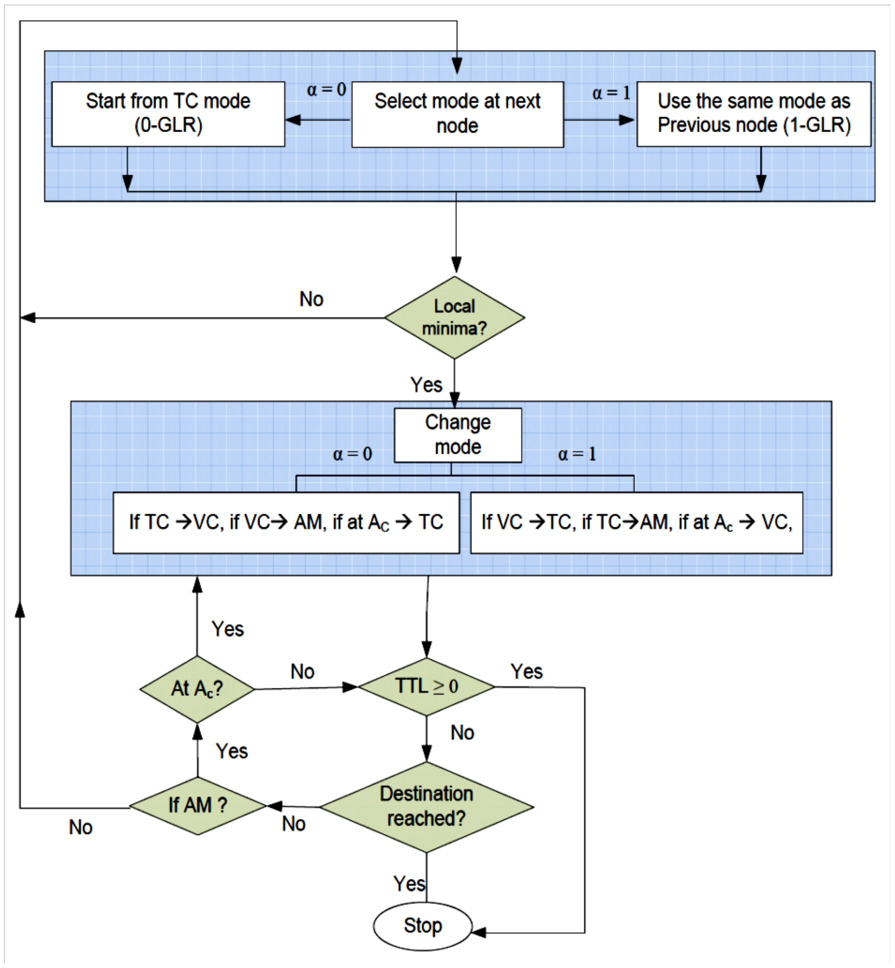

Start from TC mode (0-GLR)
α = 0
Select mode at next node
α = 1
Use the same mode as Previous node (1-GLR)
No
Local minima?
Yes
Change mode
α = 0
α = 1
If TC → VC, if VC → AM, if at A_C → TC
If VC → TC, if TC → AM, if at A_c → VC,
Yes
At A_c?
No
TTL ≥ 0
Yes
No
Yes
If AM ?
No
Destination reached?
No
Yes
Stop

Figure 8.3. Flow chart of Geo-Logical Routing at a node N_i

Table 8.2 shows the probability of a node selecting the correct neighbor to forward the packet based on L^2 distance metric using physical- and topology-based Cartesian coordinates. Topology maps generated with 10 randomly selected anchors have the capability of selecting the correct next neighbor as accurately as with physical coordinates for the networks in Figure 8.2. The results clearly indicate that topology-based coordinates are more effective (or as effective in the worst case), in selecting the correct neighbor to forward the packet in Greedy Forwarding compared to physical coordinates. With strategically placed anchors (highlighted using circles in Figure 8.2), using ENS that demonstrates better performance with as few as two anchors for Figure 8.2 (a), four anchors for 3 (c) and 3 (d), and five for the Figure 8.2 (b). This is a remarkable result, indicating that expensive localization procedures are unnecessary for the purpose of routing packets. Many other self-organization tasks can use topology coordinates instead of geographical coordinates.

The energy consumption in generating the topological coordinates is significantly lower than that in generating physical coordinates. Creating VCs involve a single flooding for each anchor, and each collecting coordinates from a set of small number of random nodes. For instance, generating anchor coordinates for M anchors require M network floods. Extreme Node Search (ENS) [40] provides a small set of nodes, $M < 3\%$, as anchors, thus the cost of flooding is reduced. Physical localization, in contrast, depends on analog measurements. Signal strength measurements require specific hardware capabilities at each node, while time delay requires accurate clock synchronization. Analog measurements have to be repeated to obtain reliable estimates and are susceptible to noise, fading, and even remaining power (battery level). The errors propagate cumulatively with localization algorithms. It has been demonstrated elsewhere that even a small error and ignoring the impact of errors of location information has a drastic effect on routability. For example, with Geographic Routing, GEAR [103], the routability performance falls below 50% when the distances estimation inaccuracy is 6% [101]. Also, note that we have used a regular grid-based placement and perfect location estimates that are very favorable to routing using physical coordinates. These considerations have significantly biased the results in favor of the physical coordinates, and the values are likely to be much less favorable under more realistic conditions.

The next section discusses the proposed routing algorithm, Geo-Logical Routing, which combines the topological information and virtual coordinate information to overcome local minima issue.

8.4 Generalized Physical and Virtual Space Routing: Geo-Logical Routing

Let the topological coordinates of node $\mathrm{N_i}$ be $[\mathrm{X_{T,i}}, \mathrm{Y_{T,i}}]$ and that of the destination $\mathrm{N_d}$ be $[\mathrm{X_{T,d}}, \mathrm{Y_{T,d}}]$. The L^2 topological line-of-sight distance between any node $\mathrm{N_i}$ in the network and the destination is

$$D_{N_iN_d} = \sqrt{\left(X_{T,i} - X_{T,d}\right)^2 + \left(Y_{T,i} - Y_{T,d}\right)^2} \tag{5}$$

As virtual coordinates have the connectivity information embedded in the coordinates, one expects the virtual domain distance estimation to yield the hop-based geodesic distance. A perfect estimate here will result in 100% routability and optimum path length. However, a proper distance metric that consistently provides such an estimate in virtual space is not known. L^1, L^2 and higher norms are typically used [23][36][113]. Here we use L^2, thus the distance between a node and the destination in the virtual domain is estimated by

$$G_{N_iN_d} = \sqrt{\sum_{j=1}^{M}(h_{iA_j} - h_{dA_j})^2} \tag{6}$$

The imperfections of the distance function are due, in part, to the fact that the different ordinates of VC are not orthogonal to each other. This imperfect distance estimate as well as improper anchor placement causes local minima in logical space.

A node identifies itself as a local minima in topological or virtual domain if its distance to the destination is lower than the distances from each of its neighbors to the destination. Distance evaluation is performed based on the current mode of routing. If the packet is routed based on topological coordinates then the local minima is identified by $D_{N_cN_d} < D_{N_iN_d}, \forall N_i \in K,$ where $D_{N_cN_d}$, and $D_{N_iN_d}$ are distance from current node to destination and neighbor(s) to destination based on topological coordinates. If the mode is VC-based routing, then the packet is at a local minima if $G_{N_cN_d} < G_{N_iN_d}, \forall N_i \in K,$ where $G_{N_cN_d}$ and $G_{N_iN_d}$ are distances from current node to the destination and neighbor(s) to the destination based on VCs. Exact destination is identified by unique node IDs. Given that the local minima in physical, topological, and virtual spaces are unavoidable, and that we are able to derive topological information from VCs, the proposed Geo-Logical Routing scheme uses one space to overcome the minima in the other space. At local minima, the node changes the routing domain that it is currently operating in (from virtual to topological and vice versa). On rare occasions, when the node is a local minima in both the domains, we use properties of VCS to escape and travel away from that minima. This third phase of routing involves sending the packet to the anchor closest to the destination. The property of VCS proposed below help achieve this.

Property 1: In a connected network of N nodes, there is a path between any two nodes via any anchor. Furthermore, a packet can be routed from any node to any anchor with 100% routability through the shortest (optimum path).

Proof: Any two nodes having a path via an anchor in a connected network is self-explanatory. Let the ordinate of any node N_i, with respect to the anchor $A,$ be h_A. There exists a neighbor to node Ni who has the ordinate of one less, i.e., $(h_A - 1)$. The new node in turn can find a neighbor a distance $(h_A - 2)$, and so on, resulting in the packet reaching the anchor in h_A hops. QED

The closest anchor to the destination is selected so that the distance from the chosen anchor to the destination is at its minimum, so the possibility of a local minima occurring in the path from the anchor to destination is minimized.

The closest anchor in hop distance to the destination is determined based on the destination's VC. Since the destination's VC is $[h_{d,A_1} \dots h_{d,A_i} \dots h_{d,A_M}]$, the closest anchor to the destination is determined by

$$A_c = \underset{A_i}{\operatorname{argmin}} \, (h_{d,A_i}) \tag{7}$$

The routing scheme switches among three modes: a) TC, which uses topology-based coordinates and $D_{N_iN_d}$ from (5) for distance; b) VC, which uses virtual coordinates and distance function $G_{N_iN_d}$ from (6); and c) AM (Anchor Mode), which routes toward selected anchor closest to the destination using A_c from (7).

Combining TC and VC information creates a family of routing schemes, α-GLR, where α is the parameter that defines the probability of using TC and VC mode. This research discusses 0-GLR and 1-GLR that combine the topological and virtual information in two significantly different approaches. The basic flowchart of 1-GLR and 0-GLR is in Figure 8.3. Routing stops when the packet has reached the destination or the TTL expires in both the algorithms. 1-GLR and 0-GLR illustrates two extremes of selecting TC and VC for routing. In the former, each node that receives a packet uses the mode that the previous packet-forwarding node used with probability one, while in the latter, every node uses TC mode with probability 1 (i.e., keep the previous mode with probability 0). In both the routing schemes proposed, 1-GLR and 0-GLR, every node saves the predecessor and successor nodes' information in order to prevent loops. The packet header contains a field that indicates the current mode of routing (TC, VC, or AM).

A. 1-GLR

The source node initiates routing in TC mode. The packet continues to be routed in this mode until it reaches a local minima in the topology space, at which time the mode is changed to VC mode. Then the packet is routed in VC mode until it encounters a local minima. At a local minima in VC, the mode is changed to TC. Afterwards, if the packet is trapped in a local minima, TC mode is changed to AM in which the packet is sent to the anchor closest to the destination. Once the anchor is reached, the anchor node switches back to VC mode. Algorithm is as shown in Figure 8.4 in Appendix.

B. 0-GLR

Every node initiates next node selection in TC mode. When a node fails to find a closer node based on TC, then the mode changes to VC. If the packet encounters local minima in VC mode as well (at the same node), then the packet is routed using the AM mode in which the packet is sent to the anchor closest to the destination based on corresponding ordinate. Once the anchor is reached, mode is again set to TC. Details of the algorithm are as in A.1 in Appendix.

It is important to note that many more variations of this algorithm are possible for switching among different modes, such as moving only a certain distance toward closest anchor in AM mode before switching, or switching between TC and VC modes more frequently or probabilistically. The particular algorithms considered in this research produce significant performance gains with straightforward switching mechanisms.

Detailed algorithm of 0-GLR and 1-GLR is as Figure 8.4.

```
INPUT: VCs, TCs, of $N_j$; $j \in K$; $N_d$ and $N_j$
OUTPUT: Closest node to destination from $N_j$; $j \in K$
while ($N_d \neq N_i$ || $TTL \geq 0$ )
    if (Mode = = AM )
      if $N_i$ = = $A_c$
         if $\alpha == 1$ { call 1-GLR}
         elseif $\alpha == 0$ { call 0-GLR}
          end
      else
           Send the packet toward the anchor $A_c$ closest to $N_d$
      end
    elseif (Mode = =TC)
         $R_k = D_{jd}$ ; $j \in K$; $N_j \neq N_{i,Prev}, N_{i,Next}$ % Calculate the topological distance from
Neighbors set K to $N_d$ excluding $N_{i,Next}$ and $N_{i,Prev}$
         $d_{jd} = D_{jd}$ %Current distance to desination
    elseif (Mode = = VC)
         $R_k = G_{jd}$ ; $j \in K$; $N_j \neq N_{i,Prev}, N_{i,Next}$ % Calculate the virtual distance from Neighbors
set K to destination excluding $N_{i,Next}$ and $N_{i,Prev}$
         $d_{jd} = G_{jd}$
    end

      if $R_k$= ={} %If there is no neighbor excluding $N_{i,Prev}, N_{i,Next}$
          Set Mode= AM   % Shift to Anchor Mode
      else
         if $Min(R_k) == 0$
           if $N_d == N_i$
              ROUTED
           else % if identical coordinates
              if $\alpha == 1$ { call 1-GLR}
              elseif $\alpha == 0$ { call 0-GLR}
              end
           end
         elseif $Min(R_k) \leq d_{id}$
           $N_{i,Prev}$ = $N_i$
```

```
                $N_i = \underset{j}{argmin} R_k(j)$
                $N_{i,Next} = \underset{j}{argmin} R_k(j)$
                    if $\alpha == 0$ { Set Mode = TC}
                    end
            elseif $Min(R_k) > d_{id}$ %Local minima, change the mode
                    if $\alpha == 1$ { call 1-GLR}
                    elseif $\alpha == 0$ { call 0-GLR}
                    end
            end
        end
    end
end
Function y = 1-GLR()
                switch Mode % current mode at a minima
                        case VC
                            Set Mode=TC % Shift to topology mode
                        case TC
                            Set Mode= AM % Shift to anchor mode
                        case AM
                            Set Mode= VC % Shift to VC mode
                end
Function y = 0-GLR() %Always starts from TC mode
                switch Mode % current mode at a minima
                        case TC
                            Set Mode= VC % Shift to VC mode
                        case VC
                            Set Mode= AM % Shift to anchor mode
                        case AM
                            Set Mode= TC %  Shift to topology mode
                    end
```

Figure 8.4. Pseudo code of GLR algorithms.

8.5 Performance of GLR

The performances of 1-GLR and 0-GLR are evaluated next and are compared with two virtual coordinate-based routing schemes, Logical Coordinate Routing (LCR) [23] and

Convex Subspace Routing (CSR) [36], as well as the geographic routing scheme Greedy Perimeter Stateless Routing (GPSR) [69].

Logical Coordinate Routing [23] scheme has two phases, GF and backtracking, when a local minima is encountered. To support the backtracking phase, each node has to store the information of message ID and at least the immediate neighboring node that the packet was forwarded. In LCR implementation, we assumed that the entire path traversed by the packet is available at each node so that backtracking can be perfectly performed, avoiding any loops. Therefore, the implemented case is the best case of LCR and is not achievable in practice due to the cost involved in transmitting the required information.

Convex Subspace Routing (CSR) [36] selects three anchors at a time for GF. The selected triplet encloses current node and destination, forming a triangular subspace to provide convex distance function from current node to destination. Routing terminates when there is no triplet of anchors to enclose the current node and destination. There are two disadvantages: at least three anchors are required, and anchors should be placed in such a way that triangular subspaces cover the network.

In the implementation of GPSR [69], the packet is routed based on GF until a physical void is met. After that, the node follows the right-hand rule, which selects the node with smallest angle to the line connecting current node and destination in clockwise direction. Then the next node will start routing in GF mode. This is slightly different from the original GPSR scheme [69]. In the original scheme, as a physical void encounters, the mode of the packet changes to right-hand rule until it discovers a node that is closer to the destination than the node that initiated the right-hand rule mode. As we observed in Section 8.5, our implementation achieves on average 5% routing improvement and significant stability in the

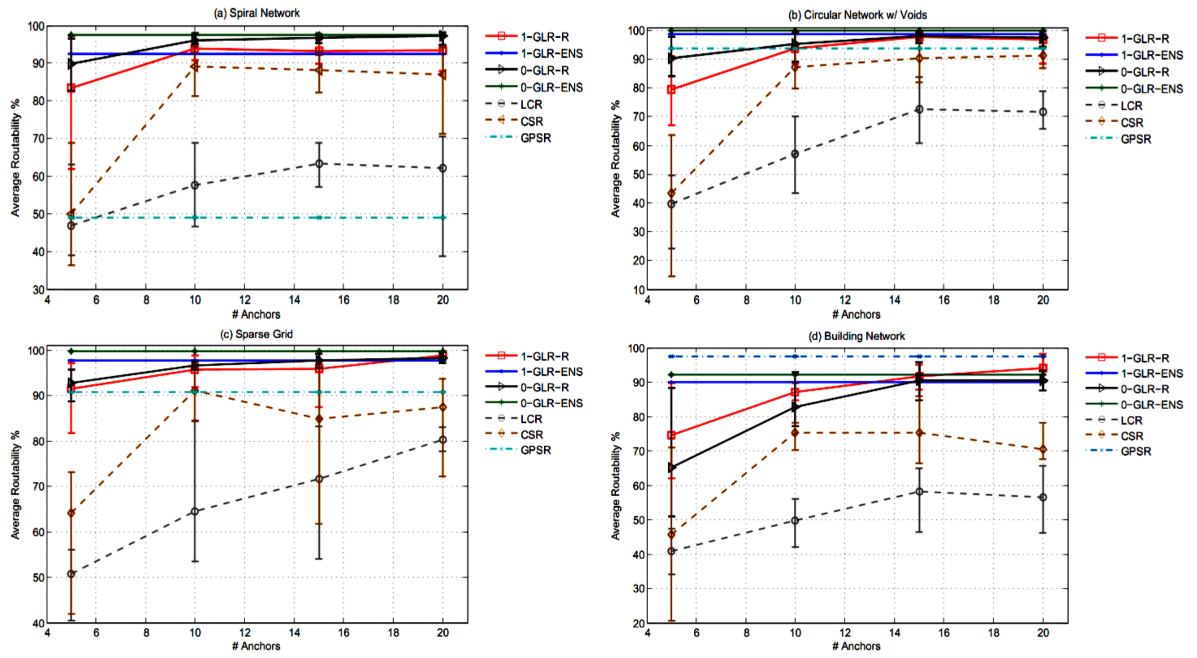


Figure 8.5. Average Routability of GLR with random and ENS anchor placement (GLR-R,GLR-ENS respectively), LCR, CSR and GPSR in (a) Spiral network, (b) Circular network with 3 voids, (c) Sparse grid with 100 missing nodes, and (d) building network. GLR-ENS and GPSR are independent from the random anchor placement.

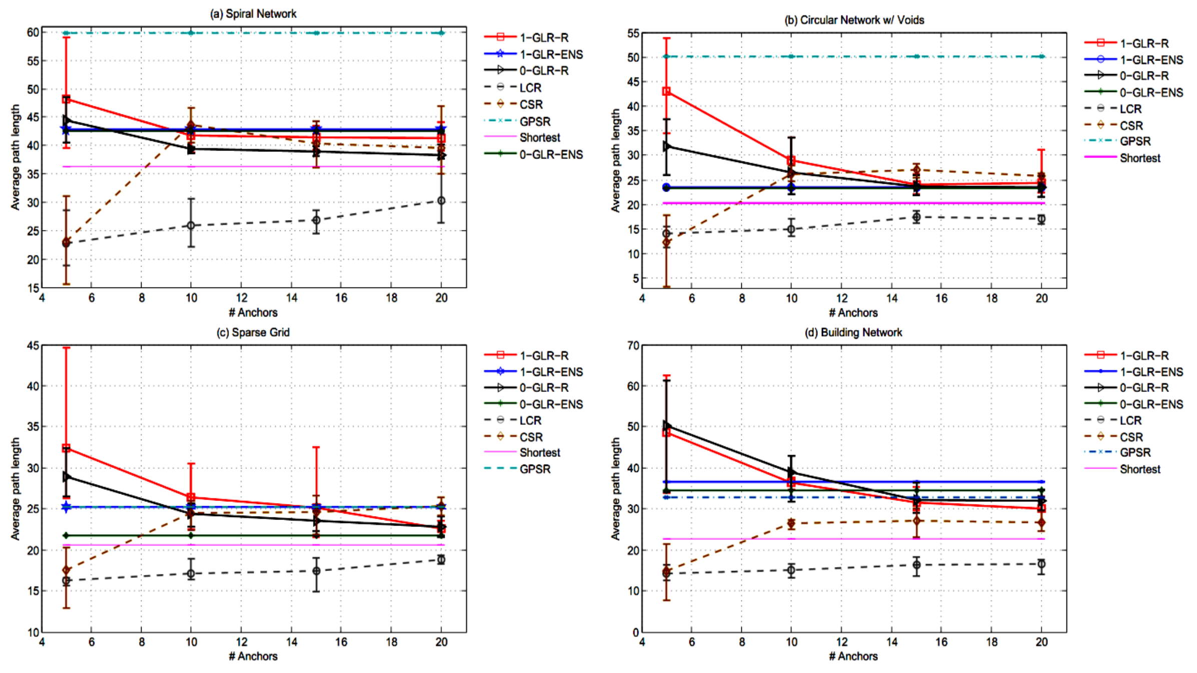


Figure 8.6. Average Path length of GLR with random and ENS anchor placement (GLR-R,GLR-ENS respectively), LCR, CSR and GPSR in (a) Spiral network, (b) Circular network with 3 voids, (c) Sparse grid with 100 missing nodes and (d) building network. GLR-ENS and GPSR are independent from the random anchor placement.

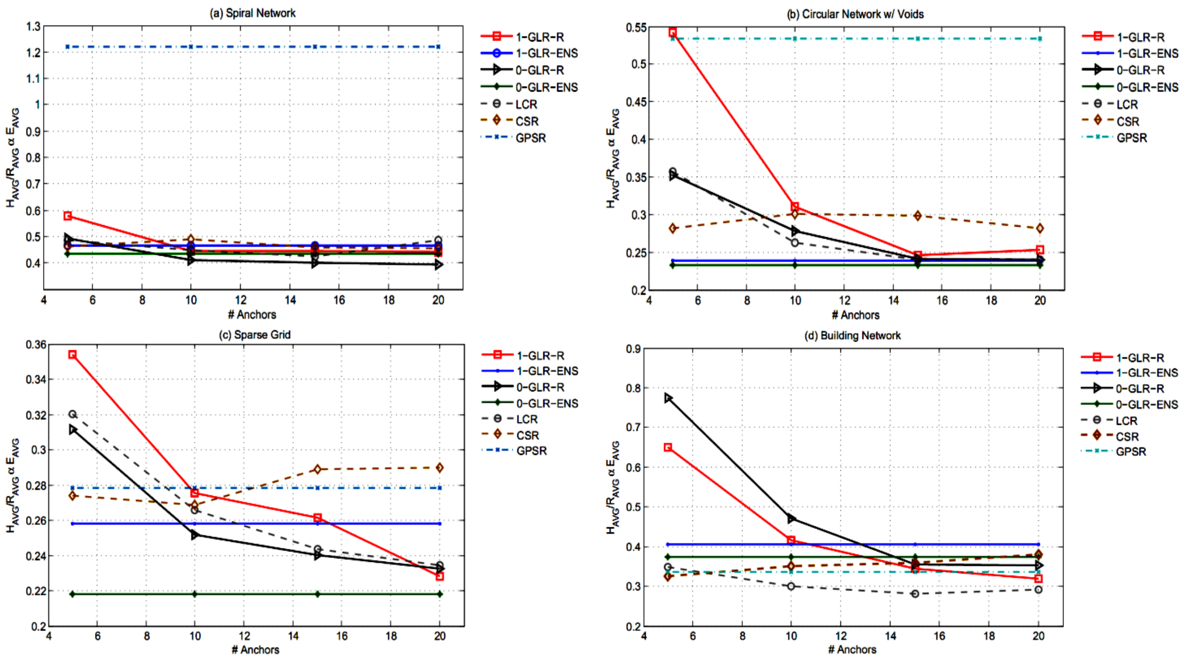


Figure 8.7. Average Energy consumption of GLR with random and ENS anchor placement (GLR-R,GLR-ENS respectively), LCR, CSR and GPSR in (a) Spiral network, (b) Circular network with 3 voids, (c) Sparse grid with 100 missing nodes and (d) building network. GLR-ENS and GPSR are independent from the random anchor placement.

path length as the TTL increases; thus, we have used our implementation of GPSR for comparison. Three options with differing complexity and accuracy are available for input dataset P for computing P_{SVD} in (2) [37]: a) the entire set of $N{\times}M$ VCS, b) the $M \times M$ matrix based on anchor's coordinates only, and c) $R \times M$ matrix based on a randomly selected coordinate set of R nodes, as briefly discussed in Section 8.3. The first is the most complex approach in terms of the computation and communication cost. When the number of random nodes selected is fewer than the number of anchors, c) is the most efficient. We use this simplest and most computationally and communication wise efficient option with only the coordinates of ten (R = 10) randomly selected nodes.

We use four CSU benchmark networks [21] that are representative of a variety of networks (see Figure 8.2). The number of nodes range from 400 to 800. A MATLAB® 2010b based simulator was developed for the computations. In selecting options for the different protocols, we have favored the competitor schemes so that we can demonstrate the effectiveness of GLR even under such conditions. The physical topologies in Figure 8.2 have four or fewer neighbors in a grid-like placement, and the communication range of a node in all four networks is set to unity. Time-To-Live (TTL) is set to 100 hops in all the routing schemes, unless otherwise stated.

This placement highly favors the GPSR scheme since the grid-like placement reduces looping and supports the right-hand rule based local minima overcome strategy. For example, the circular network with convex-shaped voids (Figure 8.2 (b)) can be significantly warped if more random positions are allowed within the communication range, and many such cases will introduce other concave physical voids that need to be overcome. However, actual positions are altered, preserving the original connectivity, resultant VCS will remain the same. Thus, actual distance between nodes, on the other hand, has no effect on the TPM as long as the corresponding nodes are within the communication range. Therefore, all such implementations correspond to the same VCS, thus the same topology map. Moreover, the performance would remain unchanged in GLR.

Average routability, average path length, and average energy consumption are the performance evaluation metrics used. Average routability evaluation considers all source-destination pairs, i.e., each node generated a set of $(N-1)$ messages, with one message for each of the remaining node as the destination.

$$\text{Average routability } (\mathrm{R_{AVG}}) = \frac{\text{Total \# of packets that reached the destination}}{\text{Total number of packets generated}} \qquad (8)$$

$$\text{Average path length } (H_{AVG}) = \frac{\text{Cumilative number of hops that all packets traversed}}{\text{Total number of packets generated}} \quad (9)$$

Note that the H_{AVG} calculation includes the path lengths for unrouted messages as well.

$$\text{Average energy consumption per successful packet delivery} = E_{AVG} = \frac{E_{\propto} \times H_{AVG} \times \text{Packet size}}{R_{AVG}\%} \quad (10)$$

$E_{\propto}$is the average energy consumption per bit transmission. Moreover, $R_{AVG}\%$ and H_{AVG} are estimated as in (8) and (9). Since $E_{\propto}$ and packet size are assumed to be fixed,

$$E_{AVG} \propto \frac{H_{AVG}}{R_{AVG}\%}$$

Compression of GLR with LCR, CSR, and GPSR contains below sectors:

1) Performance variation based on the number of anchors and the anchor placement scheme: Anchor placement is a crucial factor when deciding the performance of VCS thus quality of TPM. Hence, investigating the effect of the number of anchors and their placement in GLR is crucial. Two different anchor placement schemes are considered: random placement (Subsection A) and Extreme Node Search (ENS) based placement (Subsection B). $R_{AVG}\%$ as in (8), H_{AVG} as in (9) and E_{AVG} as in (10) are the parameters used for comparison.
2) Performance as the sparsity/node density of the network varies (Subsection C): Estimating the performance in dense as well as sparse networks is a good performance estimation to observe the uniformity in routing performance under different network properties. Moreover, TTL may play an important role in performance of routability in sparse networks.

3) Performance under node failures (Subsection D): Capturing the behavior in real WSNs, we evaluate the performance of GLR as the network nodes fail as an indicator of fault tolerance of GLR.

A. *Performance of 0-GLR and 1-GLR random anchor placement*

This section focuses only on the performance comparison based on random anchor placement. GPSR performance is independent of the number of anchors, as it uses physical coordinates. Figures 8.5 and 6 show the routability and the path lengths, respectively, averaged over ten random anchor placement configurations. Corresponding energy consumption is shown in Figure 8.7. Error bars indicate the maximum and minimum values in 10 simulation runs.

Performance of 0-GLR is about 2%-5% higher than that of 1-GLR in terms of average routability, path length, and energy consumption under random anchor placement, as shown in Figure 8.5-7 (a-c). However, in the building network (see Figure 8.5 (d)), 1-GLR outperform 0-GLR in average of 5% in routability. As the number of anchors is increased, the routability of GLR schemes increase on average by ~10% in sparse grid with 100 missing nodes, spiral, and network in the building, while that in circular network with voids is 23%. Both 0-GLR and 1-GLR outperforms existing logical coordinate-based routing schemes, LCR and CSR, with just five randomly selected anchors in all the four networks. For instance, in the building network, both the GLR schemes outperform LCR and CSR by 40% in each with five anchors; 40% and 15% with 10 anchors; 30% and 15% with 15 and 20 anchors. The average path length estimation includes the path lengths traversed even for packets are not routed correctly. Implemented LCR with complete path history, for example, discards the packets when it cannot route the packet further without looping, resulting in a smaller contribution toward path length, even though the actual shortest path length is much higher. The ‘shortest path’ corresponds to an ideal case path length. Higher routing percentages can be achieved only by routing such difficult packets, resulting in longer average path lengths. The performance for the case when there are exactly 10 random anchors is summarized in Table 8.3.

Since GLR schemes are clearly better than that of LCR and CSR, next we compare the performance of GPSR and GLR schemes. 0-GLR and 1-GLR outperform GPSR in circular network with three voids, spiral network, and in the sparse grid with missing nodes in terms of average routability and path length when there are more than 10 randomly placed anchors in the network as in Figures 8.5-6 (a-c) and Table 8.3. Backtracking scheme is used in GPSR, i.e., the right-hand rule is ideal for the structure in building the network, thus the performance of GSPR is superior in building the network (see Figure 8.5 (d)). One reason for GPSR to fail in building the network in Figure 8.5 (d) is that when it routes along the perimeter, it may take longer paths, resulting in expiration of TTL. In building the network, GPSR achieves 97.3% routability vs. 89.3% for GLR schemes on average. Moreover, the average shortest path length ratio of GPSR is 1.4, where that of GLR schemes is 1.5.

Energy consumption per successful routability is shown in Figure 8.7. Energy consumption of both the GLR schemes is comparatively low when the number of anchors is above 10. The energy consumption in spiral and circular networks is higher in GPSR than that in the rest of the schemes as shown in Figures 8.7 (a-b).

B. Performance of 0-GLR and 1-GLR with ENS anchor placement

With ENS-based anchors, more accurate topology maps are achievable with a significantly low number of anchors [40]. For the example networks considered, ENS-based anchors used for both 1 and 0 GLR-ENS rely only on two anchors for Figure 8.2 (a), four for 3 (c) and 3 (d), and 5 for Figure 8.2 (b). Note that the X-axis is not applicable for GLR-ENS schemes. As shown in [40], Greedy Routing performance in VC domain is improved due to the ENS-based anchors, which indicates that ENS provides a 'good' set of anchors that enhance performance in both TPM and VCS domains. Thus, GLR schemes under ENS anchors achieve higher routability with lower path length as shown in Figures 8.5 and 5. 0-GLR-ENS and 1-GLR-ENS achieve more or less the same routing performance in all four networks, significantly outperforming LCR, CSR, and GPSR. As summarized in Table 8.4, 0-GRL-ENS and 1-GLR-ENS accomplish routability over 98% for the circular network with three voids and sparse grid with missing nodes, with path length to shortest path length ratio of 1.4 and 1.2 respectively. Achieving routability of 92.4% and 93.4% for spiral network and network in the building correspondingly with only two and four anchors is remarkable. As shown in [40], routability of LCR and CSR under ENS anchors is 61.7% and 67.3% in the spiral network, 87.5% and 52.9% in the grid with 100 missing node, 84% and 75.5% in the circular network with voids, and 80.1% and 59.5% in the network in building. Since CSR requires at least three anchors, for the spiral network where there are just two anchors selected by ENS, two additional random anchors were used for CSR and LCR performances. Thus, proposed GLR schemes achieve significantly higher routabilities in the same networks under ENS anchors. Moreover, energy consumption comparison as in Figure 8.7, GLR-ENS schemes show significantly efficient energy performance than that of GPSR in all the four networks.

Next, we estimate the fraction of usage of TPM information in routing. Since every node starts routing in TC mode, the usage of TPM information in 0-GLR is 100%. Table 8.5 indicates that packets are forwarded by 1-GLR in the TC mode most of the time (72-86%), in VC mode 7-18% of the time, and in AM less than 10% of the time. This is a good indicator to illustrate that the higher routing performance achieved by GLR is due to the fact that it uses TPM information most of the time for the decision-making, and GLR combines the TPM and VCS information, overcoming the disadvantages in each domain.

Table 8.3
Performance Comparison between GLR, LCR, CSR and GPSR with 10 Random anchors

	Topology – Figure 8.2			
Performance Parameter	**Spiral**	**Circle with voids**	**Sparse Grid**	**Building Network**
% of nodes as anchors	2.4	2.0	1.25	2.9
Avg. Routability				
1-GLR	93.9	94.6	93.4	87.1
0-GLR	96.01	95.25	96.74	82.7
LCR	57.5	56.5	60	49.7
CSR	89.2	87.3	84.3	75.4
GPSR	49.1	93.8	89.6	97.4
Avg. Path Length				
Shortest path	36.1	20.3	20.7	22.8
1-GLR	41.8	28.3	28.1	36.3
0-GLR	39.3	26.5	24.3	38.9
LCR	25.9	15	16.3	15
CSR	43.7	26	26	26.4
GPSR	59.8	50.1	21.8	32.7

Table 8.4
Performance of GLR When Anchors are Strategically Placed using ENS

Topology Figure 8.2	**Avg. Routability %**		**Avg. Path Length**		$\frac{\textbf{Path length}}{\textbf{Shortest Path Length}}$	
	1-GLR	**0-GLR**	**1-GLR**	**0-GLR**	**1-GLR**	**0-GLR**
Spiral	92.4	97.5	42.9	42.6	1.2	1.2
Circle with voids	98.6	99.8	23.5	23.2	1.2	1.1
Sparse Grid	99.8	99.9	22.5	21.8	1.1	1.1
Building Network	90.0	92.3	36.5	34.5	1.6	1.5

Table 8.5
Active Percentile of each mode in Routing

Topology Figure 8.2	**Topological Coordinate mode**	**Virtual Coordinate mode**	**Anchor Mode**
Spiral	79.4%	12.7%	7.9%
Circle with voids	86.0%	7.0%	7.0%
Sparse Grid	75.0%	17.9%	7.1%
Building network	72.6%	17.7%	9.7%

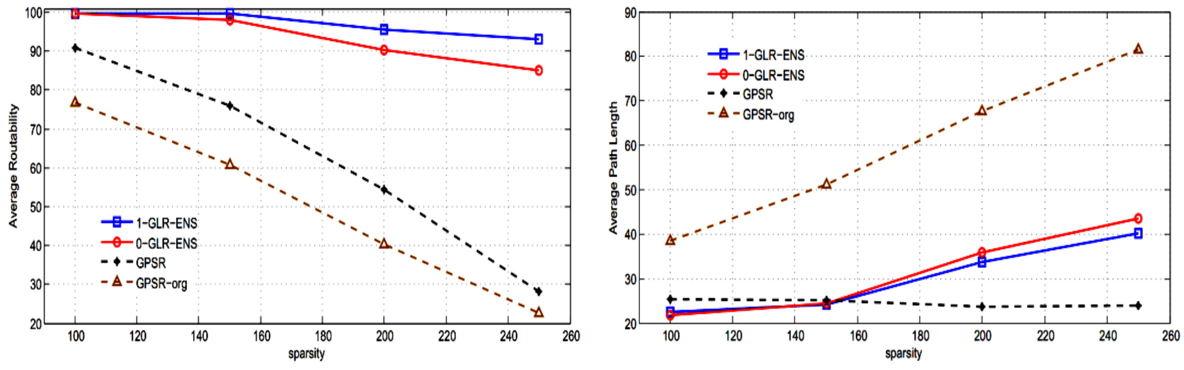


Figure 8.8. Performance of 0-GLR-ENS, 1-GLR-ENS and GPSR original (GPSR-org and GPSR our implementation (GPSR) as the network sparsity vary in a sparse grid. TTL is 100.

C. Performance of 0-GLR and 1-GLR with network sparsity and TTL

It is clear from the performance evaluation discussed in Subsection A and B that GLR-ENS schemes outperform both VCS-based schemes, LCR and CSR. Though GLR performance using over 15 random anchors achieves more or less the same performance as GLR schemes with ENS anchors, the latter is significantly energy efficient especially in VCS generation. Thus this point onwards, our focus is on the performance comparison between physical information-based routing and GLR-ENS. Next, the sustainability of performance of GLR-ENS and GPSR as the network sparsity varies under different TTLs is evaluated in the sparse grid (Figure 8.2 (c)).

In Figure 8.8, GPSR original scheme [69] and our modified implementation of GPSR (see Section 8.5(c)) are compared as the sparsity increases. Unlike in the original GPSR, in our implementation, the packet is routed based on GF until a physical void is met. After that, the node follows the right-hand rule to overcome the void. Then, the next node starts routing based on GF mode. Modified implementation achieves in average 10% routability improvement while achieving significant stability in the path length used compared to those of original GPSR. Thus, comparison of GLR is done against the modified implementation of GPSR.

For each sparsity configuration, TPM was re-evaluated. As it can be seen in Figure 8.8, both GLR-ENS schemes achieve superior performance over GPSR as the sparsity increases. In addition, 0-GLR-ENS and 1-GLR-ENS routability performance degrade by 0.1% per each missing node. After that, in Figure 8.9, performance of GPSR and GLR-ENS are evaluated as the TTL increases, keeping the sparsity of the grid fixed at 100 missing nodes. As shown in

Figure 8.9, GPSR routability does not improve, even though TTL is increased. Both GLR-ENS attain over 98% routability with just 100 hops TTL, hence performances of GLR-ENS were not re-evaluated for different TTLs. Figure 8.9 indicates the levels of routability of 0 and 1 GLR-ENS under TTL 100.

D. Performance of GLR under node failures

In this section, we are evaluating the performance of 1-GLR-ENS, 0-GLR-ENS, and GPSR under node failure environment in WSN. For GLR algorithms, initially VCS is generated for a sparse grid with 100 missing nodes. Then nodes generate their TC based on this VCS. After that, 30 nodes have been removed uniformly at random as failure nodes. VCS is not re-evaluated. Then average routability (as in (8)) and path length (as in (9)) are estimated and tabulated as in shaded row shown in Table 8.6. Next, another 30 nodes were removed uniformly at random at a time until total failed/removed nodes are 150. Performance, after removing 30 nodes at each turn, is tabulated in Table 8.6. Note that, routability as well as path length performance is evaluated, keeping the initially generated VCS and TCs unaltered after reach node removal, capturing the effect of node failure scenario. Similarly, in GPSR scheme, nodes assumed to be having exact physical positions initially. Removal of nodes does not affect the position information of the remaining nodes. Performance of GPSR is also evaluated after removing 30 nodes at a time. All the performances tabulated in Table 8.6 are averaged over five network configurations. On average, both GLR-ENS performances are higher than that of GPSR: up to 16% of the network nodes fail. Nevertheless, as we observed, GLR-ENS path length drastically increases as compared to the GPSR under node failures.

Topological coordinates are a cost-effective and good representation of the physical coordinates of the nodes. Thus, as the nodes fail in the network, it does not affect the topological positions of the available nodes. In contrast, connectivity-based VCSs will be significantly affected. Hence, updating VCS is necessary under network dynamics, which is under investigation.

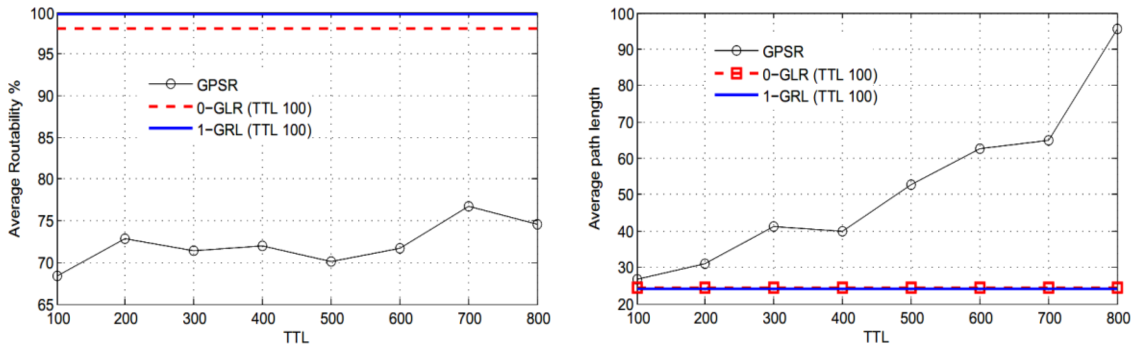


Figure 8.9. Performance of 0-GLR-ENS, 1-GLR-ENS, and GPSR when the TTL vary. Sparsity of the grid is 16.7%.

Intensive analysis of performance in terms of routability, path length, and energy consumption as the number of anchors, TTL and sparsity vary, indicating that combining VCS and topological coordinates leverages the advantages of both the domains while mitigating the drawbacks of the same by outperforming VCS-based as well as physical information-based routing schemes. This also proves that TPM is a good substitute for position information in routing.

8.6 Conclusion

Geo-Logical Routing is the first scheme to combine the advantages of logical and geographic information in routing. It uses topological coordinates (TCs) obtained from Virtual Coordinates (VCs) and VC-based routing to overcome the deficiencies associated with local minima problem in physical and logical domains. However, the overhead and uncertainty associated with node localization is avoided since the proposed scheme is independent from physical information. Cartesian coordinate estimation is using a SVD-based algorithm that uses VCs to produce a topology map. Topology-based Cartesian coordinates derived are more effective for geographic routing than the physical geographical coordinates, because topological maps preserve neighborhood information as well as connectivity information.

Table 8.6
Performance Comparission between 1-GLR-ENS, 0-GLR-ENS, and GPSR under node failures

	Initial # nodes 800			Initial # nodes 750			Initial # nodes 700		
# Failed nodes	1-GLR-ENS	0-GLR-ENS	GPSR	1-GLR-ENS	0-GLR-ENS	GPSR	1-GLR-ENS	0-GLR-ENS	GPSR
Average Routability %									
0	99.9	99.8	90.8	98.1	99.8	78.2	90.4	95.5	53.2
30	91.6	91.8	79.8	87.0	88.6	67.5	50.8	52.2	37.4
60	74.6	73.0	68.2	61.6	62.3	48.5	35.2	35.1	32.56
90	55.0	52.1	51.7	40.2	37.8	33.7	25.1	24.1	27.2
120	36.3	33.1	41.2	26.0	23.7	23.2	18.3	19.1	19.4
150	25.0	22.2	30.5	18.5	17.5	20.9	10.0	9.3	4.9
Average Path Length									
0	21.8	22.5	21.8	24.5	24.1	25.2	35.8	33.7	23.6
30	29.4	29.7	24.3	37.6	37.4	25.5	67.9	68.1	25.8
60	44.6	46.0	23.6	58.0	58.7	26.3	79.6	82.1	27.8
90	60.8	63.5	23.9	73.7	76.4	27.2	86.8	89.0	24.2
120	77.3	79.8	29.0	85.2	87.6	25.3	96.7	97.0	27.1
150	85.1	88.0	30.4	91.5	92.5	26.5	97.8	98.8	13.0

This research proposes a family of routing schemes called α-GLR, where α is the probability of selecting TC or VC for routing. We realized two instances, 1-GLR and 0-GLR, which combines topological and virtual information in two different ways. Proposed GLR schemes significantly outperform the existing VC-based routing schemes, Convex Subspace Routing and Logical Coordinate Routing. As illustrated, the strategic anchor placement scheme, Extreme Node Search (ENS), further improves the performance of GLR with about five anchors on average. Even under conditions favorable to physical coordinate-based routing, i.e., no localization errors and grid-like node placement, GLR outperforms geographic scheme GPSR in 3 out of 4 CSU Sensor-Net benchmarks. A building network where the GPSR performance seems to be better than GRL is one very favorable for geometric routing; still, GRL routability without costly, error-prone localization and with only four anchors there is noteworthy.

The novel concept of combining topological domain routing and virtual domain routing opens the path for designing novel adaptive routing protocols that operate in multiple coordinate domains. It is important to note that many variations of GLR algorithm can be developed, such as (1) moving only a certain distance toward the closest anchor in AM mode before switching and (2) tuning the parameter α to switch between TC and VC modes. Evaluation of such generalized GLR strategies is part of the ongoing work.

CHAPTER 09

ON TOPOLOGY MAPPING AND BOUNDARY DETECTION OF SENSOR AND NANONETWORKS DEPLOYED ON 2D AND 3D SURFACES

9.1 Introduction

The cost and size trends of nodes for networks of wireless devices, such as active and passive RFIDs, sensor nodes, and nano devices point to future networks where nodes in massive quantities are deployed over different structures and areas for monitoring infrastructure and environment. Consider for example, an oil pipeline, a boiler, or a bridge that needs to be monitored for corrosion, temperature distribution, or structural integrity. Tiny sensors capable of wireless communication and minimal computation capability can be deployed in massive quantities on their surfaces. As another possible application, consider a suit for an astronaut or a firefighter made with a smart fabric containing an irregular mesh of (nano) sensors. Obtaining the surface maps of such network deployments would have many potential applications. In fact, if such topology maps could be obtained quickly, it may even be possible to follow the motions made by the person wearing the suit of smart fabric. A second important problem related to such systems is the boundary identification. The boundary may correspond to the physical boundary of the network, or even an event boundary such as the one marking the boundary of a corroding patch of a pipeline.

This research presents, for the first time, a technique for obtaining topology maps of a network of nodes deployed over a 3D surface. The technique does not involve measuring signal strengths or time delays, which are costly and often impractical to implement in large-scale applications. Furthermore, they are susceptible to noise, fading, non-uniform attenuations and reflections (due to different types of materials and surfaces involved, lack of line of sight communication, etc.), factors that are difficult to overcome even in 2D wireless sensor networks. The proposed topology map generation involves only counting the number of hops a message has to pass through in getting from a set of randomly selected or strategically placed nodes (called anchors) to each of the nodes. As such, it is not affected by many of the obstacles related to techniques that rely on analog measurements. We then use these topology maps for the detection of boundaries, such as internal and external boundaries of a network, or even boundaries of events detected by multiple network nodes.

In topological domain, a network can have more than one valid embedding [90] in contrast to that in the physical domain. The actual embedding is one out of many, but identifying the correct embedding solely based on the connectivity information is challenging. Hence topological information-based boundary identification captures a union of boundary nodes in every embedding. Thus, the actual set of boundary nodes is a subset of it. Our topology preserving map based approach successfully identifies the correct network embedding without the use of physical information. Hence this approach is more pragmatic in WSN and similar applications. 2D topology preserving map generation is discussed in [37]. This research discusses the generation of topology preserving maps of 3D surface network for the first time and also develops a boundary detection strategy for 2D and 3D surface networks based on the topology maps.

A network has a specific embedding and can have three different types of boundaries which the scheme presented in this research aims at detecting. First is network's outer boundary, which consists of a unique subset of nodes. Second is an inner boundary, which depends on the network's node density as well as nodes' communication range. For example, consider a network with a lake in the middle. If the communication range is large enough to communicate with the nodes at the other end of the lake, there is no physical void in the network. Last type of boundary is an event boundary. For example, events such as target tracking or forest fires have dynamic event boundaries, while an underground chemical plume may have relatively static boundaries that change very slowly over time [64].

In Section 9.2, the related work is discussed. Next in Section 9.3, methodology of 3D topology preserving surfaces is proposed. Section 9.4 shows the results from 3D topology preserving surfaces. Then, inner, outer, and event boundary recognition of 2D and 3D networks are discussed in Section 9.5. Section 9.6 evaluates the performance of the proposed boundary detection algorithm, while Section 9.7 concludes our work.

9.2 Related Work

A. Topology Preserving Maps

Topology preserving map (TPM), in the present context, refers to a map of a network that preserves the connectivity topology of a network while capturing the physical properties of the network, such as its shape, internal and external boundaries, and their relationships. While true 3D spatial models or 2D geographical maps with exact coordinates of nodes can be used to capture this information accurately, generation of such models and maps is very expensive. Alternatives for generating the geographical or physical maps include use of GPS, which is not practical under many circumstances, e.g., when nodes are tightly packed, nodes have to be inexpensive, or when deployed on complex structures where GPS reception is not available. Another option is to use a localization algorithm [14], which requires measurement of parameters such as signal strength (RSSI), which is error-prone, difficult, and may even be impossible to use in certain structures containing many metal surfaces, such as refineries. Another method is to use the time delay to estimate distances. This requires synchronization of clocks, something that may not be feasible in harsh environments with multipath, or at inexpensive nodes subject to conditions such as temperature changes. This leaves very few practical alternatives.

Recent research [37] provides an intriguing approach for generating topology maps for 2D networks by generating a virtual coordinate set based on hop distances. In a Virtual Coordinate System (VCS), a node is identified by a vector containing its distances, in hops, to a set of nodes called anchors [36]. A VCS maps the physical network into a logical coordinate system with the dimension of that logical space equal to the number of anchors. Since this mapping is based on geodesic distances, VCS have connectivity information embedded in the coordinates while it loses the directionality information, i.e., north or south (cardinal directions), left or right (egocentric directions) [34]. As each ordinate propagates as concentric circles centered at the corresponding anchor, the angular information is completely lost. In [37], a singular value decomposition based transformation is provided that generates a set of 2D Cartesian coordinates for node positions. The topology preserving map generated using these coordinates in fact can capture the geographic features and is even better than an actual physical map with exact (x,y) coordinates for the purpose of routing and self-organization of a network [132]. This current research presents, for the first time, a technique for generating such topology preserving maps of networks deployed on 3D surfaces.

B. Network boundary detection

The ability to self-organize and route messages among sensor nodes is key to the deployment of future large-scale Wireless Sensor Networks (WSNs). Boundary detection plays a crucial role in information fusion and dissemination in 2D and 3D WSN applications such as target tracking, plume tracking [64], forest fires, animal migration, underwater WSNs, and surveillance applications [51]. Currently available schemes can be broadly categorized as physical information-based and topological/connectivity information-based boundary detection in 2D networks [24][30][31]. The former uses physical position of nodes to identify the boundary while the latter uses topological/connectivity information of the network for the detection. Physical domain schemes rely on node location or physical position information [14] obtained using localization algorithms or GPS. Localization based on parameters such as RSSI is error-prone and difficult for a network of thousands or even millions of sensors. Equipping nodes with GPS is costly and infeasible for many applications. A decentralized localized algorithm used to identify the perimeter nodes using barycentric technique on neighborhood information [54]. In [32], sensor nodes remotely collect data about various points on the boundary and estimate the boundary along with the confidence intervals using a regression relationship among sensor locations and the distances to the boundary. Voronoi and neighbor-embracing polygon-based localized boundary detection approach is discussed in [129], while [89] proposes a localized perimeter detection algorithm which incorporates the angle between nodes for dense networks. A localized algorithm for 3D boundary detection based on unit ball fitting followed by a reinforcement algorithm named isolated fragment filtering is presented in [132] for a network with known node locations.

The main challenge of boundary identification in the topological domain, which is based on connectivity information, is discovering the correct valid mapping of the physical map to the connectivity-based map. Each valid embedding has a boundary. Boundary detection proposed in [51] uses the shortest path to identify a course and those nodes connect themselves through local flooding, hence the algorithm suffers from high communication cost. In [55], nodes identify patterns called flowers. If a flower exists, the node is an internal node. In an extension of this work in [90], nodes try to identify a family of patterns by defining a set of rules. If a node satisfies the defined set of rules, then it is an internal node. An isocontours-based hole boundary detection scheme is proposed in [52]. This algorithm requires significant computational power.

The 2D and 3D boundary detection scheme proposed in this research is also a topology information-based approach. Since we use topology preserving map of the network, the correct embedding of the network can be identified. Furthermore, virtual coordinates together with derived topology coordinates provide connectivity as well as location mappings. Thus, the results of the identification are impressive compared to the existing connectivity-based approaches.

C. Event boundary detection

The goal of event boundary detection is to detect the profile or the contour of a region or a surface over which the event has occurred. Examples of events include spread of a chemical plume [64], or contour of a segment of field which needs application of fertilizer. [30]

proposes a novel algorithm for detecting event boundaries based on a Gaussian mixture model. The main disadvantages are the uncertainty associated with the probabilistic prediction and the complexity. Three different schemes to identify the event boundaries are proposed in [24]. The three approaches are based on localized algorithms, namely statistical approach, classifier-based approach, and image-processing approach. A median-based localized approach is presented in [31] for faulty sensor identification and fault-tolerant event boundary detection. A noise-tolerant algorithm for event and event boundary detection based on moving averages to eliminate noise effects in evenly distributed localized WSNs is presented in [66]. In this research, we use the relationships from topology maps to do event boundary detection.

Table 9.1
Notations Used in Chapter 9

Notation	**Description**
N	Number of network nodes
$N_i \in N$	Node *i* (current node)
M	Number of anchors
$n(N_i, N_j)$	Minimum hop distance between node N_i and N_j
$C_v(N_i)$	node N_i as a vector of M virtual coordinates
$\{A_i \| A_i \in N\ \forall i = 1{:}M, M \ll N\}$	Anchor set
$P_{N\times M}$	Virtual Coordinate Set for Transformation Computation
$[P_{SVD}]_{N\times 2} = [X_T, Y_T]$, $[P_{SVD}]_{N\times 3} = [X_T, Y_T, Z_T]$	Topological Coordinate Set in 2D Topological Coordinate Set in 3D
$[X_T, Y_T]_{(i)} = [X_{T,i}, Y_{T,i}]$ $([X_T, Y_T, Z_T]_{(i)} = [X_{T,i}, Y_{T,i}, Z_{T,i}])$ $= [P_{SVD}]_{(i)}$	Node N_i*'s* 2D (3D) topological coordinates
$D_{N_iN_j}$	Distance between nodes N_i and N_j in 2D topological domain
${N_k}^{(i)}$	Neighbors set of node i
$\Delta N_x N_y N_z$	Area of the triangle $N_x N_y N_z$
S	Semi-perimeter of the triangle

9.3 Topology Preserving Maps of 3D Surfaces

Consider a network with N nodes. Denote the *i*th node by N_i $(1 \leq i \leq N)$. A VC system is used in which each node is characterized by a vector of virtual coordinates denoting the distances to each of a set of M anchors ($N >> M$). A_m $(1 \leq m \leq M)$ denotes the *m*th anchor. Note that an anchor is one of the N nodes, but we use A_m for clarity. Let $n(N_i, N_j)$ denote the hop distance between nodes N_i and N_j; $n(N_i, N_j)$ is zero when $N_i = N_j$, otherwise it is a positive integer. A node N_i is thus identified by the vector of M virtual coordinates given by

$$[C_v(N_i)] = [n(A_1, N_i), n(A_2, N_i), \dots\ n(A_M, N_i)] \tag{1}$$

where $n[A_m, N_i]$ is the number of hops from N_i to the *m*th anchor A_m. The notations used are summarized in Table 9.1.

A VCS maps the physical network into a logical coordinate system. The dimension of the logical space is equal to the number of anchors. Depending on the number of anchors and the anchor placement, it may be a many-to-one mapping. Thus, an adequate number of anchors are needed to prevent ambiguities. Since this mapping from connectivity to VCs is based on geodesic distances, VCs have connectivity information embedded in the coordinates. As the anchor coordinates propagate in concentric circles outwards from each anchor, the VCs lose explicit directionality information, i.e., cardinal directions (north vs. south, etc.) as well as egocentric directions (left vs. right, etc.). Formation of the topology maps requires the recovery of this low-dimensional information from the high-dimensional VCs.

Let P be the $N \times M$ matrix containing the virtual coordinate of the nodes of the network. Let the singular value decomposition of P be

$$P = U.S.V^T \tag{2}$$

where U, S and V are $N \times N$, $N\times M$, and $M \times M$ matrices respectively. U and V are unitary matrices, i.e., $U^T U = I_{N\times N}$ and $V^T V = I_{M\times M}$. Each node in P thus can be represented by its Principal Components (PC) vector, given byP_{SVD}, where

$$P_{SVD} = P.V \tag{3}$$

In order to obtain the topology preserving map, we use transformed values of VCs of the nodes in the network to this new set of axis, which corresponds to the principal components of P.

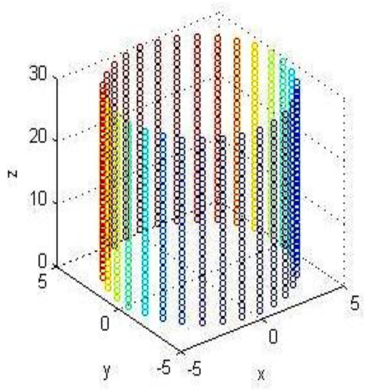


Figure 9.1: A network on a cylindrical surface (900 nodes)

Consider the uniform cylindrical surface shown in Figure 9.1 on which 900 nodes are deployed. Although the nodes are equally placed on a grid, our experiments show that the node distribution is not important as long as the entire surface is covered, and the network is not disconnected. Figures 9.2 (a)-(d) show the plots of the first four PCs for each node in the network. They are denoted by $P_{SVD}^{(1)}$, $P_{SVD}^{(2)}$, $P_{SVD}^{(3)}$, $P_{SVD}^{(4)}$ respectively. PC values of a data set is such that the first component has as high a variance as possible, i.e., it accounts for as much of the variability in the data as possible, and each succeeding component has the highest variance possible under the constraint that it be orthogonal to the preceding components.

The set of VCs has the connectivity information embedded in it, though it loses directional information. As each ordinate propagates as concentric circles centered at the corresponding anchor, the dominant component correspond to this radial propagation. Thus, the most significant ordinate based on SVD, i.e., first column of P_{SVD} shown in Figure 9.2 (a) contains radial information. One can observe that it exhibits minima on the band at the center of the cylinder, with the value increasing as the point of interest nears top or bottom of the cylinder. As the cylinder is a symmetric structure in X and Y directions, there is no variation of the value along a horizontal cross-section of the cylinder. This is consistent with the observations made for 2D networks in [37], where the first PC has a concave shape as it propagates outward radially, somewhat like an individual VC.

As SVD provides an orthonormal basis, 2nd, 3rd, and 4th PCs are orthogonal to 1st ordinate while being perpendicular to one another. Observe in Figure 9.2 (b) that the second PC varies along the height of the cylinder, thus it can be used to obtain the Z-coordinate for the topology map. More interesting are the distributions of the third and fourth components. Note that the lines joining minima and maxima of the third component on any horizontal plane are perpendicular to the corresponding line in the fourth component. Given the fact that the 2nd, 3rd, and 4th components are orthogonal to each other, and that the radial information related to VC propagation is contained in the 1st component which we ignore, these three PCs provide an orthogonal set of axis in the physical domain. Thus, the second, third, and fourth columns of P_{SVD}, $[X_T, Y_T, Z_T]$ provide a set of 3D Cartesian coordinates for node positions on a 3D topology preserving map, i.e.,

$$[X_T, Y_T, Z_T]_{(i)} = [{P_{SVD}}^{(2)}, {P_{SVD}}^{(3)}, {P_{SVD}}^{(4)}]_{(i)} = [P_{(i)}.V^{(2)},\ P_{(i)}.V^{(3)}, P_{(i)}.V^{(4)}] \qquad (4).$$

The topology map for this cylinder is given in Figure 9.3.

Thus the VCs of any node given by (1) can be transformed to the PC axis obtained in (2) using

$$[C_{SVD}(N_i)] = [C_v(N_i)].V \tag{5}$$

Let $C_{SVD}(N_i)^{(j)}$ denote the jth element of the vector $[C_{SVD}(N_i)]$. Then the 3D topology coordinates of N_i is given by

$$[X_T, Y_T, Z_T]_{(i)} = [\ C_{SVD}(N_i)^{(2)}, C_{SVD}(N_i)^{(3)}, C_{SVD}(N_i)^{(4)}] \tag{6}$$

Note that we need only the $M \times M$ unitary matrix V to convert the virtual coordinates of a point to it topological coordinates. The P matrix decomposed in (2) is a $N \times M$ matrix containing the virtual coordinates of all the nodes of the network. However, it is possible to compute an effective V with just a small number of VCs. Thus instead of using the $S \times M$ matrix, R corresponding to a subset of nodes is used. As noted in [37] for 2D networks, in the case of 3D surfaces also, the set of S nodes selected for SVD may consist of all the nodes $(S = N)$, only the anchor nodes $(S = M)$, or a selection of S $(S << N)$ nodes that are strategically placed or randomly selected. Different options provide different computation and communication overhead vs. accuracy tradeoffs. But nodes can perform the singular value decomposition of selected random set of nodes or set of anchors coordinates to acquire the matrix V_R

$$R = U_R.S_R.{V_R}^T \tag{7}$$

Thus the topology coordinates can be obtained from (5) using V_R instead of V. Also noteworthy is the fact that only the second, third, and fourth columns $\left[V^{(2)}, V^{(3)}, V^{(4)}\right]$ of matrix V are needed, and therefore the use of SVD computation is not necessary. Eigenvalue decomposition can be used to obtain vectors $\left[{V_R}^{(2)}, {V_R}^{(3)}, {V_R}^{(4)}\right]$ without calculating entire matrices V and U, reducing the computational complexity from $(4R^2M + 8NM^2 + 9M^3)$[58] to $(4M^2N + 8M^3)$[58] (upper bound). Former is the computational complexity of entire SVD decomposition while the latter is the complexity of complete eigenvalue decomposition.

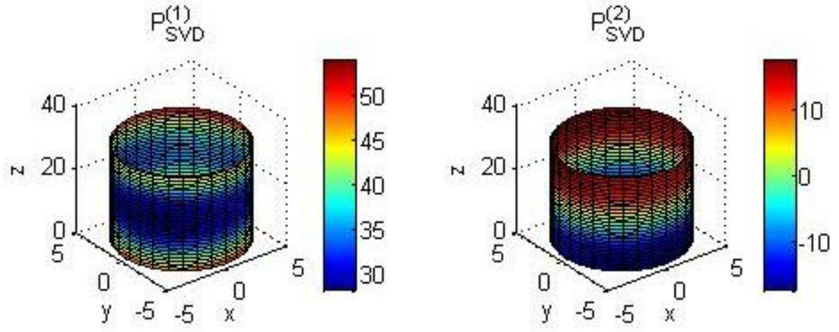


Figure 9.2: Different SVD components for cylinder: a) $P_{SVD}^{(1)}$, b) $P_{SVD}^{(2)}$, c) $P_{SVD}^{(3)}$ and d) $P_{SVD}^{(4)}$

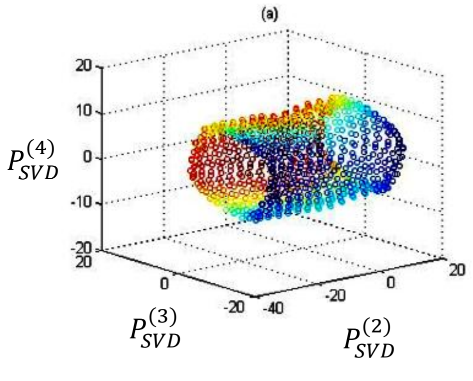


Figure 9.3: Topology Generation of a cylinder (900 nodes)

9.4 Results – 3D Topology Preserving Maps

In this section we evaluate the topology preserving maps generated using the proposed scheme. Two example networks deployed on 3D surfaces are considered as shown in Figure 9.4:

a. T-joint: A pipeline structure joining two perpendicular cylinders in a T-joint. Each cylinder has a radius 2.54 unit, height 16 units, and is covered with a grid of 512 nodes, each with a communication range of 1. 20 randomly placed anchors were used.

b. Cylinder with a hole: A cylinder of radius 2.54 and height 24 with a hole through it (two aligned voids on the surface on opposite sides) is covered with 490 nodes, each with a communication range of 0.5. 15 randomly placed anchors were used.

Topology preserving maps of the corresponding physical topologies are shown in Figures 9.5 and 6. The results clearly demonstrate the effectiveness of the topology preserving map generation for network deployed on 3D surfaces.

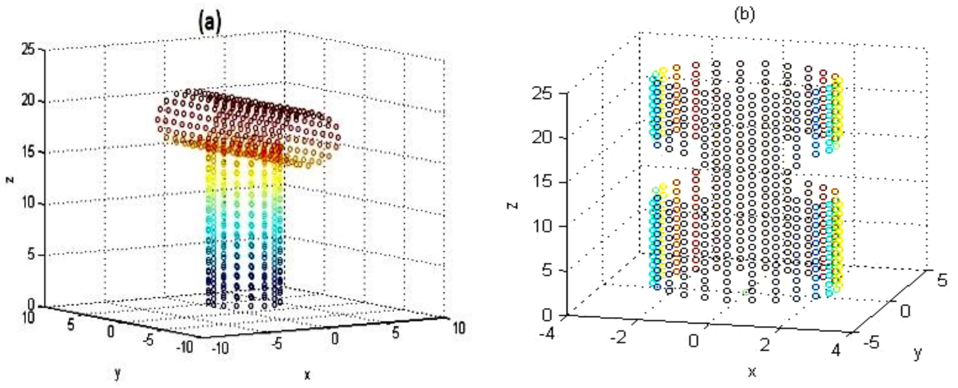


Figure 9.4. Examples 3D surfaces: a) Two perpendicular cylinders (T joint); (b) A cylinder with a hole (two voids on opposite ends).

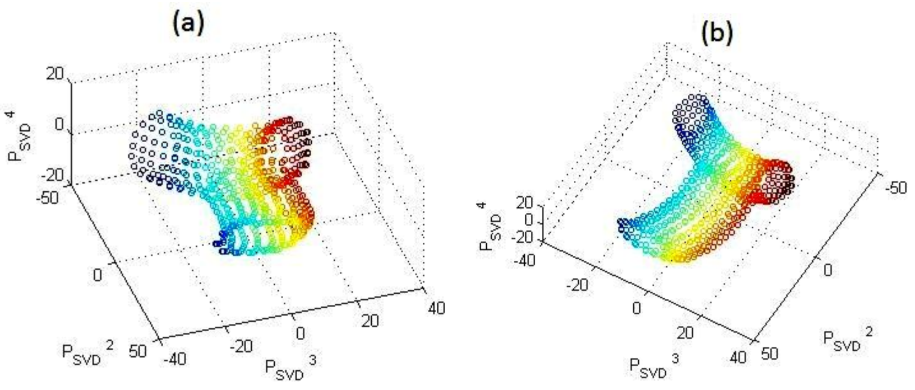


Figure 9. 5. Two views of the topology preserving maps for the network on the 3-D surface of T-joint structure.

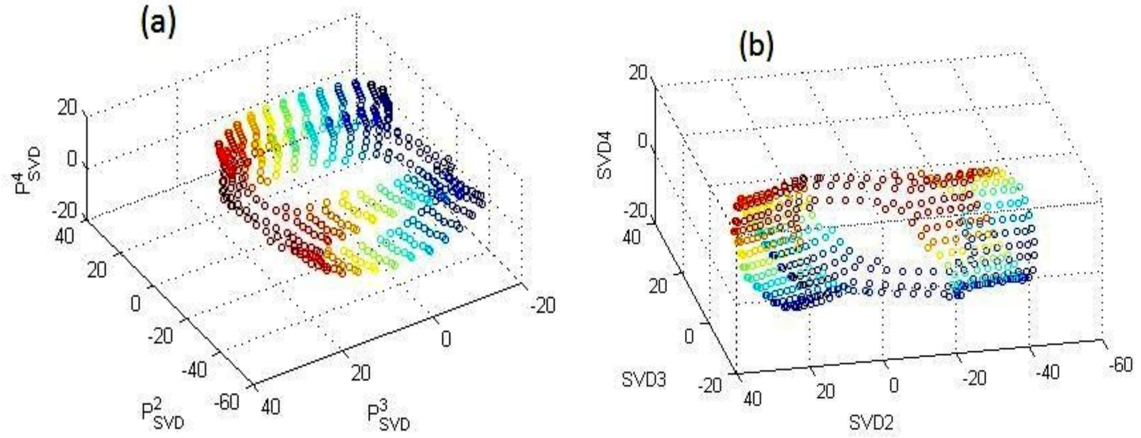


Figure 9.6: Two views of the topology preserving map of a network deployed on the cylinder with a hole.

9.5 Sensor Network Boundaries

Next we address the identification of the nodes forming the network boundary. Consider the example network in Figure 9.7 (a). Each node can communicate with its one-hop neighbors and up to four neighbors are possible (north, south, east, and west). Whether the node A belongs to the boundary or not is not clear-cut, and it depends on the way boundary nodes are defined. We define the boundary of a network as the set of nodes that has a contribution toward the outer bound of the network communication (can also be interpreted to mean sensing) coverage. Hence the identified boundary nodes need not to be connected in the communication topology. In this example, nodes B and C belong to the boundary while A does not. Consider the coverage by the entire network as in Figure 9.7 (b). If the outer physical loop of the coverage is considered, then the nodes that have contributed to this loop are boundary nodes. The outer boundary and the inner boundary of the network are as indicated in Figure 9.7 (c). Network boundary is a function of a nodes' communication range and node density of the network. Boundary nodes in the 3D case can be defined the same way.

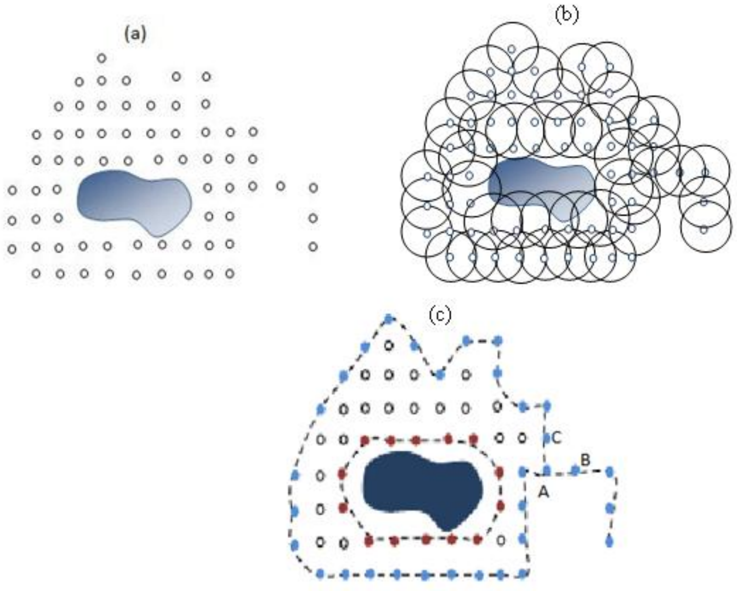


Figure 9. 7. Example network: (a) Sensor network deployed in an area with a lake, (b) Outer bound of the coverage of the network nodes, and (c) Defined network boundary

First, the detection of boundaries in 2D networks is considered, and then the algorithm is extended to 3D network surfaces.

A. 2D network boundary detection

Consider a 2D network with N nodes, with each node characterized by a vector of VCs, with distance to each of the M anchors ($N >> M$). Let P be the $N \times M$ matrix of the virtual coordinates and P_{SVD} be the topological coordinate set of the network. How the nodes can acquire its topological coordinates locally, in the case of 2D networks, is explained in [37].

In a connected network, any node has one or more neighbors. Let the node, which is to be tested for a boundary node, be N_i. Assume it has k neighbors denoted by $N_K{}^{(i)} = \{N_k\}$. If $k = 1$ or 2, N_i is a boundary node. When k is 3 or greater, an algorithm is required to check whether N_i is a boundary node or not.

Consider the case where N_i has three neighbors. Let the topological coordinates of N_i be $[x_{T,i}, y_{T,i}]$.

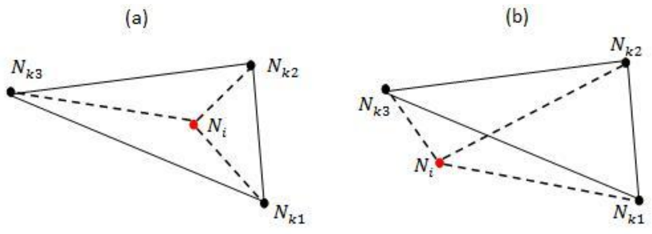


Figure 9. 8: Relationship between a) an internal node and its neighbors and b) a boundary node and its neighbors.

B. Network boundary detection algorithm

If $\mathrm{N_i}$ is an internal node, area of $\Delta \mathrm{N_{k1}N_{k2}N_{k3}}$ is the same as the total area enclosed by $\Delta \mathrm{N_{ki}N_{k2}N_{k3}}$, $\Delta \mathrm{N_{k1}N_{ki}N_{k3}}$, and $\Delta \mathrm{N_{k1}N_{k2}N_{ki}}$. If $\mathrm{N_i}$ is an external node, area of $\Delta \mathrm{N_{k1}N_{k2}N_{k3}}$ is lower than the total area enclosed by $\Delta \mathrm{N_{ki}N_{k2}N_{k3}}$, $\Delta \mathrm{N_{k1}N_{ki}N_{k3}}$, and $\Delta \mathrm{N_{k1}N_{k2}N_{ki}}$. Figure 9.8 illustrates this relationship.

This can be easily extended to the case with a higher number of neighbors by simply considering the corresponding polygon, or even three neighbors at a time. In the latter case, each triplet should identify it as a boundary node. Area of a polygon can be calculated in terms of triangles. In this research, we consider the relationship in terms of the triangle for in the boundary detection.

Triangular area calculation

Denote the area of any triangle created by node $\mathrm{N_{k1}}$, $\mathrm{N_{k2}}$ and $\mathrm{N_{k3}}$ be $\Delta N_{k1}N_{k2}N_{k3}$. Let the L^2 distances between $\mathrm{N_{k1}}$, $\mathrm{N_{k2}}$, and $\mathrm{N_{k3}}$ based on topological coordinates be $D_{N_{k1}N_{k2}}$, $D_{N_{k3}N_{k2}}$, and $D_{N_{k1}N_{k3}}$ respectively. Then

$$\Delta N_{k1}N_{k2}N_{k3} = \sqrt{S\left(S - D_{N_{k1}N_{k2}}\right)\left(S - D_{N_{k1}N_{k3}}\right)\left(S - D_{N_{k3}N_{k2}}\right)} \tag{8}$$

where S is the semi perimeter of the triangle

$$S = \frac{1}{2}\left(D_{N_{k1}N_{k2}} + D_{N_{k1}N_{k3}} + D_{N_{k3}N_{k2}}\right) \tag{9}$$

The L^2 distance between any two nodes N_i and N_j in the network based on their topological coordinates is

$$D_{N_iN_j} = \sqrt{\left(X_{T,i} - X_{T,j}\right)^2 + (Y_{T,i} - Y_{T,j})^2} \quad (10)$$

The boundary detection algorithm is summarized in Figure 9.9.

```
INPUT:VCS of neighbors
OUTPUT: $N_i$ is a boundary node or not
Algorithm for identifying whether N_i is a boundary node:
  If $|N_K{}^{(i)}| \leq 2$
      $N_i$ is a boundary node
Elseif $|N_K{}^{(i)}| > 2$
      For all possible neighbor triplets $N_{k1}, N_{k2}, N_{k3} \epsilon N_K{}^{(i)}$
            If $area\ of\ any\ of\ the\ three\ triangles\ \Delta N_{k1}N_{k2}N_{k3}$ OR $\Delta N_iN_{k2}N_{k3}$ OR $\Delta N_{k1}N_{k2}N_i$ OR $\Delta N_{k1}N_iN_{k3}$ == 0
                  $N_i$ is a boundary node candidate
            Elseif $\Delta N_{k1}N_{k2}N_{k3} > \Delta N_iN_{k2}N_{k3} + \Delta N_{k1}N_{k2}N_i + \Delta N_{k1}N_iN_{k3}$
                  $N_i$ is an internal node
                  Stop checking
            Elseif $\Delta N_{k1}N_{k2}N_{k3} == \Delta N_iN_{k2}N_{k3} + \Delta N_{k1}N_{k2}N_i + \Delta N_{k1}N_iN_{k3}$
                  If $\Delta N_iN_{k2}N_{k3}$ AND $\Delta N_{k1}N_{k2}N_i$ AND $\Delta N_{k1}N_iN_{k3}$ ~= 0

                        $N_i$ is an internal node
                        Stop checking
                  Elseif $\Delta N_iN_{k2}N_{k3}$ OR $\Delta N_{k1}N_{k2}N_i$ OR $\Delta N_{k1}N_iN_{k3}$ == 0
                        $N_i$ is a boundary node candidate
                  End
            Elseif $\Delta N_{k1}N_{k2}N_{k3} < \Delta N_iN_{k2}N_{k3} + \Delta N_{k1}N_{k2}N_i + \Delta N_{k1}N_iN_{k3}$
                        $N_i$ is a boundary node candidate
            End
      End
      If $N_i$ is a boundary node candidate for all triplets of neighbors in $N_K^{(i)}$ Then
       $N_i$ is a boundary node
      End
End
```

Figure 9. 9: Pseudocode of boundary detection algorithm.

C. *Boundary detection in 3D surfaces*

When we consider WSNs on 3D surfaces, the distance between two nodes may be the curvilinear distance between them and not the Line-of-Sight (LoS) distance. But our boundary detection algorithm operates in a one hop neighborhood, and hence the LoS distance is a reasonable approximation sufficient for decision making. L^2 is used to approximate for the curvilinear distance. However, all three coordinates are used for distance evaluation in 3D domain by simply incorporating L^2 based 3D distance calculation in area evaluation.

D. Event boundary detection

The previous section addressed the detection of the network boundaries. However, the algorithm can be extended easily for the detection of event boundaries, i.e., the boundary of set of nodes detecting a certain event. Such boundary detection is critical in many surveillance applications. In order to address this problem, the nodes will treat an event detecting neighbors as 'existing' and others as 'non-existing' neighbors. Then the same algorithm is applied to the 'existing' neighbors set.

E. Complexity

Dominant memory and computational complexities of the algorithm are related to the memory and CPU usage involved with topological coordinate generation. Once the topology coordinates are known, the computational complexity is $O(|N_K|^2)$ while the memory complexity is $O(|N_K|)$.

9.6 Simulation Results: Network and Event Boundary Detection

A. Network inner and outer boundary identification

The performance of proposed algorithm is evaluated next. We use the five example networks that are representative of a variety of networks. The number of nodes range from 300 to 800. The networks are (a) a spiral network, (b) a circular network with a C-shaped concave void, (c) a circular network with three voids, (d) a square network with an E-shaped void, and (e) an odd-shaped network. MATLAB® 2009b based simulator was used for the computations. Two performance-evaluating metrics that capture the accuracy of boundary identification and the error in boundary identification are evaluated:

$$\%\ \text{accuracy of boundary identification (A\%)} = \frac{\#\ \text{nodes that are correctly identified}}{\text{Total number of Boundary nodes}}\ \%$$

$$\%\ \text{Error in boundary identification(E\%)} = \frac{\#\ \text{nodes that incorrectly identified}}{\#\ \text{nodes that are correctly identified}}\ \%$$

In the simulation, a single hop neighborhood was considered. Results for boundary detection are shown in Figure 9.10. As can be clearly seen for networks in Figures 9.10 (a)-(e), it is zero value for E and 100% for A. This demonstrates the effectiveness of the proposed scheme and the effectiveness in using of the topology preserving maps in detecting the correct network boundary.

B. Event boundary detection

In this section, the effectiveness of the algorithm in event boundary detection is analyzed. We simulated the case where there is an event where the boundary changes with time, e.g., a forest fire or a chemical plume . Proposed algorithm identifies and tracks the event boundary

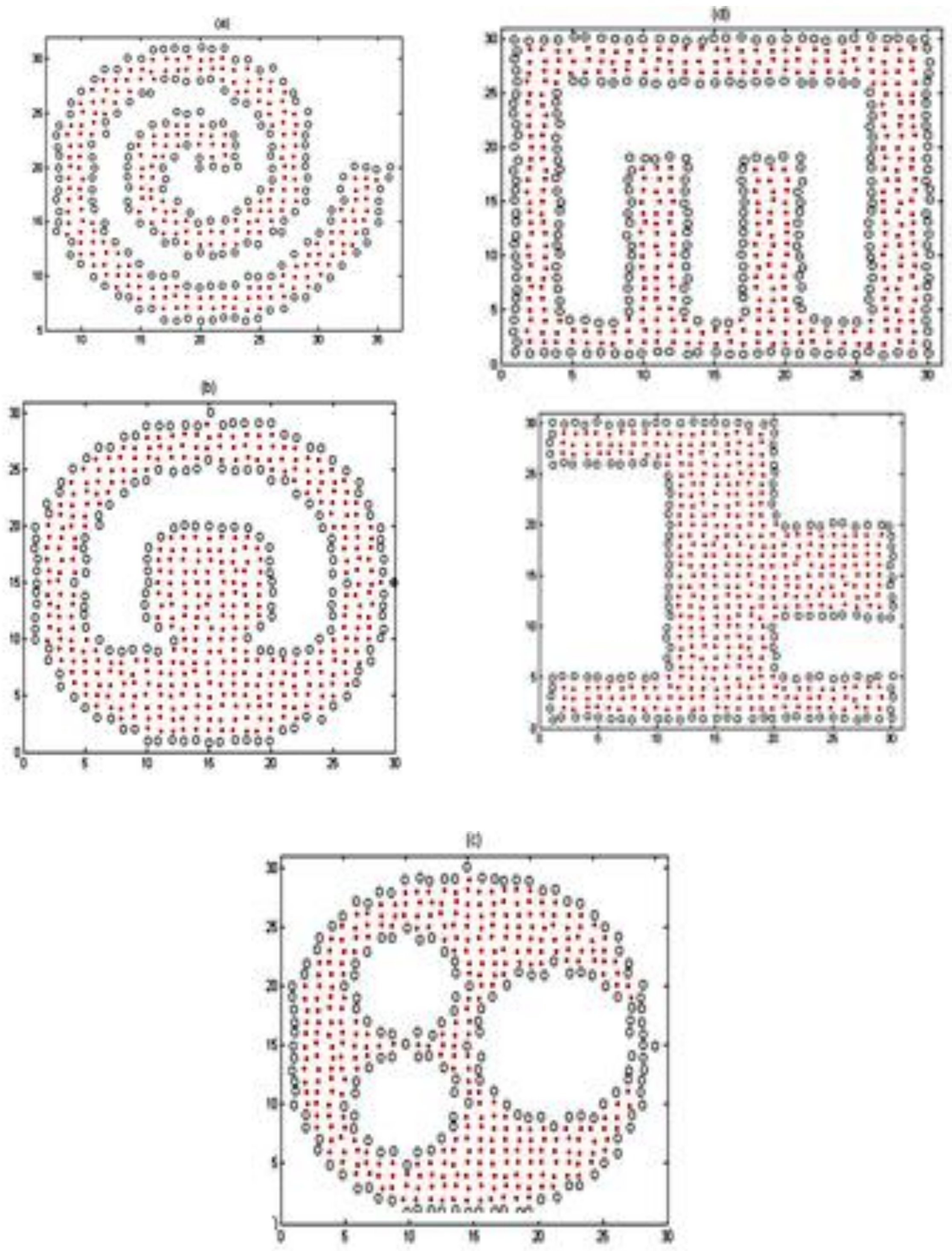


Figure 9.10: Boundary detection results for different network shapes: (a) a spiral network, (b) a circular network with a C shaped concave void, (c) a circular network with three voids, (d) a square network with an E shaped void, and (e) an odd shaped network.

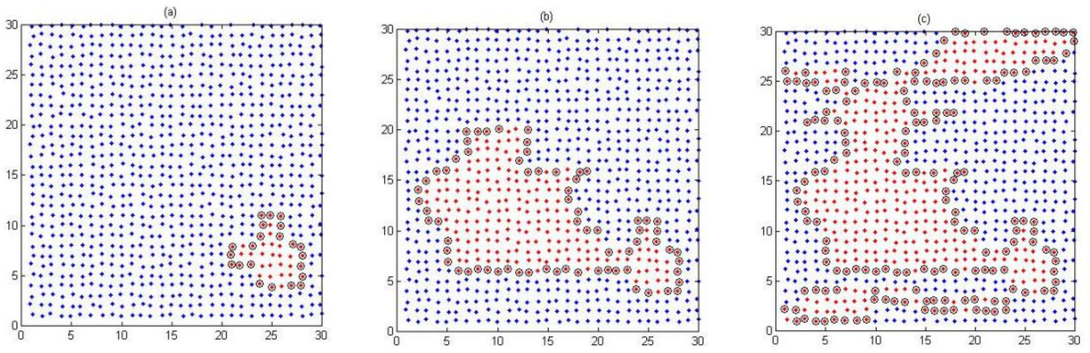


Figure 9. 11: An example of event boundary detection

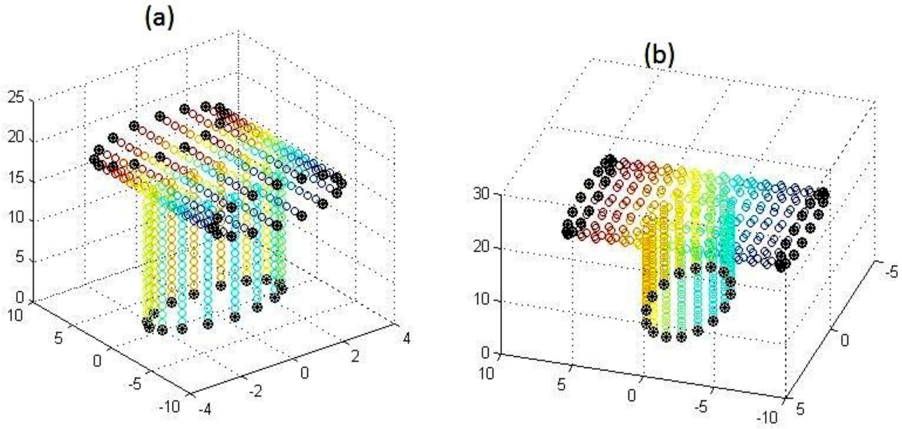


Figure 9. 12: Two different views of the 3D boundary identification of two perpendicular cylinders with no void (a), (b).

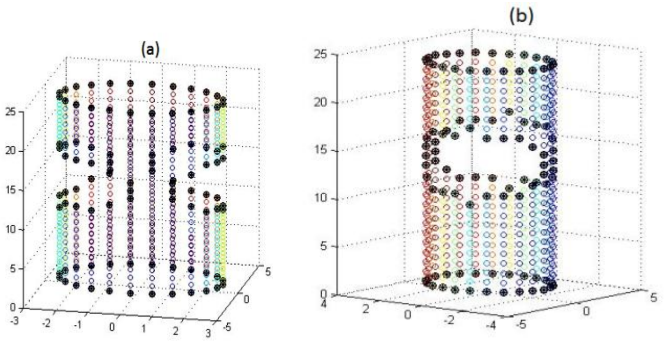


Figure 9.13: Two different views of the 3D boundary identification of cylinder with holes: (a), (b).

as illustrated in Figure 9.11. The value of $E\%$ of event boundary detection is zero, but the A% is 90%, when averaged over the three cases in Figure 9.11 (a), (b), and (c).

C. Boundary detection of 3D topologies

Next, we apply the boundary detection algorithm for the two networks considered in Section 9.4. The results are illustrated in Figures 9.12 and 13, respectively, for the T-joint and the cylinder with holes (cylinder with no voids had 900 nodes and 20 anchors, T-joint cylinders had 256 nodes and 20 anchors, and cylinder with two voids had 490 nodes and 15 anchors). The algorithm is able to detect the boundaries with A= 100% accuracy, and E= 0%.

9.7 Conclusion

An algorithm is presented for the generation of topology preserving maps of networks deployed on 3D surfaces. It uses virtual coordinates without the need for expensive and unreliable localization techniques. To our knowledge, no such algorithm exists. The main advantage of the proposed topology map is that it does not rely on localization based on GPS or other schemes. As the physical properties can be acquired from the topology map, and it is also based on the connectivity of the communication topology, a topology preserving map is more appropriate compared to physical map for many WSN applications. Routing and self-organization schemes actually perform better with topology maps than geographical maps for 2D networks [132]. The same can be expected to hold true for many 3D network protocols related to routing and self-organization. An example is the boundary detection scheme presented above. It illustrates that boundary nodes can now be detected with just the topology coordinates without the need for physical coordinates or resorting to analog measurements. It has to be noted that the boundary detection scheme presented is applicable in the topological coordinate domain as well as in the physical coordinate domain. We are currently working on a complex 3D surface structure modeled after a unit of pipes with multiple joints present in a refinery facility. We expect the results to be finalized in time for inclusion in the final research.

CHAPTER 10

NETWORK-AWARENESS VIA SELF-LEARNING AT WIRELESS SENSOR NODES

10.1 Introduction

The cost and size trends of sensor nodes, ranging from motes to active/passive RFIDs, point to future large-scale sensor-actuator networks where nodes in massive quantities will be deployed for monitoring and interacting with infrastructure and environment. Large-scale networks of nanosensors [5] and nanorobots [59] [118] exhibiting swarm behaviors [20] also present many intriguing applications related to healthcare, military, and infrastructure. Realization of the full potential of these technologies requires advances in theory, techniques, and algorithms that facilitate deployment of large-scale intelligent networks. Many of the current approaches for sensor networking have been developed considering a two-stage deployment process: the set up/organizing phase and the operational phase. Algorithms for network organization such as localization, clustering, and topology pruning in general do not evolve based on long-term learning, nor are replaced/switched based on knowledge that can be inferred during long-term operation.

We envision future sensor networks that evolve by long-term learning and inference, achieving over time increasing levels of network and sensed phenomena awareness, thus with time becoming smarter and better at what they do. We use the term “network/topology awareness” to indicate a node’s cognizance of the topology, shape, and boundary of the network and its place in that network. Upon initial deployment, the nodes would be quite oblivious of their environment, their individual place in the entire network, and the nature of the network that they belong to. However, over time, the nodes listen to the information disseminated in the network and infer additional knowledge and state information, leading the nodes and therefore the network to achieve network-awareness. In this research we take a step toward realizing such a vision with a learning approach that relies on information gleaned from ongoing packet transmissions (associated with routine network functions) to allow nodes to develop network-awareness over time.

In contrast to the traditional approaches where sensor networking algorithms aim to adapt the network to variations within limited contexts such as power availability, achieving network awareness allows networks to even switch from initial simple algorithms to more efficient and effective algorithms that require knowledge about the structure, connectivity, and shape of the network. Although a dedicated setup phase appears to be an option for achieving awareness, such a setup phase can be quite costly and, in certain cases, may not even be feasible. For example, when nodes operate by scavenging power from the environment, such as from ambient electromagnetic energy or vibrations [71], the amount of power (energy per unit time) a node can spend is severely limited. However, such nodes will be able to operate over very long time periods at this limited power level. A dedicated learning phase to achieve network awareness under such conditions can require a long initial time period during which the network is not available for the sensing task. With an evolving approach, the network attempts to carry out the sensing tasks with its current level of abilities, but becomes more effective with long-term learning.

A self-learning methodology is proposed for building a network in which nodes start operating in a network-oblivious form and, over time, infers knowledge and awareness of the network, its position in the network, as well as physical environment by listening to ongoing activities. With the proposed approach, the learning process at an individual node moves beyond sampling and storing information for subsequent table look-up or direct exchange of control information, as is the case of many traditional protocols, to actually inferring new knowledge from such directly available information.

Consider a newly deployed network of nodes, such as a set of nodes spread in an area. The nodes can find out about its immediate neighbors, but for carrying out its intended high-level application, it has to set up an infrastructure or rely on an infrastructureless routing protocol. We consider the latter case with use of random routing. Note that even in a structured network that has undergone a set-up phase, identifying sources/sinks for information of interest and dissemination of information involves random routing protocols such as rumor routing or diffusion [4]. While carrying out the application, the nodes develop a Virtual Coordinate System (VCS) specification based on information gleaned from packets that pass by [23][36]. Furthermore, each node keeps on learning about the network by making use of the source and destination addresses of the packets that it can hear. Each node, and thus the network, goes through three stages in gaining the network-awareness: (a) ascertaining virtual coordinates (VCs), (b) inference of topological coordinates (TCs) from VCs, and (c) formation of a map of the network within each node, containing information about other nodes, boundaries, and shapes. Information required for this evolution to occur is obtained by gleaning information about network nodes from ongoing packet transmissions.

Achieving network-awareness with existing methods would require node localization [4] integrated with flooding-based global distribution of the location information. Such a network set-up phase is cost-prohibitive in terms of packet transmissions and energy, and thus network-awareness has not received much interest in sensor networking. Topology awareness, with proposed evolutionary approach, is achieved without the need for costly and error-prone localization with analog measurements such as RSSI or time delay followed by network-wide distribution of such location information. Furthermore, as the VCs emerge via learning, the network can even switch to available VCS-based algorithms [23][36]. As learning proceeds and nodes start gaining TCs, nodes can then switch to much more efficient algorithms that use topological as well as physical information such as routing schemes [4][39] and boundary detection [42]. Once such a topology map is available, a node can identify where it and other nodes are with respect to the overall phenomenon being sensed, a feature useful in applications such as tracking chemical plumes.

The research includes results pertaining to the performance of the proposed self-learning scheme. The rates at which nodes acquire VCs and generate TCs are investigated. The effect of network parameters such as the number of nodes, and Time-To-Live (TTL) of the packets on performance is evaluated using 2D and 3D sensor networks. Consistency of topology maps generated independently by different nodes is investigated.

The rest of the research is organized as follows: Section 10.2 discusses related work. Then Section 10.3 explains how VCs are ascertained by the nodes and the learning-based strategy for topological map generation. Section 10.4 evaluates the performance. Section 10.5 addresses possible applications of the scheme, and Section 10.6 concludes our work.

10.2 Related Work and Background

We note that no scheme exists in literature for achieving network-awareness. Thus, in Section 10.2.A, we address existing literature related to use of learning techniques for resolving other specific sensor networking challenges. Section 10.2.B examines localization and routing schemes that incorporates network-learning approaches. Sections 10.2.C and 10.2.D review the background information on random routing schemes and VCS used for the proposed scheme.

A. Application-dependent learning schemes

An algorithm for distributed classification in sensor networks based on different classification types is presented in [45], which describes the sensed values as a Gaussian Mixture, and uses a machine-learning approach for classification. A scheme for allocation of limited resources such as energy and bandwidth in sensor networks, Self-Organizing Resource Allocation (SORA) [88], defines a virtual market in which nodes sell goods such as sensor readings or data aggregates in response to costs that are pre-programmed. Nodes take actions to optimize their profit subject to energy budget constraints. Using reinforcement strategies, the nodes adapt their operation over time in response to feedback from payments. A pattern-recognition-based event detection scheme is proposed in [12], in which the individual sensory measurements of sensor nodes are integrated into high-level event patterns to recover the state of the monitored environment. Having a costly training stage is common to all these schemes.

B. Learning-based localization and routing

Ambient Beacon Localization (ABL) [73] uses mobile sensors equipped with GPS to localize by exploiting their ambient physical environment. All nodes initially proceed through a bootstrapping phase that is computationally intensive to initialize their classifiers, which subsequently changes to adapt to the sensing phenomenon. ABL combines machine learning and free range beacon-based techniques to relate sensor data of known locations called ambient beacon points (ABPs). Supervised learning algorithms are used to allow mobile sensors to recognize ABPs.

Q-routing [19] is a routing algorithm based on reinforcement learning. Based on the packet transmission, each node develops a delivery time function that each node tries to minimize over its neighborhood. As destinations may vary, [19] proposes to train a neural network such that it will predict the next node to transmit the packet. The amount of information required and time taken for training the neural network is high, especially when many different destinations and sinks are involved. Reference [44] focuses on routing data to multiple, possibly mobile sinks. This algorithm is similar to that in [19] and uses a reinforcement-learning approach. A major disadvantage is that each different destination requires the network to learn again. A supervised learning-based approach is used in [119] to predict link quality, which is useful for routing. The supervised-learning approach, however, requires a large set of samples to train the model, and it needs to be trained again when link qualities change with time. Reference [95] presents an application of gradient ascent algorithm for reinforcement learning to a complex domain of packet routing in network communication. It is too computationally intensive for WSNs routing, especially as learning has to be repeated for each different destination.

In contrast to the schemes listed above, the method presented in this research engages in the learning process while carrying out the normal network functionality, and thus departs from the requirement of a separate and costly destination and source-dependant training phases. Moreover, the proposed learning scheme is not based on a specific application or a specific node/sink.

As the proposed scheme relies on random routing, VC generation, VC-based routing, and topology coordinate generation, their essential components are described next.

C. Random routing

Random routing algorithms where packets are forwarded according to a random selection made at a node are known for their simplicity and lack of a setup phase. Even in a structured network that has undergone a setup phase, identifying sources or destinations for information of interest and information dissemination often involve random routing. Many random routing protocols have been described in literature [4][17][18]. In this research we use rumor routing [18] as the scheme used for source or sink search. In rumor routing, a node selects a random neighbor as the next hop. Two types of randomly routed packets, called event agents and queries, exist. When a group of nodes experiences an event, each node probabilistically decides to send out a packet, called an event agent or simply an agent, informing the rest of the nodes in the network. On the other hand, if a node is looking for specific information about the sensed physical phenomenon, it sends out a packet called a query, requesting information. At each node, agents and queries are forwarded to randomly selected neighbors. To prevent a packet from circulating indefinitely, a packet is associated with a finite Time-To-Live (TTL), initialized at the source node with a maximum number of hops (H) the packet is allowed to traverse in the network. Moreover, each node that an agent or a query visits keeps a record about the event for future retrieval by a query, thus an agent and a query do not have to meet at a particular node at the same time. When the agent (query) meets query (agent) or a node, which has the information, a path is discovered between source(s) and sink nodes. In order to increase the probability of the path discovery, source (sink) nodes may send out more than one agent (query).

D. Virtual coordinate systems and routing

Virtual Coordinate based routing (VCR), also referred to as logical routing, has received much attention recently as it is more economical and efficient in WSNs than schemes based on physical location information, insensitive to localization errors, and achieves routability comparable to or even better than geographical routing schemes [69], but without the cost and complexity associated with localization.

Consider a network with N nodes in which a subset of M nodes is designated as anchors. These anchors are different from the anchors in physical position-based localization schemes in that VCS based anchors are un-localized and have the same capabilities as any other node in the network. Anchors may be selected randomly, or by an anchor placement mechanism. Each anchor floods the network so that each node in the network can evaluate the shortest distance, in terms of number hops, to each of the anchors. A node that is h_{A_j} hops away from the ${A_j}^{th}$ anchor will have h_{A_j} as the j^{th} coordinate. Thus i^{th} node is characterized by the VC vector, $P_{(i)}$ = [h_{iA_1} ... h_{iA_j} ... h_{iA_M}] of cardinality M. A few representative VC-based routing protocols are discussed next.

Scalable coordinate-based routing algorithm [101] uses a set of perimeter nodes as anchors. GF is used until a local minimum is reached, and then an expanding ring search is performed until a closer node is found or Time-To-Live (TTL) expires. In VC-assignment protocol (VCap) [22], virtual space is generated for the entire network with three farthest apart anchors. Insufficient number of anchors causes the problem of nodes having identical coordinates. As a solution, a packet is delivered to a zone of nodes with identical coordinates and then the final destination is sought using a proactive ID-based approach. Logical Coordinate based Routing (LCR) [23] uses GF followed by a backtracking scheme based on a furthest apart anchor placement. Convex Subspace Routing (CSR) protocol [36] uses a fundamentally different approach from the other schemes by dynamically moving to different convex subspaces of VCS to avoid local minima. A triplet of anchors is used at a time to define the convex subset, which is used for GF until a local minima occurs. Topology Preserving Map (TPM), in [37], refers to a map of a network obtained from VCs that preserves the connectivity topology of a network while capturing the physical properties of the network such as its shape, internal and external boundaries, and their relationships. According to [37], topology maps may be used in lieu of geographic coordinates for routing. Geo-Logical Routing [39] combines the advantages of topological domain [37] as well as in virtual domain overcoming local minima in each other's domain.

10.3 Network-awareness from Randomly Routed Network Traffic

This section presents a scheme for nodes in such a network to develop network-awareness via long-term learning. Consider a deployment of a set of sensor nodes that are initially oblivious to network connectivity and topology. Only information that a node is aware of the network is its own ID and its neighbors' IDs. Unlike conventional learning approaches discussed in Sections 10.2.A-10.2.C, this proposed method does not require a dedicated training phase based on training data, or flooding and retraining for each new source/sink pair. Each node goes through three stages in achieving network-awareness: (a) ascertaining the VCs (Section 10.3.A), (b) inferring TCs (Section 10.3.B), and (c) formation of a network map (Section 10.3.C). The notations used in the research are summarized in Table 10.1.

Table 10.1

Notations Used in Chapter 10

Notation	Description
N	Number of network nodes
N_i	Node i (current node or node under consideration)
N_s, N_d	Source, Destination
$\boldsymbol{M}$	Number of anchors
$\boldsymbol{K_h(N_i)}$	Set of nodes in $\mathbf{N_i}$'s h-hop neighborhood
$\boldsymbol{P_{N\times M} = [h_{iA_j}]_{ij}\ i = 1{:}N}$ and $\boldsymbol{j = 1{:}M}$	$\mathbf{N \times M}$ Virtual Coordinate Set of the entire network
$\boldsymbol{P}_{(i)} = [\boldsymbol{h_{iA_1}} \ldots \boldsymbol{h_{iA_j}} \ldots \boldsymbol{h_{iA_M}}]$	ith row of $\mathbf{P}$, corresponds to $\boldsymbol{N_i}$'s VCs.
$\widehat{\boldsymbol{P}}_{(i)} = [\widehat{\boldsymbol{h}}_{\boldsymbol{iA_1}} \ldots \widehat{\boldsymbol{h}}_{\boldsymbol{iA_j}} \ldots \widehat{\boldsymbol{h}}_{\boldsymbol{iA_M}}]$	ith row of $\widehat{\mathbf{P}}$, corresponds to $\boldsymbol{N_i}$'s approximated VCs via learning.
$\boldsymbol{P^{(i)}}$	ith column of $\mathbf{P}$, ordinate w.r.t. anchor $\mathbf{A_i}$ in VCS.
$[\boldsymbol{P_{SVD}}]_{\boldsymbol{N\times 2}} = [\boldsymbol{X_T}, \boldsymbol{Y_T}]$	$\mathbf{N \times 2}$ Topological Coordinate Set of the network
$\boldsymbol{Er}$	Error in $\mathbf{P_l}$ with respect to $\mathbf{P}$
H	Number of hops in Time-To-Live (TTL)
E	Energy per hop packet transmission
I	Number of packets that have already traversed the network
$\boldsymbol{L_p}$	Threshold on number of packets heard by a node without any VC updates before it can start using its VC. Termination criterion of self-learning VCS generation phase
$\boldsymbol{L_N}$	Threshold on number of unique VCs that have to be gathered by a node to generate topological coordinates
$\boldsymbol{Q}$	$\mathbf{L_N \times M}$ Matrix of unique nodes' virtual coordinates
$\boldsymbol{B}$	Original length of the packet in Bytes

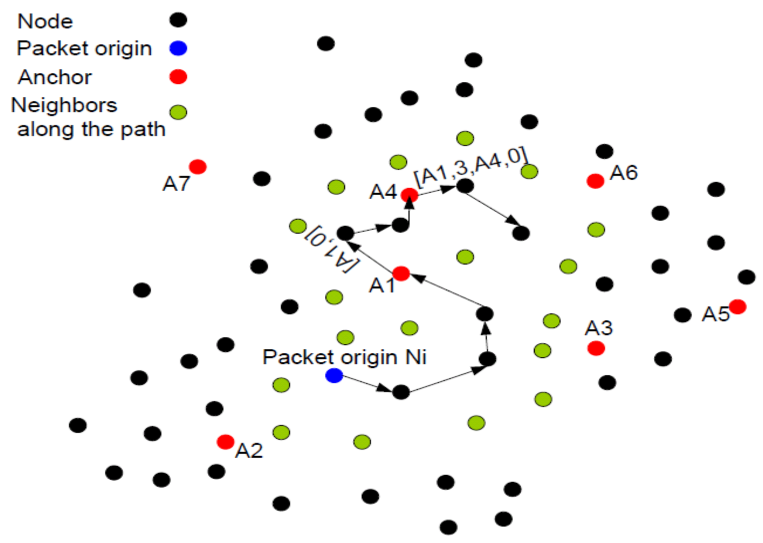


Figure 10.1. An example network indicating the virtual coordinate propagation in the VC developing stage. Network has 7 anchors.

A. Virtual coordinate generation via self-learning

The proposed self-learning strategy for VC generation is discussed next. At initial deployment, a network node N_i can discover its set of one-hop neighbors $K_h(N_i)$. The random routing scheme described in Section 10.2.4 is assumed to be used by sensing application initially for dissemination and discovery of sensed information, although many other strategies, random or otherwise, can be used for this purpose. Different nodes send out packets, i.e., agents/queries, on random routes as illustrated in Figure 10.1.

The mission of the VC learning process is for each node to ascertain the shortest path hop distances to the anchors. A node decides on its own to become an anchor with a certain probability p_A. VC-based routing and topology mapping can be successfully carried out with randomly selected 2-5% of the nodes as anchors [23][37], thus a value p_A<0.05 suffices.

```
% Algorithm at a node for random routing based virtual coordinate propagation. Any node that
relays or passively hears a packet updates its estimates for virtual coordinates
INPUTS:
    $\boldsymbol{Array\_A}$ → stored anchor ID of anchor $\boldsymbol{A_k}$at node
    % Set of anchor IDs stored at node
    $\boldsymbol{\hat{h}_{iA_k}}$ →stored hop distance to anchor $\boldsymbol{A_k}$ at a node $\boldsymbol{N_i}$ % Set of hop distance estimates to
    anchors at node
    $\boldsymbol{Packet\_A}$=[$\boldsymbol{A_k}$];$\boldsymbol{A_k}$:anchor ID of anchor $\boldsymbol{A_k}$ in the packet
    % Set of anchor IDs stored in packet
    $\boldsymbol{Packet\_H}$=[$\boldsymbol{H_k}$]; $\boldsymbol{H_k}$:hop distance to anchor $A_k$ in the packet % Set of hop distances to
    corresponding anchors in packet header
OUTPUT:
    Estimated VC, $\boldsymbol{\hat{P}}_{(i)}$ = [$\boldsymbol{\hat{h}_{iA_1}}$ ... $\boldsymbol{\hat{h}_{iA_j}}$ ... $\boldsymbol{\hat{h}_{iA_M}}$] of a node $\boldsymbol{N_i}$
1   For all anchor IDs NOT already saved in node $\boldsymbol{N_i}$
2       Create a new entry for Anchor ID and hop count
3   End
4   For all anchor IDs already saved at node
5    %if the received VC is less
6       If $\boldsymbol{\hat{h}_{iA_k}}$> **Packet_H(k)** +1
7               $\boldsymbol{\hat{h}_{iA_k}}$= **Packet_H(k)** +1
8               Reset **No_updates_Flag**  % Have updates
9       %if the Stored VC is less
10      Elseif $\boldsymbol{\hat{h}_{iA_k}}$< **Packet_H(k)** +1
11              Set **No_updates_Flag(k)** % No updates
12      %update the packet VC iff the node is to relay packet
13
14              If $\boldsymbol{N_i}$ is a relaying node
15                      **Packet_H(k)**= $\boldsymbol{\hat{h}_{iA_k}}$ +1
16              End
17      %if the Stored VC and received VC are the same
18      Elseif  $\boldsymbol{\hat{h}_{iA_k}}$ == **Packet_H(k)**+1
19            Set **No_updates_Flag(k)** % No updates
20      End
21  End
22  If **No_updates_Flag** = =1 for all ordinates
23      %count the number of consecutive packets for which there is no updates
24      Counter =Counter +1;
25      Elseif **No_updates_Flag** = = 0
26      % reset  Counter if not consecutive
27      Counter = 0
28  End
29  %Node assumes its VCs is correct when $\boldsymbol{L_p}$ number of packets have been received without
    updates
30  If **Counter** $> \boldsymbol{L_N}$
31      Stop updating VC
32  End
```

Figure 10.2. Algorithm for VC estimation at a node.

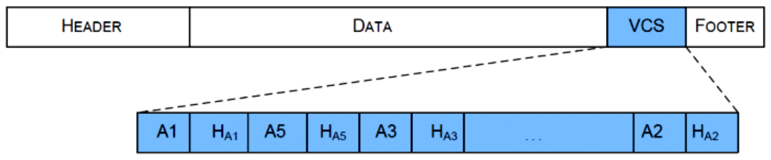


Figure 10.3. Format of a packet in WSN with the added information of VCS. Length of the VCS vary.

Alternatively, a set of nodes may be pre-designated to become anchors. When an anchor node A_k receives a packet (an agent or a query) for relaying or if it generates one, $\boldsymbol{A_k}$ appends a tuple $\{A_k$ ID, $\hat{h}_{iA_k}\}$ containing its self ID and a hop count field to the packet, and forwards the packet to a randomly selected neighbor from $K_h(N_i)$. Each forwarding node increments $\hat{h}_{iA_k}$ by one. $\hat{h}_{iA_k}$ is the distance from A_k to the current node along the path traversed. It is an upper bound for the shortest hop distance between the two nodes, and the lowest value of such bounds from multiple packets converges to a good estimate for the hop distance to the anchor nodes. Moreover, if a packet containing hop count to A_k, passes through A_j then A_j will append its ID and hop distance to the packet as $\{$ A_k ID, $\hat{h}_{iA_k}$, A_j ID, $\hat{h}_{iA_j}\}$ to begin generating ordinates with respect to A_j. Packet format is illustrated in Figure 10.3.

When a node receives a packet with anchor ID and hop count tupples, for instance $\{$ A_k ID, $\hat{h}_{iA_k}$, A_j ID, $\hat{h}_{iA_j}\}$, it updates the estimated distance corresponding to each such anchor as follows. If it is the first time the node hears about a particular anchor, it stores the anchor ID and corresponding hop count received incremented by one hop (see lines 1-3 in Figure 10.2). If the node already has a hop count corresponding to a particular anchor, the new value is stored only if it is less than the existing value for the hop count (see lines 6-7 in Figure 10.2). Then the node adds one hop to the hop count(s) in the packet and forwards it to a neighbor (see lines 14-16 in Figure 10.2). When a node forwards a packet to a neighbor, the rest of the neighbors passively listening to the channel will also update their coordinates (see line 6-21 in Figure 10.2).

Initially the coordinates received by nodes, $[\hat{h}_{iA_1} \dots \hat{h}_{iA_j} \dots \hat{h}_{iA_M}]$, are not the shortest path distances to the anchors, but as many packets traverse the network, the nodes keep on updating the ordinates, and the values converge to the shortest path distances to anchors, i.e., $[h_{iA_1} \dots h_{iA_j} \dots h_{iA_M}]$. For instance, consider the sample network with seven anchors A_1, …, A_7 as shown in Figure 10.1. Assume node N_i originates a packet and randomly forwards it to a neighbor. When the anchor A_1 receives the packet, it appends the tuple $\{A_1,0\}$ and forwards it to a neighbor. When the same packet goes to anchor A_4 and it does not has information received about A_1, it will store the distance to anchor A_1, i.e. $\{A_1,2+1\}$. Then it updates the distance to A_1 as $\{A_1,3\}$, appends $\{A_4,0\}$ and forwards to a neighbor as in random routing protocol. Each node maintains a vector $[\hat{h}_{iA_1} \dots \hat{h}_{iA_j} \dots \hat{h}_{iA_7}]$ and updates it based on the path information that it receives until it gains the actual VC $[h_{iA_1} \dots h_{iA_j} \dots h_{iA_7}]$, i.e., the shortest hop distance from node itself to anchors. Moreover, green nodes are the neighbors of the nodes visited by the packet. Not only having the nodes that relay the packet but also such nodes that hear the packet update their estimates speeds-up the learning process.

Now the critical question is how a node figures out whether it has converged to the correct ordinates. A node starts using the VCs after it forwards L_p number of packets and not having VC updates consecutively; this is discussed in detail in Section 10.4. Implementation of the criterion is in line 22-32 in algorithm in Figure 10.2.

```
%Algorithm implemented at each node to evaluate its topological coordinates.
INPUT
        Node_Count($\mathrm{N_i}$)%Node $\mathrm{N_i}$, stores $L_N$ number of unique source and destination VCs in
        an array Node_Count($\mathrm{N_i}$)
OUTPUT
        $V^{(2)}, V^{(3)}, \mathrm{V}^{(4)}$ for TPM at $N_i$
1       If Node_Count($\mathrm{N_i}$) $\leq L_N$
2           $flag$_$\mathrm{N_d} = 0; flag$_$\mathrm{N_s} = 0$;
        End
3        %Check whether the destination and/or source are a new entries
4       For x = 1: Node_Count($\mathrm{N_i}$)
5           If  Node_Entry($\mathrm{N_i}$) == $\mathrm{N_d}$
6               $flag$_$\mathrm{N_d} = 1$; %Not a new entry
7           Elseif  Node_Entry($\mathrm{N_i}$) == $\mathrm{N_s}$
                    $flag$_$\mathrm{N_s} = 1$; %Not a new entry
8           End
9       End
10      %If destination is a new entry
11      If flag_$\mathrm{N_d}$ == 0
12          Node_Count($\mathrm{N_i}$) = Node_Count($\mathrm{N_i}$) + 1;
13          Node_Entry($\mathrm{N_i}$) = $\mathrm{N_d}$
14      If Node_Count($\mathrm{N_i}$) $\geq$ R
15          ready to generate TP coordinates;
16          TP_Coordinates_ready($\mathrm{N_i}$) = 1;
17      End
18      End
19      %If destination is a new entry
20      If flag_$\mathrm{N_s}$ == 0
21          Node_Count($\mathrm{N_i}$) = Node_Count($\mathrm{N_i}$) + 1;
22          Node_Entry($\mathrm{N_i}$) = $\mathrm{N_s}$
23          If Node_Count($\mathrm{N_i}$) $\geq L_N$
24              ready to generate TP coordinates;
25              TP_Coordinates_ready($\mathrm{N_i}$) = 1;
26          End
27      End
28      %If node has R number of nodes' VC, it will generate topological coordinates.
29      if  TP_Coordinates_ready($\mathrm{N_i}$)==1
30      %TC generation -  get the matrix Q which is R $\times$ M
31      Q = Node_Entry($N_i$)
32      [U S V] = svd(Q);
33      %------Topological coordinate of node $\mathrm{N_i}$---------
34      %For a 3D network
35      $[X^T, Y^T, Z^T]_{(i)} = P_{(i)}[\mathrm{V}^{(2)}, \mathrm{V}^{(3)}, \mathrm{V}^{(4)}]$
36      %For a 2D Network
37      $[X^T, Y^T]_{(i)} = P_{(i)}[\mathrm{V}^{(2)}, \mathrm{V}^{(3)}]$
38      End
```

Figure 10.4. Topological coordinate generation algorithm implemented at each node.

Proposed VC-learning scheme is not merely an alternative for flooding. Nodes gain the ability to infer based on what they receive or what they hear, by intelligent decision-making capability. When the entire network has the VCS available, the network nodes switch from random routing to using VCS-based routing schemes (discussed in Section 10.2.4) that are much more efficient and effective. The source address and destination address in packets now will contain VCs.

B. Distributed self-generation of topological coordinates from virtual coordinates

Once the VCs are estimated, moving another step forward, self-learning process at a node turns to develop TC of the network via listening to ongoing transmissions. This allows the network nodes to move from VC-based algorithms to more sophisticated TC-based algorithms.

With each node characterized by a vector of VCs, with distance to each of the M anchors ($M << N$), the next step is for each node to develop a linear transformation that translates the VCs of a node to the corresponding coordinates on a topology map. Let P be the $N \times M$ VC matrix of the network. Reference [37] proposes three possible options of generating TCs of the network: (a) based on P, the entire VCS; (b) based on VCs of anchors; and (c) based on VCs of a randomly selected set of nodes. In this research we use (c), VCs of L_N nodes.

Let the $L_N \times M$ matrix of the VCs be Q. Using Singular Value Decomposition on Q;

$$Q = USV^T$$

where U and V are unitary matrices of size $L_N \times L_N$ and $M \times M$ respectively and S is a matrix with singular values as diagonal entries. Second and third colomns of basis V are used to generate the TC of node N_i , i.e.,

$$[X^T, Y^T]_{(i)} = P_{(i)}[V^{(2)}, V^{(3)}] \qquad (1)$$

$P_{(\boldsymbol{i})}$ is the VC of $N_{\boldsymbol{i}}$. $V^{(\mathbf{2})}$and $V^{(3)}$ are second and third column vectors of basis V.

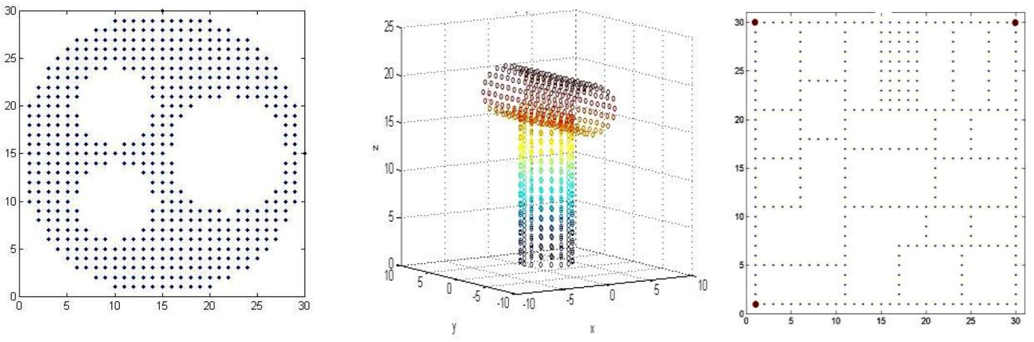


Figure 10.5. (a) Circular network with 496 nodes. (b) Network deployed on a 3-D surface (T joint) Each cylinder has a radius 2.54 units, height 16 units, and is covered with a grid of 512 nodes. (c) A network of 343 nodes mounted on walls of a building. Three anchors are manually placed at three corners as shown. Figure 10.4. Topological coordinate generation algorithm implemented at each node.

As a node forwards or hears packets, it stores L_N number of unique destination and source addresses as matrix Q. Once the required number entries are available, an individual node can generate its own matrix V, thus the transformation for converting VCs to topology coordinates. Note that in this case, since every node has VCs of a different set of nodes, V matrix at different nodes is not the same.

In the ideal case, when Q matrices at all the nodes are good representative samples of the entire network's VCS, hence, V of $N_{\mathbf{i}}$ can expect to be a rotated version of V of node $N_{\mathbf{j}}$.

Reference [37] addresses the TPM generation in 2D networks. To generate the topology maps of 3D networks, we extend the scheme using

$$[X^T, Y^T, Z^T]_{(i)} = P_{(i)}[V^{(2)}, V^{(3)}, V^{(4)}] \quad (2)$$

The algorithm implemented at node N_i is summarized in Figure 10.4. Line 1-27 is the implementation of a node collecting unique source and destination VCs. TC generation is as in line 28-38.

Computational complexity of SVD-based evaluation of $[V^{(2)}, V^{(3)}]_{(\mathbf{i})}$ for 2D networks [37] and $[V^{(2)}, V^{(3)}, V^{(4)}]_{(\mathbf{i})}$ for 3D networks at node N_i can be significantly reduced by using eigenvalue decomposition. The complexity in evaluating the full V matrix using eigenvalue decomposition is an upper bound on this computation complexity. Thus, upper bound for number of computations is $(4M^2L_N + 8M^3)$ [37] while that for memory usage is $(2 \times M) + (L_N \times M)$, where L_N is the number of node addresses used for V matrix estimation. Note that we require just second and third column of matrix V.

C. *Topology map from virtual coordinates*

Each node will now attempt to construct its own view of the network, in essence a map of the network nodes becoming network-aware, i.e., gaining the knowledge of the position of the node with respect to other nodes, the boundary of the network, the location and shapes of voids, etc. An example of a useful map for a node to acquire would be the physical map of the network. However, obtaining such a map requires resorting to distance measurements (via GPS, RSSI, or flight delay) and broadcasting such information. Network-awareness can be gained using other non-linear mappings as well, for example, birds-eye views and fish-eye views are real-life examples of distorted, yet practically useful maps. Our approach produces a topology map that provides a distorted view of the physical map, yet is more useful for tasks such as routing as it is based on connectivity.

As discussed above, a node N_i is able to generate its own V, and therefore convert the any given VC tuple to the corresponding TCs. Next step then is for each node to develop a view of the network and its place in the network, essentially generating its own map of the network. As a node learns the VCs corresponding to the sources and destinations that are embedded in the packets it relays or it overhears, it adds those to its network topology map. Thus with time, the topology map of the network at each node will gradually build up, thereby increasing its awareness of the network.

In practice, the TPMs generated at different nodes would be different from each other due to the fact that V is not unique. However, what is important for network awareness is the consistency among the maps at different nodes. We evaluate the consistency among the maps in the next section.

As the topological map is generated based on the source and destination addresses of routine network packets that each node hears or forwards, no additional transmission cost or energy is spent for topology map generation.

10.4 Performance Analysis of Self-learning Scheme

The performance of the proposed scheme for network-awareness is evaluated next. Section 10.4.A presents the effectiveness and efficiency of VC generation using self-learning during the random routing phase. Section 10.4.B demonstrates the major outcome of this work, that of nodes becoming aware of network topology and its place in the topology. Section 10.4.C uses routing as an example to demonstrate how performance of the network improves as nodes become topology aware. Three networks representative of different applications shown in Figure 10.5 are used for the evaluation: a) a circular network with three voids [37], b) a sensor network deployed on a 3D surface, and c) a network deployed along the walls in a building [37]. MATLAB® 20011a was used for the computations. In all the networks, a node has a communication range of one unit, resulting in up to four neighbors per node. Though a real sensor network implementation would have been the ideal evaluation, the proposed scheme is applicable in large WSNs where the number of nodes required is over 500, thus the real implementation is expensive.

A. *Self-learning based VC generation*

Only the results corresponding to the circular network in Figure 10.5 (a) are presented for the purpose of evaluating the convergence of [$\hat{h}_{iA_1}$ ... $\hat{h}_{iA_j}$ ... $\hat{h}_{iA_M}$] to VCs, as the observations for the other two networks are consistent. In addition to the configuration shown in Figure 10.5 (a) with 496 nodes, we also consider two networks with similar shape with 1081 and 2048 nodes respectively to evaluate the scalability. Corresponding to a probability 0.02 at which a node decides to become an anchor, 10, 20, and 40 randomly selected nodes served as anchors for the networks with 496, 1081, and 2048 nodes respectively. Randomly selected source nodes were considered to emulate circulation of packets in the network searching for information using random routing. As explained in Section 10.3.A, such packets cause the nodes to generate the VCs. Convergence of this distributed process is evaluated using the number of packets it takes for the nodes to learn the VCs and the accuracy of the generated VCs. Figure 10.6 illustrates the error in evolved VCs ($\hat{P}$) compared to correct VCs (P), averaged over each node and anchor, defined as

$$Error\ in\ VCS\ (E_V) = \frac{1}{MN}\sum_{i=1}^{N}\left(\sum_{j=0}^{M}|h_{iA_j} - \hat{h}_{iA_j}|\right) \tag{3}$$

E_V is simply an indication of average hop error in a VC at a node. In the circular network with voids containing 496 nodes, the error of approximated VCS, E_V approaches zero after about 200 rumors with TTL 100 have traversed in the network. When TTL is 10 times lower, the same network requires over 1500 packets to achieve error free VCS, i.e., $E_V = 0$. Figure 10.6 gives the number of packets required to get error-free VCS under different TTLs. All these cases confirm the ability of the nodes to converge to the set of VCs by self-learning.

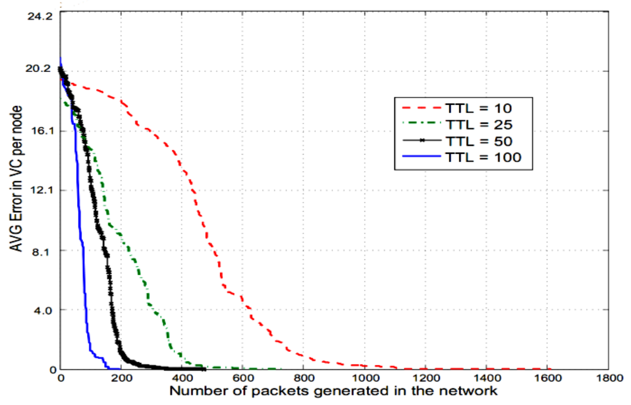


Figure 10.6. Error in entire VCS of the network (E_V) averaged over 10 configurations as the number of packets traverse in the network increases. Network with 496 nodes and 10 anchors.

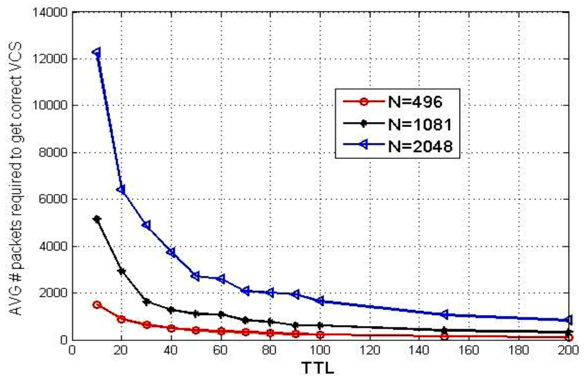


Figure 10.7. Number of packets required for all the nodes to generate error free VCS averaged over 10 different configurations as TTL varies. Network has 10,20 and 40 anchors respectively for N=496 , 1081 and 2048.

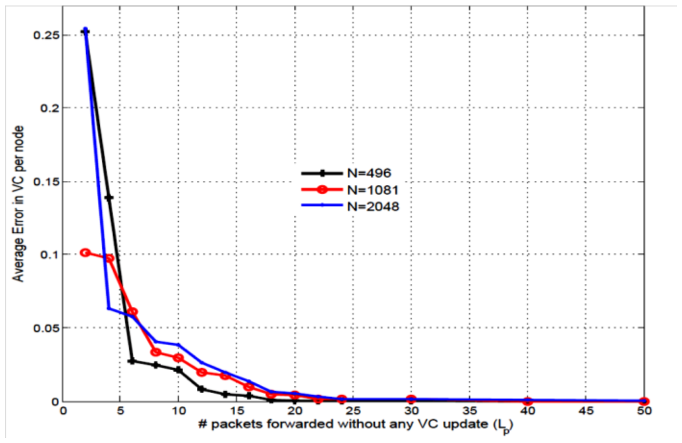


Figure 10.8. Error in VCS vs. number of packets forwarded by each node without any VC updates L_p, averaged over 10 anchor placements.

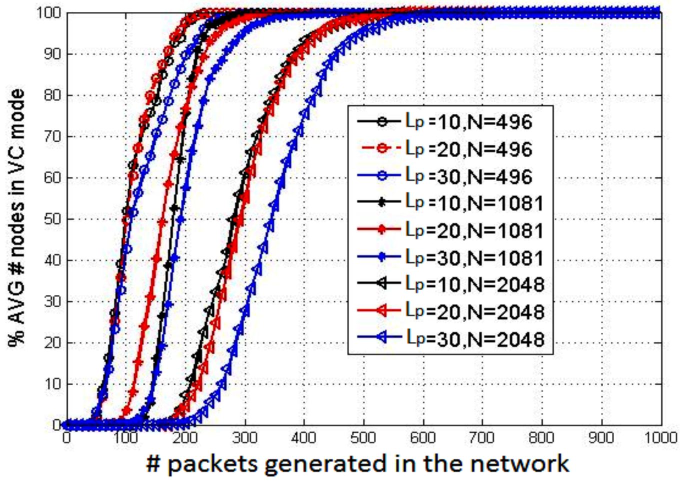


Figure 10.9. Percentage of nodes in VC mode vs. the number of packet traversed in the network averaged over 10 random anchor placements. TTL is 100.

TTL has to scale with the network size for packets to be able to span the network. Thus, we set TTL to be proportional to the diameter of the network, i.e.,$\sqrt{N}$. For the rest of the simulations, TTL for networks of N, 496, 1081, and 2048 are set to 100, 150, and 200 respectively. From Figure 10.7, the number of packets required to get accurate VCS for the networks of size 496, 1081, and 2048 are 232, 397, and 830 with the selected TTLs, respectively.

The next challenge is how a node can determine on its own whether it has correct VCs, since the proposed self-learning scheme is implemented in a distributed fashion. When the number of consecutive packets a node has heard that did not cause any updates of its VC estimates, exceeds a threshold L_p, a node assumes that it has correct VCs. In order to decide a value for L_p, the variation of error in generated VCS (E_V) vs. L_p is evaluated for three different network sizes in Figure 10.8. It can be seen that there exists a L_p for which E_V goes to zero. For L_p at 30, all the three networks achieve almost error-free VCS. When a node has determined that it has accurate VCs, it switches to the VC mode of operation.

Variation of the percentage of nodes in the VC mode with the number of packets transmitted in the network under different L_p settings is illustrated in Figure 10.9. Moreover, scalability is examined using networks of sizes 496, 1081, and 2048, for $L_p = 10$, 20, and 30. Consider the network of 496 nodes. When L_p changes from 10 to 30, the number of packets required for all the nodes to get into VC mode is more or less the same. As observed in Figure 10.8, the larger L_p is, the smaller the error in VCS evaluated using Eq. (3). A network of size 496 requires around 300 packets for all the nodes to get their VCS, while for 1081 nodes, the number of packets required is around 400. Moreover, a network of size 2048 requires about 700 packets to achieve 100% VC mode nodes. L_p is set to 30 in the rest of the simulations because at L_p is 30; the error of the generated VCS (from Figure 10.9) is 0.0001, 0.002, and 0.004 in 496, 1081 and 2048- node networks.

Next, we consider the additional energy consumption due to lengthening of packets due to the additional field of anchor IDs and hop count required by the learning scheme.

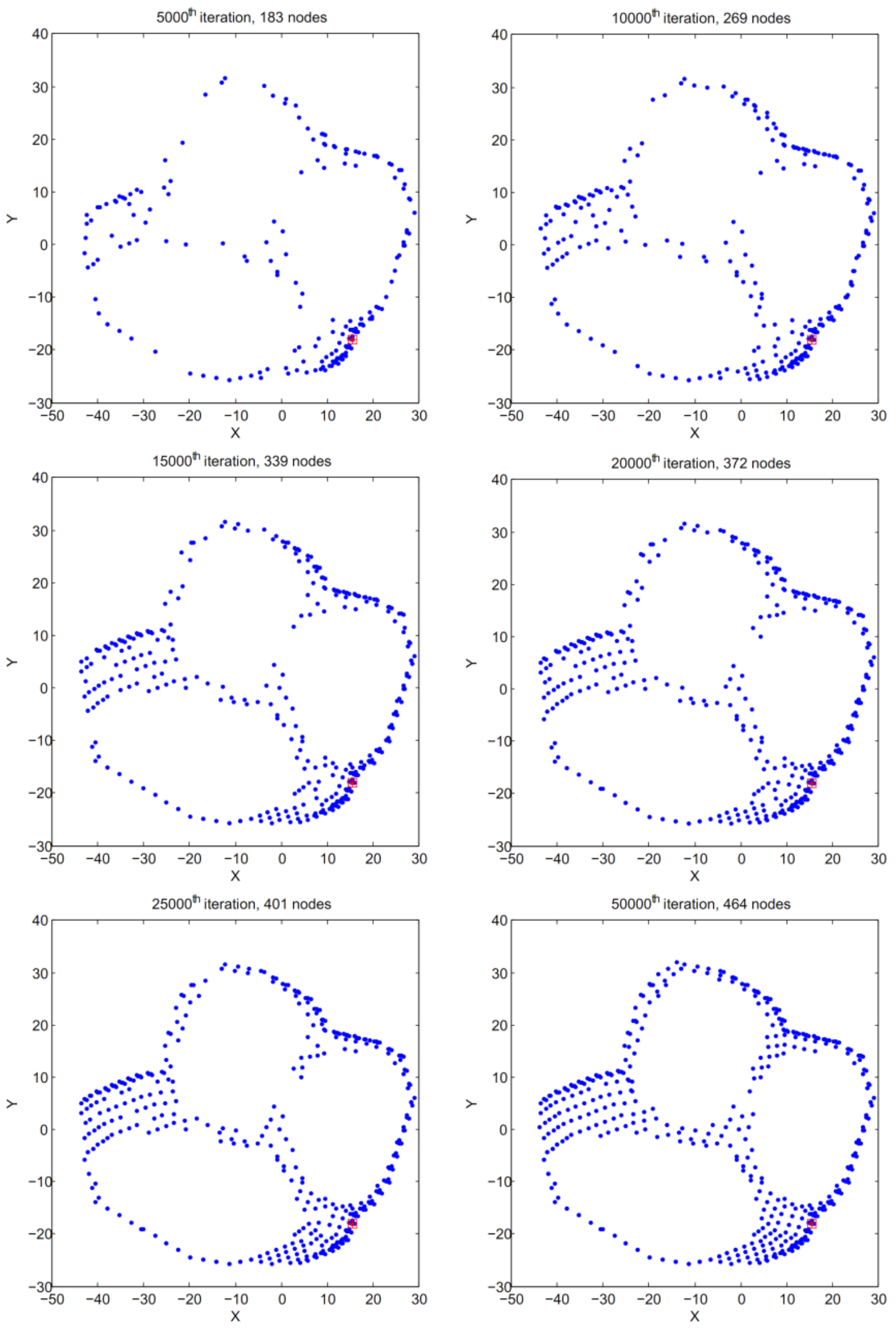


Figure 10.10. Topology preserving map development at a sample node (identified by color red). Number of anchors is 10 and total number of nodes is 496. TTL of a packet is 100.

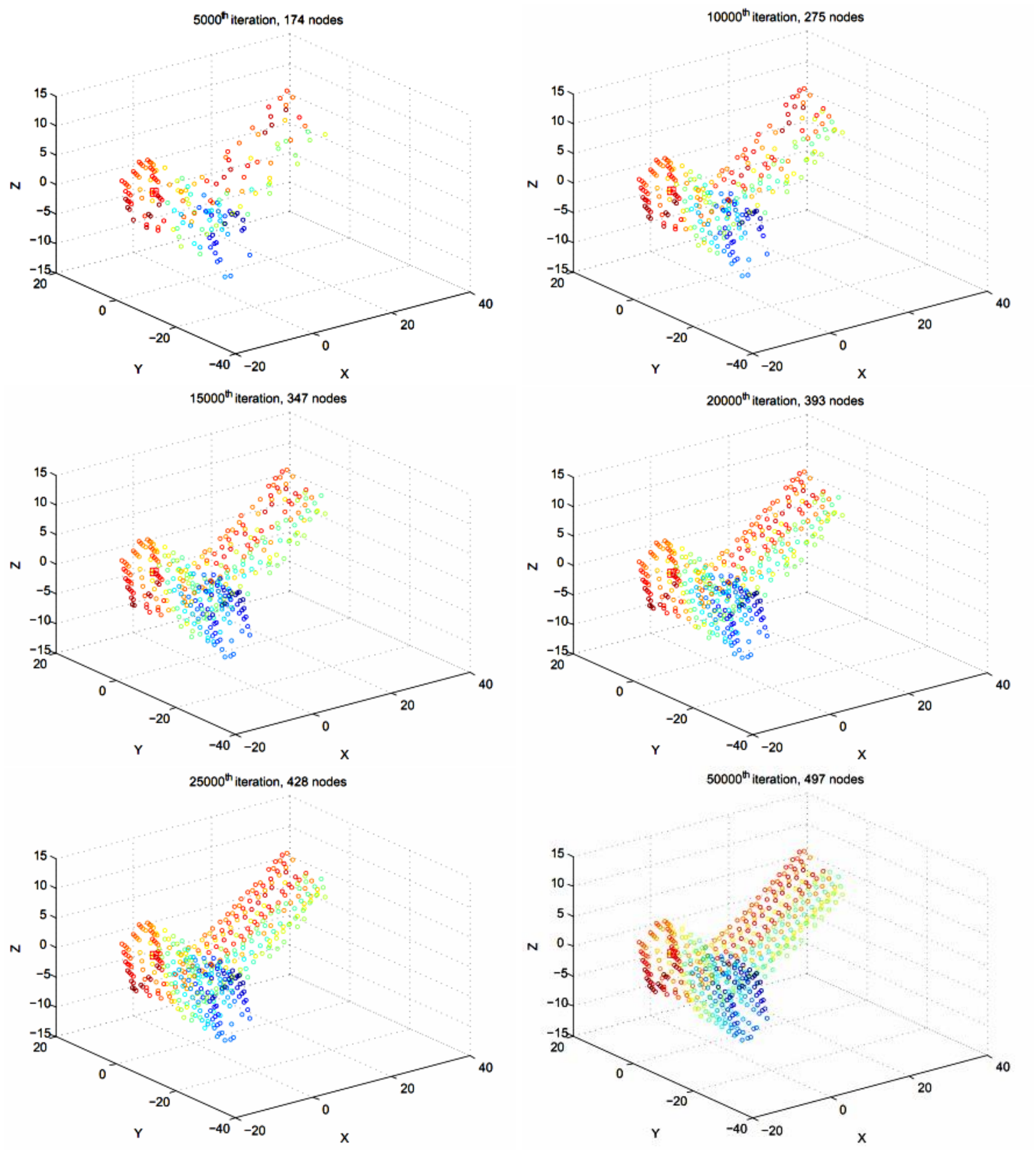


Figure 10.11. Topology preserving map development at a node of a 3D T-joint assuming it store the destination topology coordinate of every destination node that goes through it. Number of anchors is 10 and total number of nodes is 512.TTL of a packet is 100.

The traditional method for VC generation involves multiple flooding, and the communication complexity is of the order of $O(MN)$, where M is the number of anchors and N is the number of network nodes. In the proposed scheme, there are no additional packets involved in VC generation since VCS is generated using routine network traffic. Energy consumption of traditional VC generation is $E \times O(MN)$, where E is the energy consumption per packet. In the evolving VC generation, the probability of a packet visiting all the anchors within its TTL is very low. The probability of a node becoming an anchor is $\frac{M}{N}$. Packet with TTL = H visits H nodes. Hence the expected number of anchors that a packet visits is $M/N \times H$. So the expected number of bytes padded to a packet is $2M/N \times H$, assuming one byte per anchor ID and one for its ordinate.

B. Evolution of topology awareness in self-learning networks

As explained above, when a node decides that its VCs have converged, based on the number of packets that passes by without any update to its VCs, it enters the VC mode, i.e., it is capable of using any VCS-based algorithm. One such is TPM generation. A node in this mode identifies itself using its VCs, and thus packets generated by this node contain the VCs. Each node that hears a packet with VCs can store the corresponding VCs locally to create an entry for the matrix Q required for topology coordinate calculation. A node collects L_N unique virtual addresses from the packets that it forwards or hears. The results presented below use $L_N = 10$. As explained in Section 10.3.C, eigenvalue decomposition of Q matrix yields $[\mathrm{V}^{(2)}, \mathrm{V}^{(3)}]$ also $\mathrm{V}^{(4)}$ for 3D networks, which can now be used to convert any VC specification, including its own, to the corresponding topology coordinates using Eq. (1) for 2D and Eq. (2) for 3D networks.

The node is now in possession of a transformation capable of generating topology coordinates from the VCs of any node. Thus, the node listens to ongoing packet transmissions to gather VCs corresponding to different nodes, calculate the corresponding

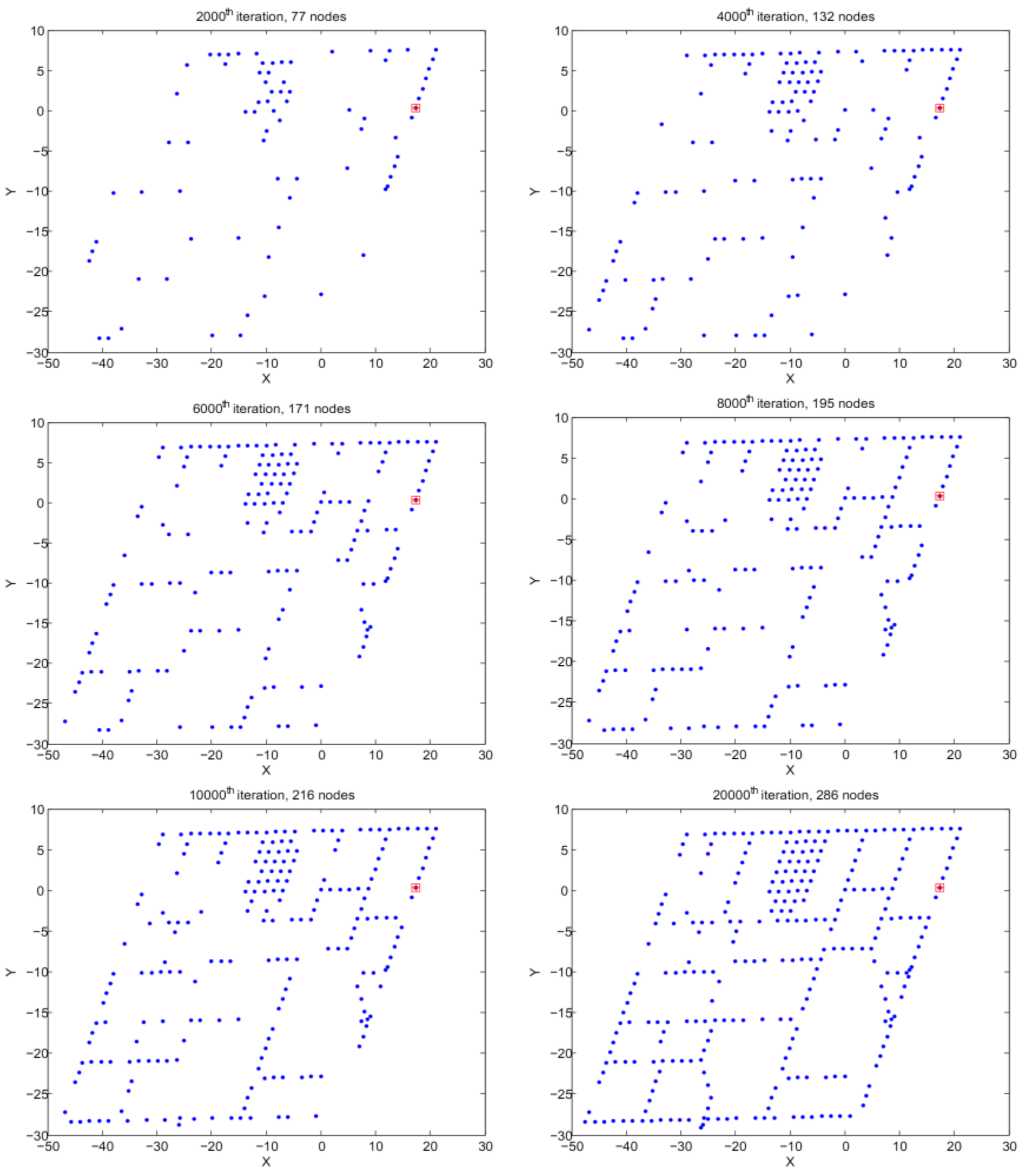


Figure 10.12. Topology preserving map development at a node of a 3D T-joint assuming it store the destination topology coordinate of every destination node that goes through it. Number of anchors is 10 and total number of nodes is 512.TTL of a packet is 100.

topology coordinates, and add them to the topology map of the network. The map at a node initially starts with L_N nodes but will expand, accommodating new nodes that it comes to know of. Figure 10.10 illustrates the evolving view of the topology map at one of the nodes in the circular network with three voids, which is representative of what happens at each node. As time evolves, the node finds out about more and more nodes, and the topology map completes. The variation of the percentage of nodes known to the observing node *V*s. the number of packets that have traversed the network is shown in Figure 10.13.

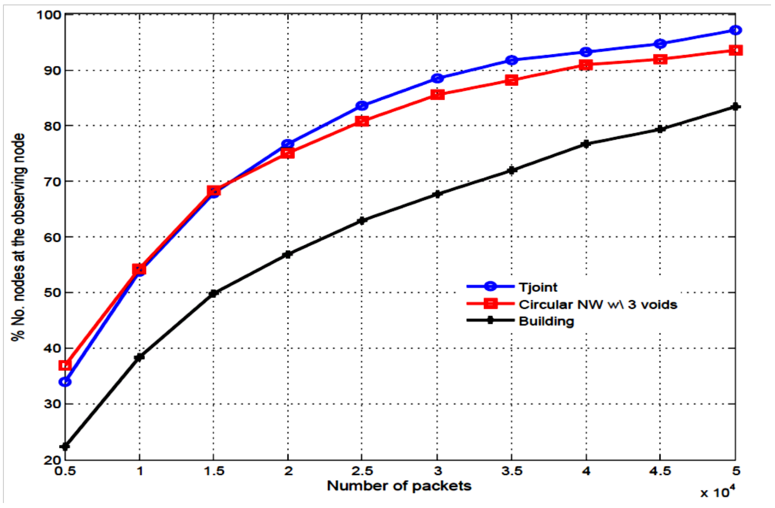


Figure 10.13. Variation of percentage number of nodes at the observing node Vs. number of network packets, for circular network with 3 voids (N=h 496, M=10, H=10).

Next, the effectiveness of the proposed learning scheme is illustrated for the 3D network (T-joint of Figure 10.5 (b)) and the 2D building network (Figure 10.5 (c)). The T-joint is illustrative of an example such as a sensor network for monitoring the structural integrity of a junction in an oil pipeline or a tunnel. Each cylinder has radius 2.54 units, height 16 units, and is covered with a grid of 512 nodes, each with a communication range of 1. Ten randomly placed anchors were used, and TTL set to 100 hops. Different stages of topology

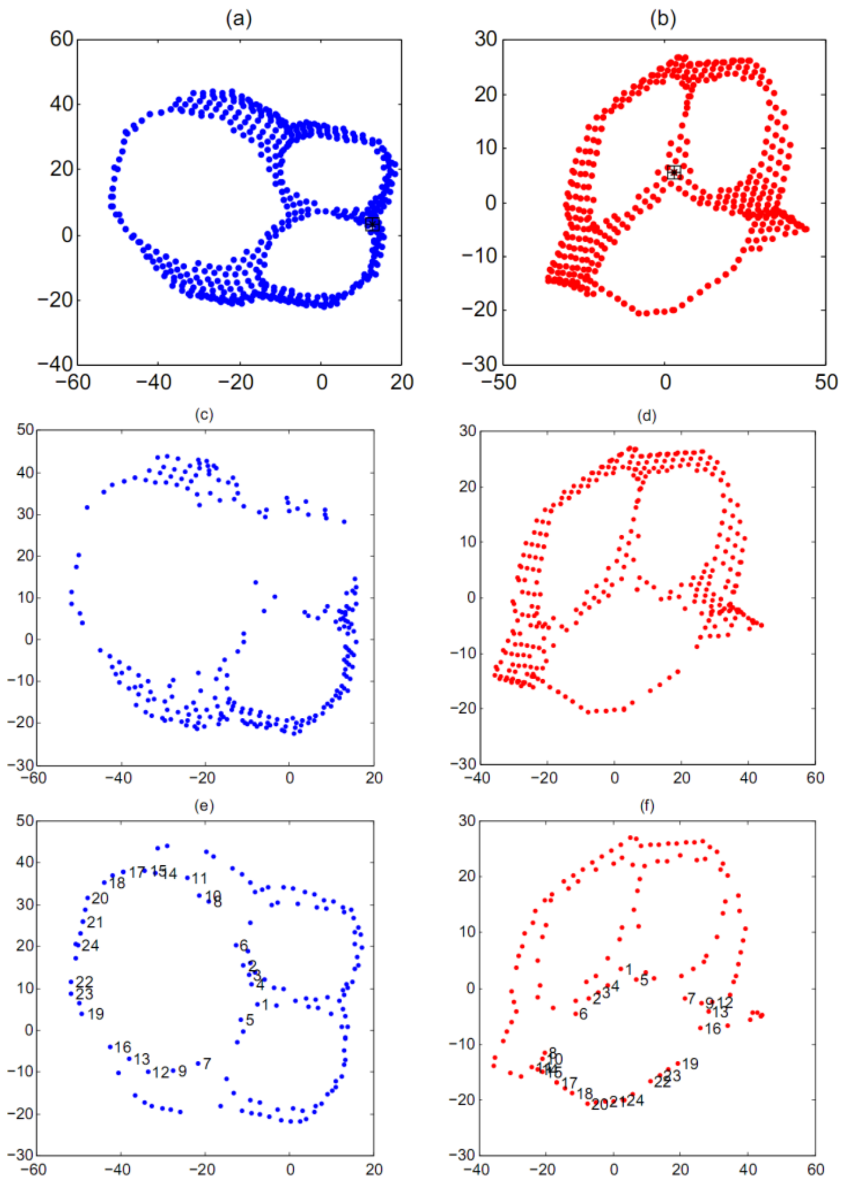


Figure 10.14. (a-b) Two view of TPMs at two randomly selected nodes after 20000 packets disseminate in the network (c-d) Common set of nodes that two randomly selected nodes have (e-f) Set of common boundary nodes at two randomly selected nodes. Nodes on the inner boundary have the same order, which indicates the maps are consistent.

map development at a randomly selected node are illustrated in Figure 10.11, as the number of packet transmissions accumulate over time. The percentage of nodes that have been mapped by the observing node is as shown in Figure 10.13. The third example is a network deployed along the foundation or walls of a building, consisting of 343 nodes. Three anchors are manually placed at three corners (see Figure 10.5 (a)). Though these three anchors provide a good topology map for the building as in [37], random placement of anchors also gives the correct topology map. A map at a random node evolves as illustrated in Figure 10.12. All the three examples clearly illustrate how a network of sensors can be deployed and allowed to develop topology awareness without requiring much overhead.

Initially no node had any idea of its position, the neighbors' positions, or the network topology. The only information it knew about the network was how many neighbors it had and the neighbor IDs. Nevertheless, with the proposed scheme, nodes now are aware of their position in the overall network, the shape of the network and its topology. All this has been achieved without a dedicated setup phase requiring its own packet exchange, but by listening to ongoing transmissions and inferring information about the nodes and the network.

C. Topology map consistency between two nodes

As explained, TPMs at two nodes are different, and the nodes available on the map are not the same. The question then is whether the different nodes have a consistent picture of network. We examine this next. Figure 10.14 (a) and (b) show the TPMs at the two randomly selected nodes after 20000 packets have propagated in the network. Not only the orientation, but also the number of nodes available at each node is different since each node relays/hears a different set of packets. Figures 10.14 (c) and (d) capture the set of nodes that are common to both the maps. Note that Figures 14 (a), (c), and (e) illustrate the map at one node, and Figures 14 (b), (d), and (e) are the corresponding views at the second node. As described in Section 10.3.C, topology maps at different nodes can be different but still be useful. Fish-eye views of an area from two different vantage points could be vastly different, yet would be useful. Our approach also produces topology maps that provide somewhat distorted views of the physical map depending on the vantage points (node); however, they are useful for tasks such as routing as it is based on connectivity. Figures 10.14 (e) and (f) illustrate the subset of nodes that are not only common to each of the maps (as in Figures 14 (c) and (d)), but are also on the boundaries of the network as determined by each node. Then, in order to illustrate the consistency of the map, we have selected one inner boundary and used the node IDs to identify them on the two maps. As in Figure 10.14 (e), as the inner boundary is traversed counter-clockwise, the order of the common boundary nodes is $\{1,4,3,2,6,8,10,11,14,15,17,18,20,21,24,22,23,19,16,13,12,9,7,5,1\}$. As can be seen in Figure 10.14 (f), the order of the inner boundary at the other node is the same. This illustrates and justifies our claim that maps at two random nodes are consistent. Since nodes take the decisions locally, for example for routing, it is sufficient to have topologically consistent maps.

D. *Energy consumption and compatibility with power harvesting schemes*

In this section, we address several scenarios in which a self-learning strategy for network awareness is more desirable compared to achieving network awareness with a dedicated setup phase. A dedicated setup consists of two phases: a) a VC-generation phase involving multiple flooding, with communication complexity of the order of $O(MN)$; and b) a phase in which VCs of each of the nodes is distributed to other nodes to calculate the VC to TC transform and generate the map, with communication complexity of the order of $O(N^2)$. Instead of all these additional messages, the proposed scheme piggybacks small amount information, anchor IDs and hops, on each routine packet. Consider the additional energy consumption due to this increase of packet length. The probability of a node becoming an anchor is M/N. As a packet with TTL = H visits H nodes, the expected number of anchors that a packet visits can be approximated by $(M/N)H$. Assuming one byte per anchor ID and one for its ordinate, the expected number of bytes padded to a packet is $2(M/N)H$. Let the original packet length be B bytes. Therefore, percentage increase in the amount of energy consumption during the learning stage due to padding is given by $2\left(\frac{M}{N}\right)\left(\frac{H}{B}\right)100\%$. For the network with 496 nodes shown in Figure 10.5 (a), with 2% of nodes as anchors and a TTL of 100, the percentage incremental cost due to self-learning per packet is $\frac{4}{\mathrm{B}}\%$, and even this small increase occurs during only the self-learning stage.

Although one can conceive a dedicated setup phase for achieving network awareness, there are many situations where such an implementation is not desirable or even feasible. An example is the emerging class of power-limited networks, as opposed to traditional battery-operated networks in which the total energy was considered the limiting factor. WSNs will be deployed in regions that are difficult to access and so the sensor nodes must be autonomous and independent in terms of energy. Recent advances in power-harnessing schemes [70][112] remove a technical barrier for long-term deployment of sensor networks. Under energy-harvesting strategies, for a perpetual sensor node operation, it must be such that $P_g \geq P_c = P_{idle} + d\,.P_{active}$, where P_g and P_c are the generated and consumed average powers respectively. d, P_{idle} and P_{active} are duty cycle, power consumptions when the node is idle and active. When d is large, i.e., the node is active for a long period, P_c would be high. Hence P_g may not be sufficient to power the sensor node's operation. Even with certain battery-operated devices, one may need to limit the peak power consumption. As stated in [112], a device with a coin-size battery restricted to consume 200 μW of power on average, lifetime can be extended to a maximal 167 days, which is equivalent to half a year. In such cases the power is limited, but nodes are able to pursue long-term strategies to enhance its capabilities.

A dedicated learning phase to achieve network awareness under power-limited conditions can require a long initial period during which the network is not available for the sensing task. The time it takes to harvest energy to transmit a packet is E_c/P_g, where E_c is the energy per packet and P_g is the power generated by harvesting scheme. The duration that a network is not available for sensing would grow at least linearly with the number of nodes, and it is inversely proportional to the harvested power. With an evolving approach, the network carries out the sensing tasks with its current level of abilities, but becomes more effective with long-term learning.

Even with a dedicated setup phase, there are circumstances in which a network may not benefit fully from the available structural information. Completion of a setup phase allows a message going from source to a destination to find a shorter path compared to random routing. Yet, the destination (node needing information) or event (node that has information) still has to be discovered using a scheme such as random routing. Only a few exceptions for this exist in which the destination is known apriori, e.g., a fixed-base station. In cases where each event involves only a very few message exchanges, it takes a significant amount of time to compensate for energy used in a structuring phase. However, if the event triggers multiple message transfers after the first discovery message, those subsequent transfers can follow shorter paths, significantly benefiting from the structuring information.

10.5 Topology Awareness – Applications

During TC generation, a node initially gains knowledge of neighbors' relative positions. Note that now the decision-making capability is higher than during the initial state, when nodes were completely oblivious. After that, the node expands its awareness beyond the neighborhood, transforming from neighborhood-aware state to become network-aware. Now the node can become aware of the boundary nodes and physical holes in the network that prevent those nodes from sensing or routing through certain areas. By conditioning the map with sensed data, even dynamic events and their boundaries can be identified. The overall effort of the scheme is to dramatically improve the distributed decision-making capability of individual nodes, and by the network as a whole. Below we consider two examples that can directly benefit from the proposed scheme.

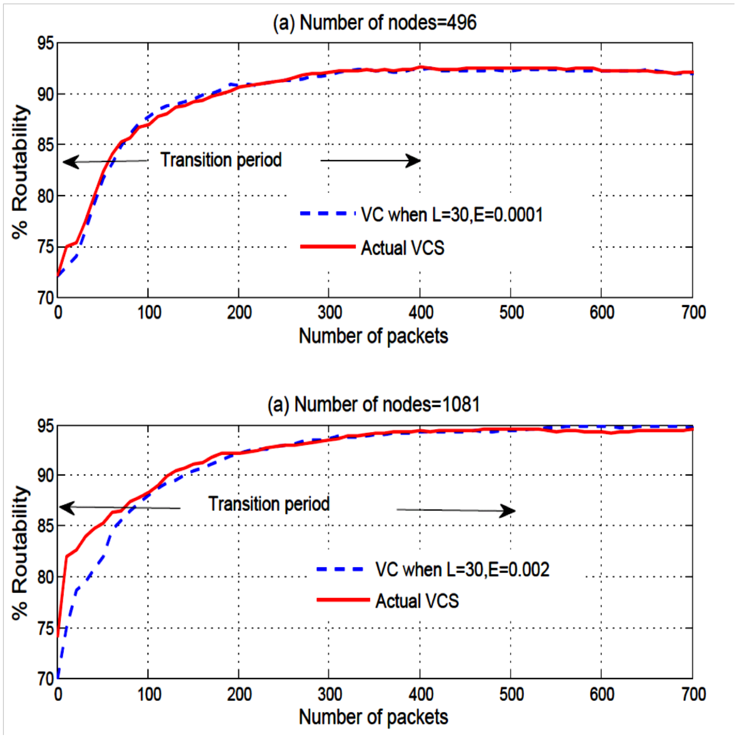


Figure 10.15. Variation of percentage number of nodes in the topological domain as the number of packets generated in the network varies.

A. Routing in a self-learning environment

Efficient routing is the key to efficient data dissemination, and therefore for any sensor network application. Routing algorithms ranging from random routing to coordinate-based routing can significantly benefit from topology awareness. Random routing schemes depend on the rendezvous of queries seeking data and agents carrying information about events. Queries and agents follow random paths in the network, with the expectation of visiting a node visited by its counterparts. With the presence of a topology map within a node, it is possible to direct such queries to dense regions (market places), junctions, or certain geometric features, thereby increasing their rendezvous probability, and thus significantly enhancing the efficiency. Alternatively, with the availability of TPMs, they may be sent in straight lines or some geometric pattern to maximize the rendezvous probability. Straight line rumor routing and market place random routing are examples that can significantly benefit from the proposed approach, without having to resort to physical node localization. Even partial topology maps, i.e., when a node does not have information about the entire network, can still be beneficial. With the development of a topology map within a node, the node now has information about the network boundary and obstacles between it and any other node, and therefore is capable of mapping the best path from it to any other node on the map very accurately.

Above we discussed how the different routing schemes could benefit from network awareness. Next, we illustrate how a node can upgrade the routing algorithm, i.e., switch to better and more efficient ones, as it goes through different learning stages. From random routing, network develops VCS. Now, it can use VCS-based routing schemes. Moving further along, the nodes can generate TCs. Having both TCs and VCs provide the ability to achieve very high routability by using one domain to overcome the voids in the other domain. Geo-Logical Routing (GLR) scheme [39] switches among three modes: a) TC, which uses topology-based coordinates where TC is used for distance evaluation; b) VC, which uses VCs and distance, is based on VCS; and c) AM, which routes toward selected anchor A_c, which is closest to the destination. The source node initiates routing in TC mode. The packet continues to be routed in this mode until it reaches a local minima in the topology space. Then, the mode is changed to VC mode. If a packet comes across a node, which is a local minima in VC mode, the packet is routed using the AM mode in which the packet is sent to the anchor closest to the destination. Further detail can be found in [39].

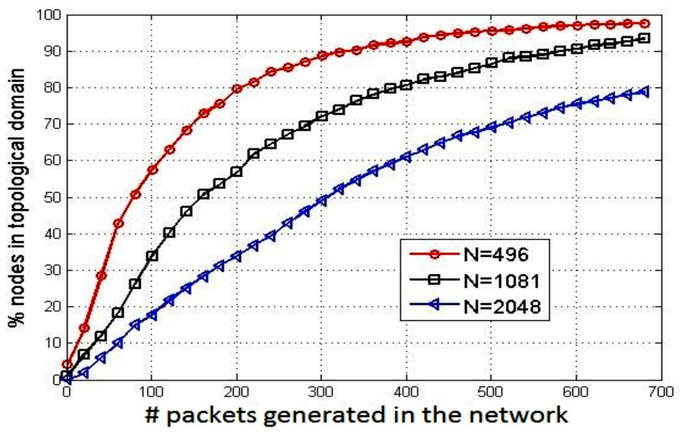


Figure 10.16. Variation of percentage average routability of the learning network as the number of packets in the network varies.

Figure 10.15 illustrates the routability performance improvement that can be achieved by changing the routing algorithm as nodes reach different stages of learning. For this evaluation, circular network with 496 nodes (for Figure 10.15 (a)) and 1081 nodes (for Figure 10.15 (b)) are used. Routability of the network is defined as,

$$\% \text{ Routability} = \frac{\text{Total \# of packet that reached the destination}}{\text{Total number of packet generated}} \quad (4)$$

In the proposed self-aware network routing, which is initially based on VCS will gradually transform to topology map based routing, GLR achieving significantly high performance compared to VCS-based routing. Since the network is self-evolving, a node may even switch to VCS algorithm prior to the convergence of VCS. We already examined error due termination criteria of VCS learning using Figure 10.9. To consider the impact of such an error, we compare the cases where a node with a certain error E_V in VCS starts using VCs for routing and one without error in Figure 10.15. When the number of packets is zero, the network uses only VCS-based routing. While any VC-based routing proposed in literature may be used, the simulation is based on GLR, excluding the TC mode. In the simulation, nodes generate TCs when it collects $L_N = 10$ VCs or source/destinations. As it can be seen from Figure 10.15, during this transient period, a fraction of the network nodes use VC-based routing while rest uses GLR. It can be clearly seen that routing performance is more or less the same when E_V is zero and E_V not zero, which indicates that learning network may tolerate small percentage of error in VCS. Moreover, overall routability improves by 25% for both the network sizes due to upgrading the routing scheme from VCS-based routing to TPM-based routing.

Finally, Figure 10.16 shows the percentage of nodes in the topological mode as the number of packets in the network vary, thus those nodes use GLR-based neighbor selection. The rest of the nodes is in VC mode and uses VC-based greedy forwarding.

B. Limitations and extensions of self-learning scheme

This section discusses briefly the flexibility and possible extensions of the learning algorithm under node failures and node mobility.

The algorithm for network-awareness relies on a set of nodes becoming anchors, and each such anchor inserting a count in each header of packets it forwards. Although it is used only during the VC development, it does consume network resources and requires modification of packets. However, this overhead can be eliminated as explained next, if one is willing to spend a longer time for achieving network-awareness. This is done by using the TTL field as a counter for the purpose of VC generation. A packet in a WSN generally carries the source address as well as a TTL field. Consider initializing TTL to zero at the source node, so the packet can be dropped when a certain threshold is reached. This counter is an estimate of the distance to the source node of that packet and, as in the case discussed above, will converge over time to the actual shortest distance. However, it is also important to note that *M* need not be the same at each node for the purpose of topology map generation, thus providing a significant degree of flexibility and efficiency, as well as the opportunity to speed up the learning process. This scheme does not require modification of fields of the packet header. It allows for a completely passive listening-based learning process. Other strategies can also be developed that trade off overhead and learning time in different ways.

Node failures, including those of anchors, have an effect on the VCs when the shortest path between two nodes is altered due to the node failure. Though TPM is generated based on the VCS, it is a representation of the physical map of the network. Therefore the topology map of remaining nodes is not affected by the failure of a node, even that of an anchor. In fact, this is a major advantage of TCs over VCs, even though the former is derived from the latter. Thus, node failures that occur after the network has gone through the first stage of the learning process will have very little impact on the network awareness. The presence of dead nodes on the TPM can lead to wrong decisions; however, the means for handling such cases exist, for example, by associating a time-out for node entries on the TPM.

Another interesting challenge is learning in a network where a fraction of nodes is mobile. If the network contains a significant fraction of nodes that are stationary, network awareness can be achieved only with respect to those nodes. An ongoing study [63] uses the centroid of the static neighborhood of a mobile node as its VC and TC without degrading the overall routing performance. In cases where the time intervals related to significant physical movements are larger compared to the time it takes the network to generate topology maps, the scheme described can be used repeatedly to achieve network awareness in mobile networks. In fact, network awareness would be very useful for generating desired swarm behaviors on 2D and 3D mobile robot and nano-robot networks. Realizing this requires further investigation.

Finally, we address the question of applying the proposed learning scheme to massive networks in the scale of trillions of nodes. Today's localization and other network structuring algorithms are not pragmatic for such networks. As the learning time increases rapidly with the number of nodes, the key to achieving topology awareness in such networks lies in hierarchical organization. Such networks can be partitioned into regions or clusters in such a way that nodes in each region are mapped to a single VC. The network awareness can be generated at cluster level for the entire network, i.e., each cluster will know the positions of different clusters and their relationship to one another. Combining such inter-cluster awareness with intra-cluster node awareness, better scalability can be achieved. Thus, nodes can get a sketch of the network, which will provide sufficient information of the network voids, boundaries, etc. This is also under investigation.

10.6 Conclusions

This research presented a novel scheme that allows nodes in a distributed sensor network to acquire network-awareness by listening and reasoning based on network activities over time. As network-awareness develops, the nodes individually and the network collectively are able to switch to more sophisticated algorithms for functions such as routing, sensing and fusion. Network-awareness at nodes can revolutionize the implementation of many application level algorithms as well.

The effectiveness of the scheme was demonstrated using 2D and 3D sensor networks. Routing was used as an example to illustrate how the overall performance can benefit from node awareness. As nodes gain the knowledge of network topology using the VCS, performance of routing improves by 25%. In fact, by storing a complete topology map, for example as a bit map to minimize memory consumption, it is possible to identify the source to destination paths completely and accurately, and thus achieving 100% routability.

This work is a step towards future sensor networks that evolve over time, becoming smarter by learning and inferring information about the network, based on information gleaned from ongoing packet transmissions.

CHAPTER 11

RECOVERY BOUNDS AND PHENOMENA AWARENESS IN WSNS – A COMPRESSIVE SENSING BASED APPROACH

11.1 Introduction

The novel theory of Compressive Sensing (CS) is an emerging mathematical approach with a significant potential to recover functions using a few samples/measurements. It attempts to reconstruct certain signals/images from highly incomplete samples. CS is based on the empirical observation that most of the signals can be represented by a sparse expansion under a suitable basis or a frame.

Compressive Sensing [28][26] is posed as recovering an n-dimensional signal vector x ($\in \mathbb{R}^n$), that is k-sparse in its sparse representation with m ($\ll n$) number of samples y ($\in \mathbb{R}^m$) which are linear combinations of the signal vector x given by $y = A.x$. With given y and A, this under-determined system is solved for x as

$$\min \|x\|_1, \quad s.t. \quad A.x = y \tag{1}$$

where $\|.\|_1$ is the L^1 norm. A ($\in \mathbb{R}^{m\times n}$) is called the sensing/measurement matrix/frame. In applications where the sparse representation of the function is defined by a basis, such as Fourier or Cosine, A is defined by the selected basis. Most of the theoretical bounds in CS are derived based on the Restricted Isometric Property (RIP) of the sensing matrix A. RIP requires every combination of support of x many columns of A to be well conditioned and written as

$$(1-\delta)||x||_2^p \leq ||A.x||_2^p \leq (1+\delta)||x||_2^p \tag{2}$$

where δ is called Restricted Isometric Constant (RIC) and is specific for the support of $x.$ If y is a subset of samples of x and x has a sparse representation in a known domain ψ, the problem can be re-written as

$$y = R\,.\,x = R\,.\,(\psi\, X) = (R\,\psi)\,.\,X \tag{3}$$

Consequently, the measurement matrix A = R ψ. R is a subset of rows of identity matrix selected under a probability mass function (pmf). As $\psi_{n\times n}$ is an orthonormal basis, RIP of A is achieved via R.

The goal is to recover the sparse representation of the function/signal in a transformed domain. Thus, two main design criteria emerge: the choice of the basis/frame and the row selection scheme. The basis/frame is chosen to have a sparsest possible representation. The row selection scheme essentially is the sampling scheme. Reference [100] derives the probability of failure and the minimal number of samples required, when the measurement matrix is constructed by drawing rows from an orthonormal basis according to a so-called orthogonalization measure defined next.

Let $\mathcal{D} \subset \mathbb{R}^d$ be endowed with a probability measure υ. Moreover, ψ is an orthonormal system of complex-valued functions on $\mathcal{D}$.

For $j, k \in [n]$, if

$$\int_{\mathcal{D}} \psi_j(t)\overline{\psi_k}(t)d\upsilon(t) = \begin{cases} 0 \; if \; j \neq k \\ 1 \; if \; j = k \end{cases} \tag{4}$$

then υ is said to be an orthogonalization measure. The fundamental idea behind orthogonalization measure is that it selects rows from a basis in such a way that the resultant sub-matrix has orthogonal columns satisfying (2), resulting in a measurement matrix with high recovery performance. The widespread favor for uniform distribution to draw rows is due to its orthogonalization property. In this research, we generalize the result in [100] for any row selection scheme from any basis not requiring to satisfy orthogonalization property in (4).

In certain cases, a sparse representation in a common basis as Fourier or Cosine may not be available, and it may be hard to form an orthonormal basis producing a satisfactory sparsity level. As pointed out in [11] frames provide a more flexible and convenient alternative to bases. It is shown that finite frames play a central role in the design of both sparse representations and compressed sensing methods. Moreover, it has been proven that frames with small spectral norm and/or small worst-case coherence, average coherence, or sum coherence are well-suited for making measurements of sparse signals.

In general, applications of CS span network topography, face recognition, image recovery, and sparse signal recovery. Proposed bounds are applicable in the area of sparse signal recovery. For instance, images and musical notes are sparse in Wavelet and Cosine basis, respectively. A few example scenarios where CS can be used in Wireless Sensor Network (WSN) applications are discussed next. The hydrologic study by USDA-ARS Great Plains Systems Research (Fort Collins, Colorado) has 110ha of a winter wheat and fallow strip cropping system [85], where soil measurements are collected using sensors mounted on a pickup truck. Traditional CS would demand uniformly at-random samples over the field, which is un-realizable to achieve with a truck. Moreover, Intel's Wireless Vineyard [10] uses ubiquitous computing for agricultural monitoring, data collection relies on data mules, which are small devices carried by people/dogs/robots that communicate with the nodes and collect data. Again, samples are not collected in a uniformly at-random manner. Thus, the proposed theoretical formulation is of much use in such scenarios.

We formulate generalized mathematical limitations to use a sampling scheme of choice under a preferred basis or a frame. Mathematical bounds found so far have been limited to uniform sampling under the Fourier basis. We derive three main results under this relaxation:

1. Recovery failure probability,
2. Minimum number of measurements/samples needed, and
3. Measurement overhead on the number of measurements/samples required for use of non-orthogonalization measure.

The rest of the research is organized as follows: Section 11.2 presents the related work. Section 11.3 derives recovery bounds of CS-based phenomena discovery under relaxed constraints on sampling distribution. Application of RW-based sampling in WSNs is proposed in Section IV and evaluated in Section V. Section VI concludes this research.

11.2 Related work

This section briefly discusses the CS-based approaches for data aggregation and localization in WSNs. Conventionally, measurement matrix A gives m uniformly at-random number of linear combinations of samples. Uniform random sampling based CS recovery in DCT and wavelet domains are addressed in [96]. Nevertheless, the proposed approach lacks the required mathematical justifications for using CS in DCT and wavelet domains, which we provide in the research. Reference [87] focuses on binary sparse event discovery using the

Bayesian detection. However, their performance decays as the signal-to-noise ratio (SNR) approaches 20dB. Reference [124] investigates minimizing the network energy consumption through joint routing and compressed aggregation where uniformly at-random samples are routed to a sink through a tree-based structure. An energy efficient compressed sensing scheme for wireless sensor networks using spatially localized sparse projections is proposed in [75]. Here, measurements were obtained from clusters of adjacent sensors in order to reduce transmission cost. Single dimension function recovery in underwater sensor networks is discussed in [47], where function is assumed to be sparse in Fourier domain and sensors send their information directly to the base station in a uniformly at-random manner. Different from the above, [104] proposes spatial domain sparse function recovery at a sink using RW-based linear combination of the sensed values. The drawback of this scheme is, the larger the size of the network, the larger the number of RW-based linear combinations required.

A CS for manifold learning protocol (CSML) is proposed in [48] for localization in wireless sensor networks. Each sensor transmits a subset of distance measurements to a central node. Then the central node recovers the full pair wise distance matrix through an L^1-minimization algorithm. A CS-based approach for sparse target counting and positioning scheme is proposed in [130]. The proposed greedy matching pursuit algorithm (GMP) in [130] complements the well-known signal recovery algorithms in CS theory and proves that GMP can accurately recover a sparse signal with a high probability.

All the applications discussed above share two comment factors: uniformly at-random sampling and recovery at a base/central station. The focus of this research is smooth function discovery using a transformation. Thus, in uniformly sampled sensor values need to be sent to a BS. Why don't we make use of the sensed information lie on the path, which uniformly sampled sensed information travel to BS? Additionally, sensor nodes must be placed/selected uniformly in order to collect samples of the phenomena uniformly at-random, which is not practical in most of the real applications. We propose RW-based sampling, and also make it feasible to discover the phenomena centralized as well as distributed. Moreover, we rigorously justify the validity of the problem formulation.

Table 11.1
Notation and Description Used in Chapter 11

Notation	Description
N_T	Total number of samples of a function
P	2D phenomena
x	Vectorized phenomena
X	Transformed x
y	Subset of samples of x
m ($\overline{m}$)	Number of samples required in uniform (non-uniform) row selection
A	Measurement matrix
s	Sparsity of x
S	Support of x , $\lvert S\rvert = s$
A_S	Sub-matrix constructed by selecting columns from A on S
δ	Restricted Isometric Constant (RIC) for sparsity s
R	Row selection matrix
Ψ	Transformed domain basis
Ɗ	Space of ψ
K	Bound on the largest element of basis ψ
$\phi_{i,j}$	Inner product between the ith and jth column of ψ
$\rho_{i,j}$	$\phi_{i,j}$ when $i \neq j$
Ɲ	Off-diagonal elements of inner product of ψ over generalized sampling
E_p	Deviation from normalized inner product of A_S from $\boldsymbol{I}$
p	Power used in RIP condition
Q	Change in E_p due to generalization
ξ	Upper bound on Q

11.3 Recovery Bounds for Sparse Function Recovery with Non-orthogonalization Measure

Most of the transformed domain sparse function recovery schemes assume that the spatial domain function is sampled uniformly at-random, which corresponds to selecting rows of a basis under the uniform distribution. Many also assume a sparse representation in the Fourier basis, as uniformly selected Fourier basis rows satisfy the required conditions of the existing theorems [100]. However, this may not be the case with real-world applications. Gathering samples uniformly at random may not be pragmatic. In addition, Fourier basis may not produce a satisfactorily sparse representation, requiring consideration of other bases or frames. Thus, we investigate the case where the rows of a basis of interest are drawn from an arbitrary distribution, relaxing the requirements imposed on basis and sampling distribution in existing theorems. This section formulates the following recovery bounds associated with CS-based recovery under relaxed constraints:

1. Recovery failure probability,
2. Minimum number of measurements/samples needed, and
3. Measurement overhead on the number of measurements/samples required for use of non-orthogonalization measure.

Notations used in the derivations and theorems are tabulated in Table 11.1.

A. Recovery failure probability

Let x with support S be the signal to be recovered. The support of the signal vector is the set of non-zero elements in x. s, the number of non-zero elements, in other words sparsity of x, is the cardinality of S. Failure of recovery of a signal with support S is viewed as A being unable to satisfy RIP. In this case, RIP implies that any sub-matrix formed by s number of columns of A corresponding to S, referred as A_S being nearly orthonormal, i.e.,

$$||\widetilde{A_S}^*\widetilde{A_S} - \boldsymbol{I}||_2 \leq \delta \tag{5}$$

here $\widetilde{A_S}$ denotes column wise normalized A_S and δ is the RIC. The probability with which the above is unable to be satisfied is defined as the probability of failure, ϵ.

$$\epsilon = \mathrm{P}[||\widetilde{A_S}^*\widetilde{A_S} - \boldsymbol{I}||_2 \geq \delta] \tag{6}$$

Theorem 1: A signal with sparsity s in a transformed domain under a basis ψ with at most $\bar{m}$ samples will perfectly recover with probability of (1- ϵ) for:

$$\epsilon \leq 2^{3/4} s \exp\left(-\frac{\delta^2 \bar{m}}{8\,K^2 \bar{\kappa}^2 s}\right) \tag{7}$$

where ϵ is the failure probability, δ is the RIC, elements of ψ is bounded by K and $\bar{\kappa}$ is a constant determined by the sampling scheme.

Proof: A general normalized system ψ on $\int_{\mathrm{D}}$ can be written as

$$\int_{\mathrm{D}} \psi_j(t)\overline{\psi_k}(t)dv(t) = \phi_{j,k} = \begin{cases} \rho_{jk} \; if \; j \neq k \\ 1 \; if \; j = k \end{cases} \tag{8}$$

where υ is a probability measure and $j, k \in S$. If υ is an orthogonalization measure, $\rho_{jk} = 0 \;\forall j, k$ (See (4)). Let $X_l = (\overline{\psi_j}(t_l))_{j \in S}$, then, expected value of $X_l X_l^*$

$$\mathbb{E}(X_l X_l^*)_{j,k} = \phi_{j,k} \tag{9}$$

$$\mathbb{E} X_l X_l^* = \boldsymbol{I} + \mathrm{N} \tag{10}$$

where t_l is a set of arbitrary indices and $\boldsymbol{I}$ is the identity. N is the off-diagonal elements of $\mathbb{E} X_l X_l^*$, and it is a critical term when the sampling scheme and the basis do not follow the condition for orthogonalization measure given by (4). Recovery performance is degraded due to nonzero N, thus, requiring additional number of samples to achieve similar recovery properties such as probability of failure, error in recovered function, etc.

The general form of the RIP condition is given in (2) can be re-arranged to:

$$\delta = \max_{S \subset [n], |S| \leq s} ||\widetilde{A_S}^* \widetilde{A_S} - \boldsymbol{I}||_2 \tag{11}$$

From Markov Inequality the probability of failure is bounded above by;

$$\mathrm{P}[||\widetilde{A_S}^* \widetilde{A_S} - \boldsymbol{I}||_2 \geq \delta] \leq \frac{E_p}{\delta} \tag{12}$$

where E_p is:

$$E_p = \mathbb{E}\left\|\widetilde{A_S}^* \widetilde{A_S} - \boldsymbol{I}\right\|_2^p \tag{13}$$

Substituting for $\boldsymbol{I}$ from (9)

$$E_p = \mathbb{E}\left\|\frac{1}{\bar{m}}\sum_{l=1}^{\bar{m}} X_l^* X_l - \mathbb{E}X_l X_l^* + \mathrm{N}\right\|_2^p \leq \mathbb{E}\left\|\frac{1}{\bar{m}}\sum_{l=1}^{\bar{m}} X_l^* X_l - \mathbb{E}X_l X_l^*\right\|_2^p + \mathbb{E}\|\mathrm{N}\|_2^p \tag{14}$$

where $\bar{m}$ is the number of measurements. Using Lemmas 6.7 and 6.18 in [100],

$$E_p \leq D^2\left(\sqrt{E_p} + 1\right) + Q \tag{15}$$

where D :

$$D^p = \left(\frac{2}{\sqrt{\bar{m}}}\right)^p 2^{3/4} s p^{p/2} \exp\left(-p/2\right) \tag{16}$$

and Q ,

$$Q = \mathbb{E}\|\mathrm{N}\|_2^p|_{p=2} \leq \xi \tag{17}$$

Note that Q is the noise generated due to violation of the condition for orthogonalization measure given by (4). Q vanishes, as in the case of [100], where the sampling is an orthogonalization measure. By solving for E_p

$$E_p{}^{1/p} \leq D\left[\sqrt{1+\frac{D^2}{4}+\frac{Q}{D^2}}+\frac{D}{2}\right] \tag{18}$$

Let,

$$\bar{\kappa} = \sqrt{1+\frac{D^2}{4}+\frac{Q}{D^2}}+\frac{D}{2} \tag{19}$$

Substituting E_p (18) in (12) and rearranging terms as in Proposition 6.5 in [100], gives us our first results: the failure probability ϵ.

$$\mathrm{P}[\left\|\widetilde{A_S}^*\widetilde{A_S} - \boldsymbol{I}\right\|_2 \geq \delta] \leq\ 2^{\frac{3}{4}}\, s\, \exp\,(-\frac{\delta^2\bar{m}}{8K^2\bar{\kappa}^2 s}) \tag{20}$$

where K is bound on the L^2 -norm of rows of ψ , and $\bar{m}$ is the number of measurements needed under relaxed constraints. QED

B. Minimum number of samples required

The minimum number of samples required in order to achieve a predefined success probability is derived next.

Theorem 2: The minimum number of measurements needed $\bar{m}$ to recover an s-sparse signal in a transformed domain, where the basis is , with probability (1- ϵ) is

$$\bar{m} \geq \frac{K^2\bar{\kappa}^2 s}{\delta^2}.\,ln\left(\frac{2^{3/4}s}{\epsilon}\right) \tag{21}$$

where δ is the RIC, $\|\psi\|_2$ is bounded by K and $\bar{\kappa}$ is a constant determined by the sampling scheme.

Proof: From (20):

$$\epsilon \leq\ 2^{\frac{3}{4}}\, s\, exp\,(-\frac{\delta^2\bar{m}}{8K^2\bar{\kappa}^2 s}) \tag{22}$$

Rearrange (22) to obtain (21). QED

A class of functions can be defined with the same sparsity. Hence (21) provides a lower bound for the number of samples required to reconstruct the original function with a success probability of (1- $\boldsymbol{\epsilon}$) .

C. *Measurement overhead due to use of a non-orthogonalization measure*

When rows from an orthonormal basis are selected not necessarily at an orthogonalization measure, more samples may be required to achieve similar performance, than if otherwise. These extra measurements are termed "penalty" paid due generalization.

Theorem 3: A signal s-sparse in a basis ψ that can be perfectly recovered with m samples under an orthogonalization measure with probability (1- ϵ); under a non-orthogonalization measure will require $\bar{m}$, given by:

$$\bar{m} = m\left(\frac{\bar{\kappa}}{\kappa}\right)^2 \tag{23}$$

where κ is a constant corresponding to an orthogonalization measure, $\bar{\kappa}$ corresponds to a non-orthogonalization measure and

$$\frac{\bar{\kappa}}{\kappa} = \left(\frac{\sqrt{17+16\xi}+1}{\sqrt{17}+1}\right) \tag{24}$$

and $\mathbb{E}\|\mathrm{N}\|_2^p|_{p=2} \leq \xi$.

Proof: Let m be the number of samples required sampling under an orthogonalization measure and $\kappa = \bar{\kappa}|_{Q=0}$ from (19). To find the measurement overhead (the fraction of extra measurements) in relaxing the orthogonalization constraint, we observe the following relationship when both the methods achieve identical ϵ:

$$\frac{m}{\kappa^2} = -ln\left(\frac{\epsilon}{2^{3/4}s}\right).\frac{8K^2 s}{\delta} = \frac{\bar{m}}{\bar{\kappa}^2} \tag{25}$$

$$\therefore \bar{m} = m.\left(\frac{\bar{\kappa}}{\kappa}\right)^2 \tag{26}$$

The term $\left(\frac{\bar{\kappa}}{\kappa}\right)^2$ is referred to as the multiplicative measurement overhead of generalization. From [100] $\kappa = \frac{\sqrt{17}+1}{4}$ and since Q is bounded by ξ,

$$\bar{m} \geq m\left(\frac{\sqrt{17+16\xi}+1}{\sqrt{17}+1}\right)^2 \tag{27}$$

which completes the proof. QED

11.4 Compressive Sensing Under Random Walk-Based Sampling in Discrete Cosine Domain

The main contribution of this research is the derivation of the bounds for the relaxed restrictions to be satisfied by the sampling scheme and the transformed domain basis to construct a measurement matrix. In Section 10.3, we derived failure probability, minimum number of measurements needed, additional number of measurements required due to relaxing constraints on sampling scheme and basis. We now evaluate the above-derived bounds under random walk sampling of Discrete Cosine Basis.

A. Why Discrete Cosine Basis?

Theorem 1, operating on a basis where the signal is sparsest, provides highest recovery probability for a given number of measurements. Therefore, when it comes to a WSN deployment for sensing real-world physical phenomena, the Discrete Cosine Transform (DCT) of the phenomenon is rather promising. As [7] points out, the DCT achieves nearly optimal energy compression, comparable to Karhunen-Loeve transform, yielding the fewest coefficients, i.e., the sparsest representation, of natural signals.

B. Why Random Walk as the sampling scheme?

Random Walk is one of the simplest motion models. The concept behind random routing, which is the basis for a large number of routing algorithms for WSNs [8], is similar that of RW or Brownian motion: each node randomly selects a neighbor and forwards the received message. Rumor routing [18] is an example of RW routing protocol, in which messages such as agents and queries, also called rumors randomly traverse the network. Even when the network is structured and deterministic routing is possible, random routing schemes play a crucial role in WSNs in discovery of resources and disseminating information, especially in the absence of a base station that acts as a global repository for such information. Moreover, RW motion models are applicable for the case where samples are collected by a carrier.

However, using RW routing to gather a set of uniformly scattered measurements from a sensor field is rather inefficient. Making the messages traverse in a RW manner, while collecting measurements along the path it traverses is a more practical approach. However, such will not result in a uniform selection of measurements.

C. Implementation: Random Walk-based phenomena discovery

This section discusses the details of the RW-based phenomena discovery algorithm for the centralized and the distributed realizations.

Centralized Realization of Phenomena Awareness:

In centralized implementation, the network has a base station (BS) with a higher computational capacity. There are many scenarios where a centralized implementation is feasible or even preferable [10]. In this setup, we assume there is a carrier, a robot/vehicle/animal, collecting sensed information while traversing the network on a RW. At the end, the carrier either returns or transmits the collected data to the BS. Then BS will form and solve the CS problem to recover the phenomenon. Under similar conditions, forcing the carrier to collect samples uniformly at random is not pragmatic.

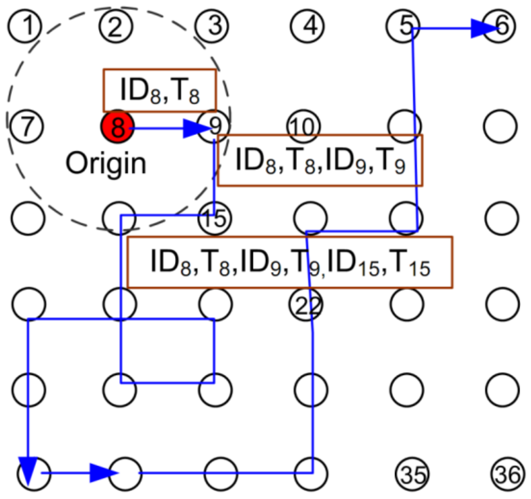


Figure 11.1: RW based sample collection on an example grid

Distributed Realization of Phenomena Awareness:

A node becoming phenomena-aware in a distributed way is crucial for many future ubiquitous sensor/actuator network applications. This phenomena awareness may be achieved using messages that continuously disseminate in the network for even/destination discovery or other management purposes. Let x be the vectorized 2D-sensed phenomena.

Then each node has a corresponding entry in x. For simplicity, we assume nodes are numbered in ascending order from the top left corner to bottom right corner (see Figure

```
INPUT: Sensed samples of the phenomena
OUTPUT: Recovered phenomena
1: $\bar{m}$ ⟵ number of collected samples
2: $y \leftarrow T_i, i = 1:\bar{m}$
3: $Z \leftarrow ID_i, i = 1:\bar{m}$
4: if $\bar{m} \geq m_T$ then
5:     for $i = 1 \rightarrow \bar{m}$ do
6:         for $j = 1 \rightarrow N_T$ do
7:             if $j == 1$ then
8:                 $\Psi(i,j) \leftarrow \sqrt{1/N_T}$
9:             else
10:                $k \leftarrow ID_i$
11:                $\Psi(i,j) \leftarrow \sqrt{2/N_T}\cos(\frac{\pi(2k+1)j}{2N_T})$
12:            end if
13:        end for
14:   end for
15:   $\min |X|_1, \text{s.t. } \Psi X = y$
16:       $x \leftarrow IDCT(X)$ % inverse DCT of X
17: else % more samples required
18:   for $j = 1 \rightarrow \bar{m}$ do % check whether i is a new sample
19:        if $ID_i == ID_j$ then
20:            flag $\leftarrow 1$
21:        end if
22:   end for
23:   flag $\leftarrow$ 0
24:   if flag == 0 then
25:        $ID_{m+1} \leftarrow ID_i$
26:        $T_{m+1} \leftarrow T_i$
27:   else
28:        flag   1
29:    end if
30:   Forward the packet to a neighbor to which packet has not been previously forwarded
31: end if
```

Figure 11.2: Distributed Phenomena Discovery – Algorithm implemented at a node

11.1), which is used as the node ID and as well the index in x. In a localized network, physical position information of nodes can be used to organize x. If the network is not localized, a hash function can be used to map an actual node ID to index range of $= [1, N_T]$.

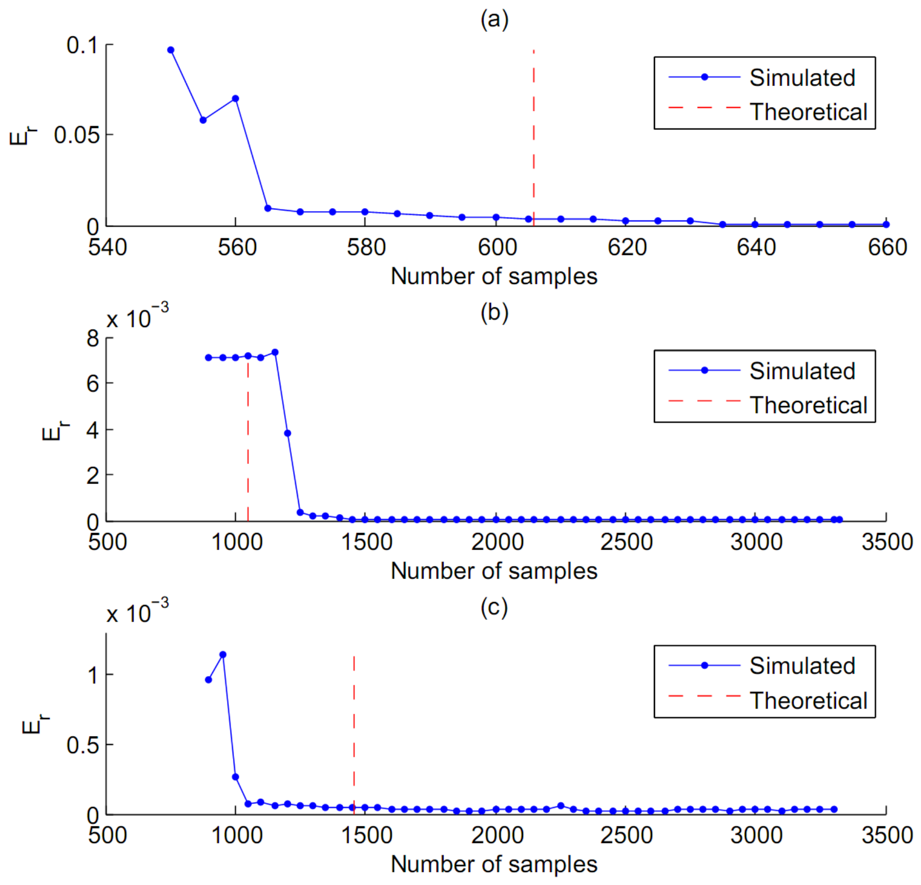


Figure 11.3: Variation of E_r and number of samples used when (a) s=2 (b) s=3 (c) s=4

Consider the example grid network in Figure 11.1, where a message generated by node 8 starts traversing the network in a RW manner while disseminating information it gathered so far at the nodes it visits. When a node receives a message, it stores the content in the message. Then the node piggybacks its node ID and the measurement to the message and forwards to a randomly selected neighbor. For instance, node 15 receives the message [ID_8, T_8, ID_9, T_9] from node 9. Node 15 then stores the message, appends its node ID ID_15 and measurement T_15, and transmits to a neighbor. When a node accumulates a sufficient number of samples, m_T, for recovery, possibly after visits by multiple packets, the node can construct the entire phenomena with a certain error. According to Theorem 2, we expect m_T to be bounded by $\bar{m}$. However, in practice m_T may be defined based on E_r required in the recovered function, which will result in a lower value compared to the theoretical value. Function recovery algorithm implemented at each node is explained in Figure 11.2.

11.5 Performance Evaluation of Phenomena Discovery from Random Walk-Based Samples

Temperature distribution of State of Alabama [25] during August averaged between 1951 and 2006 was selected to demonstrate the effectiveness of the RW-based phenomena discovery (see Figure 11.3 (a)). There are 7653 data points in total in a grid structure where we assume 7653 sensors are deployed. Each node is capable of communicating with its immediate four neighbors, i.e., communication range is one.

Table 11.2
Number of Measurements Needed For E_r of 1%

s – sparsity	Theoretical	Empirical
2	606	635
3	1048	1450
4	1458	1700

Let us consider a scenario where a rumor (message) originated at (0,0) with TTL 4000, disseminated into a network of 7653 nodes. Note that due to the possibility of revisiting to the same node, the rumor will not be able to collect 4000 unique samples. The reconstruction error E_r, of the recovered function is defined as:

$$E_r = \frac{1}{N_T}\sum_{k=1}^{N_T}(|x_k - \bar{x}_k|/x_k) \times 100 \tag{28}$$

where N_T is the total number of samples in the function, which is the same as total number of sensors in the network. x_k and $\bar{x}_k$ are the kth sample of the original function and the reconstructed function respectively. Percentage reconstruction error E_r (28) is used as the metric of performance evaluation. E_r indicates the average error in reconstruction if the sample value is 100.

Table 11.2 lists theoretical and empirical values for the number of measurements needed to obtain an $E_r$of 1%, for sparsities 2, 3, 4. Based on the samples collected by the rumor, recovery error with the samples used for reconstruction was plotted as in Figure 11.3 for three sparsity cases: 2, 3, and 4. Red dashed lines indicate the theoretically estimated m, which is a satisfactory lower bound for the samples required for recovery.

Pre-analysis of temperature distribution in Figure 11.4 (a) was performed first. If the function to be reconstructed is sparse in DCT domain, the number of non-zero coefficient should be less. The number of non-zero coefficients defines the sparsity. 3183 DCT coefficient out of 7653 in total was required to reconstruct the temperature function in Figure 11.4 (a) allowing 0.1% error, implying that even in DCT domain, the selected phenomena is not sparse as expected. Although such natural phenomena can be approximated by a sparse representation with only a few non-zero coefficients, in our simulation we aim to recover the original dataset, not an approximation. Both the centralized and distributed phenomena discoveries are discussed. MATLAB® 2011a implementation of L-1 magic [99] is used for both centralized and distributed phenomena discovery.

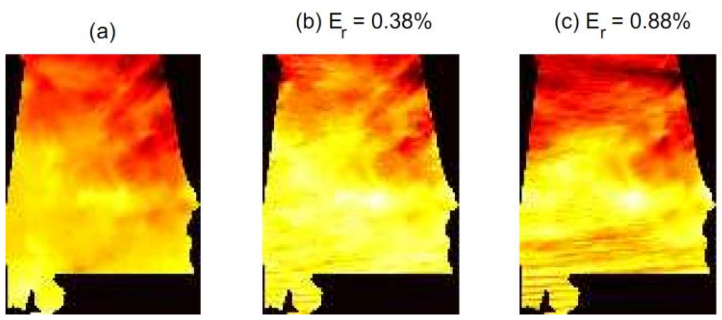


Figure 11.4:(a)Average temperature distribution of State of Alabama in August. 7653 sensors in total are available, (b) Reconstructed image based on 2583 samples collected by a single carrier and (c) Reconstructed image at randomly selected node when 1056 samples were collected by that node

A. Performance of Phenomena Discovery at the Base Station

As explained earlier, the base station is assumed to be at (0,0). A carrier (robot) collects sense information and node IDs while moving from one node to another in a RW of step size one. The maximum number of steps that the carrier will take is set to 4000 when there is a single carrier. Note that revisiting to the same node twice is allowed in the simulation, thus the carrier will collect less than 4000 samples. Figure 11.4 (b) shows an example recovery under RW sampling at a BS when 2583 samples were available. In reality, the true sparsity of the phenomenon is unknown. Therefore, the actual number of samples needed is undetermined. In Figure 11.5 we demonstrate the variation of E_r with the number of samples used. The variation of error as the number of carriers is doubled also plotted in the same figure.

Each carrier has the TTL 2000. As can be seen, performance in terms of E_r is more or less the same in the case of single or double carrier case. Even though collecting samples under uniform distribution is difficult in practice, we have used the recovery error under uniform sampling as a comparison. Error performance under RW sampling is about 0.2% less than that of under uniform sampling, when 2000 samples are available.

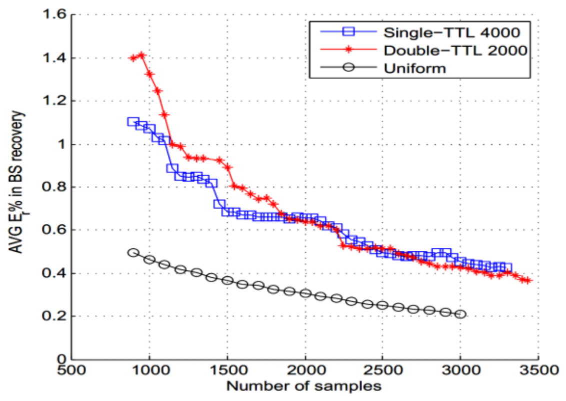


Figure 11.5: Variation of average error with number of samples used for reconstruction in centralized recovery. Single carrier with 4000 steps in total and two carriers with 2000 steps per each were used

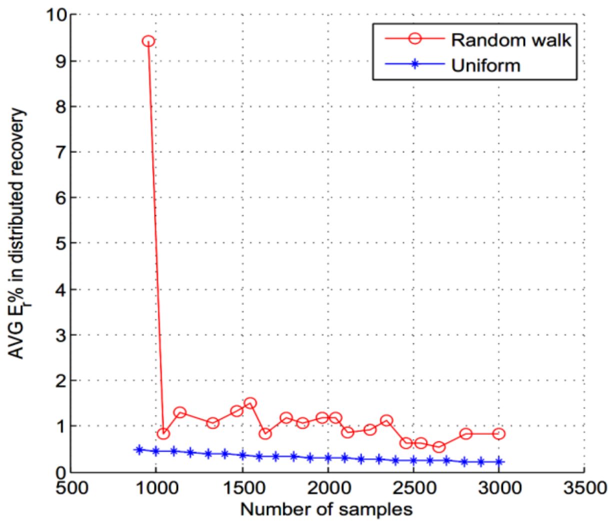


Figure 11.6: Variation of average error with number of samples used for reconstruction in a distributed manner. Evaluation is after 1000 messages with 300 TTL disseminated in the network.

B. Performance of Distributed Phenomena Discovery

Phenomena discovery in WSNs in a distributed manner is discussed for the first time. We envision future WSNs that evolve over time and become aware of the sensed phenomena. Distributed implementation of phenomena awareness may provide advantages to an array of applications. Ubiquitous networks, random routing protocols are some instances. Networking vehicles with one another and with an infrastructure that gives drivers information on the situation beyond their field of vision and warns them about accidents or traffic jams is another example.

For the distributed phenomena awareness implementation and evaluation, the same temperature data set is used. In a WSN where there is no fixed BS, random routing is used for event and sink discovery [18]. Those messages can be used to make the network learn about the phenomena being observed. Note that we used carriers in phenomena discovery at a BS while the distributed implementation uses packets disseminated in the network. In the simulation, 1000 messages with TTL 300 are generated at a randomly selected node in the network and traversed following a RW. Rumors may revisit the same node but rumor will carry only unique samples. A view of the phenomena at a randomly selected node, which has collected 1056 samples from the rumors that passed through it, is given in Figure 11.4 (c). Figure 11.6 shows the average error E_r in the recovered phenomena at different nodes. When there are multiple nodes with the same number of samples collected, the mean E_r is taken. Fluctuation in mean E_r can be observed in Figure 11.6, since different nodes have different set of samples that may result in different E_r .

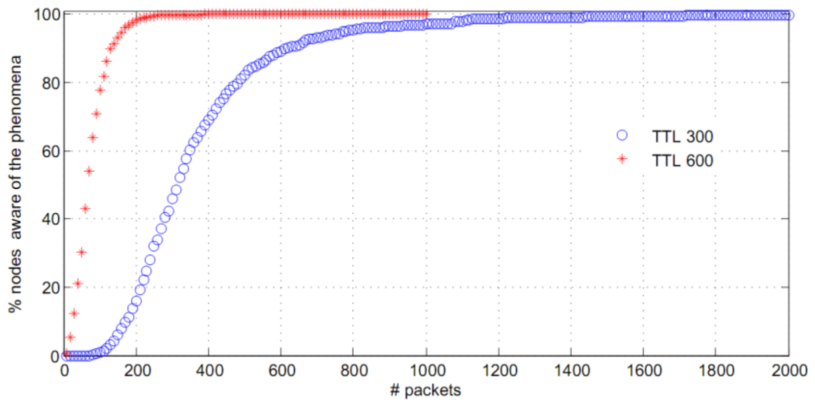


Figure 11.7: Convergence rate of nodes achieving phenomena awareness in a distributed implementation when messages has TTL 300 and 600

Next, we consider the convergence of the entire network achieving phenomena awareness in a distributed implementation. Figure 11.7 shows the rate of nodes achieving phenomena awareness under two different TTL values. From Figure 11.6 we conclude that a node needs at least 1000 samples to become aware of the sensed phenomena with an E_r of less than 2%. When the TTL is 300, at least 1200 messages need to be disseminated in the network, while when TTL is doubled then the required number of messages is less than 400. Note that the network considered has 7653 nodes. If the traditional way of uniform sampling is used for entire network to become aware of the phenomena, 1000 randomly selected nodes need to flood the network, which leads to at least 7653000 transmissions. Nevertheless, the proposed approach achieves a similar E_r map with at least 240000 transmissions, which provides at least 31.8% reduction in transmission cost.

11.6 Conclusion

A rigorous mathematical formulation of compressive sensing (CS) was presented. Recovery bounds: probability of failure, minimum number of samples needed, and measurement overhead due to sampling under a probability measure that is not necessarily an orthogonalization measure, were derived. Unlike the classical uniformly at random sampling, the model we proposed is capable of estimating recovery bounds for many pragmatic sampling schemes for phenomena recovery in a suitable transformed domain. These mathematical bounds provide insight into the performance under different sampling schemes in different transformed domains. Effectiveness of bounds was illustrated using sparse function recovery in Discrete Cosine domain under Random Walk (RW) based sampling. Proposed theoretical formulation predicts that functions with sparsity 2, 3, and 4 can be reconstructed with 606, 1048, and 1458 minimum number of samples respectively out of a total of 7653 measurements, while simulations show the actual average samples required to be 635, 1450, and 1700, respectively, indicating a close relationship.

Based on the theoretical foundation and moving beyond the traditional way of uniform sampling based CS for function recovery, we also illustrated how RW-based sampling can lead to phenomena awareness at different sensor locations, with minimal additional overhead. Central as well as distributed phenomena awareness was illustrated based on RW-based sampling of a large-scale distributed phenomenon, the temperature distribution over the State of Alabama. Performance bounds for CS-based phenomena discovery using a frame, an over-complete basis, instead of an orthogonal basis and other practical sampling schemes that accurately captured by motion models are under investigation.

CHAPTER 12

SUMMARY AND FUTURE WORK

We envision future wireless senor networks as collections of large number of sensor nodes, which are capable of sensing, wirelessly communicating, and information processing, deployed over vast terrains or on complex structures for years--learning, adapting to, and exploring the physical world. Sensor networks that combine sensing and actuation will be a critical component of many emerging cyber physical and ubiquitous networking applications, including healthcare, navigation, rescue, intelligent transportation, social networking, and gaming. Many such networks will be deployed in inaccessible or large terrains; thus, the lifespan will be heavily dependent on efficient data fusion and dissemination algorithms. The algorithms and techniques developed for such networks must operate under the constraints of low memory, limited energy/power, and low computational capability.

WSN "organization and structuring" commonly refers to developing a coordinate system, which contains the information required for data exchange, routing, node location identification, and boundary detection. Such coordinate systems may contain the information of directionality and connectivity, providing each node a virtual or an actual position. In large-scale networks, the only information that a node may contain upon deployment is its own identifier and neighbors' identifiers. Target tracking, habitat monitoring, underground plume tracking, underwater sensor networks, military surveillance applications, and healthcare applications are examples where networking, data fusion, and sensing algorithms can significantly benefit from the nodes having knowledge about their own coordinate and network structure. Currently available organization schemes can be divided into two main categories: physical information-based structuring and virtual information-based structuring. Geographic Coordinate System (GCS) is an example for the former and Virtual Coordinate Systems (VCS) and hierarchical addressing schemes such as clustering are examples for the latter.

Geographic/Physical Coordinate System (GCS) is a commonly used coordinate system in WSN context. Developing GCS requires costly equipment such as Global Positioning System (GPS) or signal strength measurements, of which the latter is error-prone due to the measurement noises introduced by the medium. Received Signal Strength Indication (RSSI), radio hop count, time difference of arrival, and angle of arrival are few such error-prone measurements. Moreover, GPS is infeasible in some WSN applications such as in indoor, underwater, forest, and metropolitan sensor networks. Thus, alternative coordinate systems that are independent from physical coordinates are proposed.

VCSs assign each node a virtual position, which captures the sense of relative position and/or connectivity of the actual network without the need for physical information. For many applications, especially those involved in data dissemination, what nodes actually need to be aware of is the relative position based on connectivity rather than the actual position. In fact, in the absence of anchors or erroneous information of actual anchor positions in GCS, nodes are oblivious to their real positions. Hence, it is not necessary to obtain costly actual coordinates, and a coordinate system that has relative information can provide an economical alternative for WSN algorithms. Due to the above reasons, the concept of VCSs shows great promise as a structuring scheme in WSN applications.

12.1 Research Contributions

In this research, we developed several techniques and algorithms to overcome the limitations and disadvantages associated with traditional VCS-based approaches due to lack of directionality and physical layout information. In addition, we also address the problem of determining the number and the placement of anchors in a unified manner. The main contributions of this research can be summarized as: (a) novelty filtering and Singular Value Decomposition (SVD) based techniques for ‘good’ anchor identification and compressed representation of VCS, (b) a transformation to obtain Directional Virtual Coordinate System (DVCS) that restores the lost directionality in VCS and the applications of DVCSs, (c) Topology Preserving Map (TPM) generation techniques that produce network maps with important physical features such as physical voids and boundary information, without the need for geographic information, (d) localization-free routing and boundary-detection schemes, and (e) distributed self-learning algorithms for each node to infer the information of its position and the rest of the nodes’ relative positions in the network. Each of the above contributions is described next.

The number of anchors and their placement play a crucial role in the performance of VCS-based algorithms. Moreover, the length of the coordinate, local minima problem, and identical coordinate problem are directly correlated with the anchor placement. Properties and issues of VCSs are investigated, and then a novelty filtering-based technique to identify and quantify the importance of each anchor was proposed. This technique can be used to select ‘good’ anchors from any underlying anchor selection scheme. The higher the size of a network, the larger the number of anchors required. Thus, the virtual address length of each node increases with network size. A VCS compression scheme based on SVD was proposed. Performance of data dissemination was not affected when using the proposed address compression technique.

Virtual coordinate systems do not have the cardinal direction information of nodes. A transformation that regains the lost directionality in VCS was proposed. The novel coordinate system generated under this transformation is called Directional Virtual Coordinate System (DVCS). Vector representation and angular estimation among virtual directions radically changed the use of VCS in WNSs. With these directional properties, for the first time it is possible to consider deterministic algorithms for routing in the virtual domain. We illustrated the benefits and the use of DVCS using a constrained tree network as an example.

A novel routing scheme called Directional Virtual Coordinate Routing (DVCR), which takes advantage of the Directional Virtual Coordinate domain, was proposed. DVCR significantly outperforms existing VCS routing schemes such as Convex Subspace Routing (CSR) and Logical Coordinate Routing (LCR), while achieving performance similar to the geographical routing scheme Greedy Perimeter Stateless Routing (GPSR) without the need for node location information.

Extreme Node Search (ENS) selects an effective and ‘good’ set of anchors based on a directional ordinate derived using two initial random anchors. ENS achieves significant performance improvement of greedy ratio in greedy forwarding with a significantly less number of anchors compared to that of randomly placed anchors, even on the boundary.

Fascinatingly, with the proposed anchor selection, DVCR outperforms GPSR in spite of the use of node location information in the latter.

Two schemes that are capable of extracting maps containing physical characteristics and topological features were developed. First scheme is based on SVD while the other is based on identifying near orthogonal directions in DVCS. Prior to this work, the details of networks' physical boundaries, voids, shape, etc., were considered lost in VCSs. The most significant feature of TPM is preservation of topology/neighborhood. The derived TPMs, ideally, should be homeomorphic (topologically isomorphic) to the physical layout of the sensor network, i.e., between two topological spaces there has to be a continuous inverse function. Thus, given the absolute position of a subset of nodes, global localization is realizable based on a non-linear transformation. The derived maps are close to the above ideal situation. Hence, TPMs is a geographical information free, relative map generation scheme that is based on VCS. The generalized SVD-based TPM generation scheme for 3D networks produces topology maps even in situations where obtaining accurate physical information is impossible, due to signal interference and complexity of triangulation in 3D.

Discovery of the TPM generation techniques that generate a map with important physical information embedded in it now provides the ability to develop algorithms without requiring physical information. Geo-Logical Routing (GLR) is a novel technique that brings the best of geographic domain and virtual domain to achieve higher routability at a lower cost. TPMs provide a better low-cost alternative for location information. Performance evaluation indicates that GLR significantly outperforms existing VC routing schemes: Convex Subspace Routing (CSR), Logical Coordinate Routing (LCR), and geographic scheme GPSR as well. Moreover, GLR achieves outstanding routing performance with lower path length compared to that of LCR, CSR, and GPSR. Proposed GLR scheme provides a unified framework for localization free, low-cost, and efficient family of routing schemes.

A novel method of identifying boundaries of sensor networks deployed on 2D and 3D surfaces was presented. VCS is a connectivity-based higher dimensional representation of the network where the network can have more than one embedding, making boundary identification challenging. TPM generated from VCS of a network is an identification of the correct embedding, which is the actual physical map, out of all the possible embeddings in the connectivity domain. The boundary detection scheme proposed based on 2D and 3D TPM is simple, energy-efficient, and computationally less intensive. Moreover, the proposed algorithm can be used with physical coordinates as well. The use of TPM-based boundary detection scheme for detecting dynamic event boundaries, such as those in plumes, in a distributive manner is also illustrated.

A node aware of the sensed phenomena of the network is useful in network management and data dissemination. 'Phenomena awareness' is either the Base-Station (BS) or any node discovering the monitoring phenomena. Moving beyond traditional uniform sampling-based Compressive Sensing (CS) for function recovery, we take the first step towards forming mathematical foundation, acquiring phenomena awareness for large-scale WSNs. We derived function reconstruction failure probability and minimum number of samples needed for any pragmatic sampling scheme under any suitable basis, which so far has been limited to uniform sampling in the Fourier domain. The extra number of samples required to achieve the same performance as uniform sampling is also obtained. The distinguishable novelty of our work is the deviation from the traditional uniform sampling, which makes it feasible to implement phenomena discovery at individual nodes distributive manner eliminating the constraints of the sampling scheme. The number of samples required, energy efficiency, and error in recovered phenomena show that random walk-based sampling is similarly effective as uniform sampling in CS recovery. This makes our proposed scheme a practical solution in applications where uniform sampling is less economical and infeasible.

So far, we have described the developed algorithms that are capable of routing, structuring via topology map generation and data fusion, keeping the computations and communications to a minimum, and optimizing the energy consumption in order to expand the lifespan of the WSNs. We envision future sensor networks that evolve by long-term learning and inference, achieving over time increasing levels of network and sensed phenomena awareness, thus with time becoming smarter and better at what they do. We use the term network/topology awareness to indicate a node's cognizance of the topology, shape, and boundary of the network and its place in that network. Algorithm and techniques such as TPM, boundary detection, and GLR described above provide the ability towards achieving such a vision.

A method for achieving network awareness at individual nodes via self-learning, with which wireless sensor nodes become cognizant of the network topology, network boundaries, and individual node positions in the network was proposed. Network awareness can significantly increase the flexibility, performance, and functionality of WSNs and facilitate intelligent distributed algorithms. We proposed a learning scheme, whereby the nodes, initially oblivious to their position in network as well as to network topology, gradually infer such information by listening to routine network traffic. The method works by individual nodes gleaning the hop distances to a set of nodes to build a VCS from which each node infers its own view of the network in the form of a TPM. Evaluation of this self-learning strategy demonstrates the convergence of each learning stage, the gradual development of network-awareness at a node with both 2D and 3D WSNs, and consistency among TPMs generated via independent self-learning at individual nodes. Routing is used as an example to demonstrate how the network can progressively switch to more and more effective algorithms as network awareness develops within nodes.

12.2 Future Work

More recently, Cyber Physical Systems (CPS) have emerged as a promising direction to enrich human-to-human, human-to-object, and object-to-object interactions in the physical world as well as in the virtual world. CPSs exploit the physical information collected by multiple diverse WSNs to bridge real and cyber spaces. CPS applications significantly benefit from the proposed economical and scalable algorithms for structuring, data dissemination, as well as achieving network awareness. However, several areas should be further investigated and improved. Several possible future extensions are listed next.

In order to increase the accuracy and scalability of topology preserving maps, non-linear transformation techniques such as curvilinear component analysis can be investigated. Another possible solution is to partition the network, generate TPM of partitions, and stitch the map of the partitions together. The approach of network partition and stitching not only makes the TPM accurate but also it makes the TPM generation scheme scalable. Some applications require exact physical information. Thus, identifying a transformation to achieve actual physical information from TPM with the knowledge of partial physical information is significantly more efficient than existing localization schemes.

VCS based strategies provide a promising approach for routing in 3D or higher dimensional networks such as Internet. Existing physical information-based routing schemes fail in 3D. Moreover, acquiring physical information itself is a challenge due to high interference. In contrast, connectivity information-based VCS and TPM are more appropriate and effective for routing in higher dimensional spaces.

In reality, the network connectivity is not static due to node failures, etc. Moreover, there can be a fraction of mobile nodes in the network or in some cases all the nodes may be mobile. Hence, VCS should be adaptive to such environments to avoid regeneration of VCS. Furthermore, natural networks as Internet exhibit scale-free structure where VCS may require a large number of anchors to overcome identical coordinates. Thus existing VCS approaches should either be modified or adapt to the network type.

Current TPM are generated using hop distance based VCS. One can investigate coordinate systems that have hop distances to anchors as well as physical properties that nodes sense as additional coordinates. This new matrix is called as similarity matrix rather than a coordinate system. TPM of this matrix will contain not only the connectivity information but also the physical phenomena that the network is observing. Such systems will be useful in social networking and P2P networks, where one can define the similarity of two nodes (users) based on their connectivity distance, activities/ groups/ hobbies, etc.

Moving beyond, the existing sensor network applications, cyber physical systems, smart grids, and ubiquitous networks, require adaptive, self-evolving, as well as reliable algorithms. In essence, this research laid a firm foundation to use of VCS in WSN applications without the need for physical coordinates and to achieve a novel concept of network awareness.

REFERENCES

[1] I. F. Akylidiz and J. M. Jornet, “The internet of nano-things,” IEEE Wireless Communication Networks, Dec. 2010.

[2] Akyildiz, S. Weilian, Y. Sankarasubramaniam, and E. E. Cayirci, “A survey on sensor networks,” Proc. IEEE Communications Magazine, 2002, pp. 102–114.

[3] J. Ai, A. A. Abouzeid, “Coverage by directional sensors in randomly deployed wireless sensor networks,” Springer Science+Business Media, 2006.

[4] J.N. Al-Karaki, and A.E. Kamal, “Routing techniques in wireless sensor networks: a survey,” IEEE Wireless Communications, Vol. 11, Dec. 2004, pp.6-28.

[5] A.A. Abbasi, M. Younis, “A survey on clustering algorithms for wireless sensor networks,” Computer Communications, 2007, pp. 2826–2841.

[6] I.F. Akyildiz, J.M. Jornet, “Electromagnetic Wireless Nanosensor Networks, Nano Communication Networks,” Nano Communication Networks, Vol. 1, March 2010, pp. 3–19

[7] N. Ahmed, and T. Natarajan, and K.R. Rao, “Discrete cosine transform,” Proc. IEEE Transactions on Computers, 1974, 90 – 93.

[8] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks", journal of Ad Hoc Networks, 2005, pp. 325 – 349.

[9] M. Awad, X. Jiang, and Y. Motai, “Incremental support vector machine framework for visual sensor networks,” Journal Advances in Signal Processing (EURASIP), 2007.

[10] J. Burrell, T. Brooke, and R. Beckwith, “Vineyard computing: sensor networks in agricultural production,” IEEE journal of Pervasive Computing, 2004, pp. 38 – 45.

[11] W.U. Bajwa and A. Pezeshki, “Finite frames for sparse signal processing,” to appear in Finite Frames, P. Casazza and G. Kutyniok, Eds. Cambridge, MA: Birkhäuser Boston, 2012.

[12] M. Baqer, and A.I. Khan, “Energy-Efficient Pattern Recognition Approach For Wireless Sensor Networks,” 3rd Int. Conf. Intelligent Sensors, Sensor Networks and Information, 3-6 Dec. 2007, pp: 509 – 514.

[13] H.M.N.D. Bandara, and A.P. Jayasumana, “An enhanced top-down cluster and cluster tree formation algorithm for wireless sensor networks,” Proc. 2nd Int. Conf. on Industrial & Information Systems, Aug. 2007,pp. 37-42.

[14] J. Bachrach and C. Taylor, "Localization in sensor networks," Ch. 9, Handbook of Sensor Networks, Stojmenovic (Editor), John Wiley 2005.

[15] H. Balakrishnan, R. Baliga, D. Curtis, M. Goraczko, A. Miu, B. Priyantha, A. Smith, K. Steele, S. Teller, and K. Wang, “Lessons from developing and deploying the cricket indoor location system,” in MIT Technical Report, Nov. 2003.

[16] Y. Bengio, J-F. Paiement , P. Vincent , “Out-of-sample extensions for lle, Isomap, MDS, eigenmaps, and spectral clustering,” In Advances in Neural Information Processing Systems, 2003

[17] T. Banka, G. Tandon, and A.P. Jayasumana, “Zonal Rumor Routing For Wireless Sensor Networks,” Int. Conf. Information Technology: Coding and Computing (ITCC), 2005, pp: 562 – 567.

[18] D. Braginsky, and D. Estrin, “Rumor routing algorithm for sensor networks,” 1st Workshop on Sensor Networks and Applications (WSNA), 2002.

[19] J.A. Boyan, and M.L. Littman, “Packet Routing In Dynamically Changing Networks: A Reinforcement Learning Approach,” In J. D. Cowan, G. Tesauro, and J. Alspector, editors, Advances in neural information processing systems, 1994, pp: 671-678.

[20] F. Celik, A. Zengin and S. Tuncel , “A Survey On Swarm Intelligence Based Routing Protocols In Wireless Sensor Networks ,” Int. Journal Physical Sciences, 4 Nov, 2010, pp. 2118-2126.

[21] CSU Sensor-Net Benchmarks, Available: http://www.cnrl.colostate.edu/Projects/VCS/, 2011

[22] A. Caruso, S. Chessa, S. De, and A. Urpi, “GPS free coordinate assignment and routing in wireless sensor networks,” Proc. 24th IEEE Joint Conf. of Computer and Communications Societies, Vol. 1, pp. 150- 160, Mar. 2005.

[23] Q. Cao and T. Abdelzaher, “Scalable logical coordinates framework for routing in wireless sensor networks,” ACM Transactions on Sensor Networks, Vol. 2, pp. 557-593, Nov 2006.

[24] K.K. Chintalapudi, R. Govindan, “Localized edge detection in sensor fields,” Proc. First IEEE Int. Workshop on Sensor Network Protocols and Applications, 11 May 2003, pp: 59 – 70.

[25] ClimateWizard ,PRISM Group, Oregon State University, created 4 Feb 2007, Available at:http://www.climatewizard.org/index.html

[26] E.J Candes, and T. Tao, “Decoding by linear programming” IEEE Transactions on Information Theory,2005, 4203 – 4215.

[27] E.J. Candes, and T. Tao, “Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies?”, IEEE Transactions on Information Theory, 2006, 5406 - 5425.

[28] D.L Donoho, “Compressed sensing”, IEEE Transactions on Information Theory, 2006, pp. 1289 -1306.

[29] A. Daubechies, R. DeVore, and M. Fornasier, and C. S. Gunturk, “Iteratively re-weighted least squares minimization for sparse recovery”,ArXiv e-prints, 2008.

[30] M. Ding, X. Cheng, “Robust event boundary detection in sensor networks - A mixture model based approach,” IEEE INFOCOM, April 2009, pp: 2991 – 2995.

[31] M. Ding, D. Chen, K. Xing, and X. Cheng, “Localized fault-folerant event boundary detection in sensor networks,” IEEE INFOCOM, March 2005, pp: 902 – 913.

[32] S. Duttagupta, K. Ramamritham, and P. Ramanathan, “Distributed boundary estimation using sensor networks,” IEEE Int. Conf. Mobile Adhoc and Sensor Systems (MASS), Oct. 2006, pp. 316-325.

[33] A. Doerr and K. Levasseur, “Applied Discrete Structures for Computer Science,” SRA, 1985.

[34] D. C. Dhanapala, “On Performance of Random Routing and Virtual Coordinate Based Routing in WSNs”, M.S. Thesis, Colorado State University, Fort Collins, CO, USA, 2009.

[35] D. C. Dhanapala, Q. Han and A.P. Jayasumana, “Performance of random routing on grid-based sensor networks,” Proc. IEEE Consumer Communications and Networking Conference (CCNC), Jan. 2009.

[36] D. C. Dhanapala and A. P. Jayasumana, "CSR: Convex Subspace Routing Protocol for WSNs," Proc. 33rd IEEE Conf. on Local Computer Networks, Oct. 2009.

[37] D. C. Dhanapala and A. P. Jayasumana, "Topology preserving maps from virtual coordinates for wireless sensor networks," Proc. 35rd IEEE Conf. on Local Computer Networks, Oct. 2010.

[38] D. C. Dhanapala and A.P. Jayasumana, “Dimension reduction of virtual coordinate systems in wireless sensor networks,” Proc. IEEE Ad-hoc and Sensor Networking Symposium (Globecom), Dec. 2010.

[39] D. C. Dhanapala and A.P. Jayasumana, “Geo-Logical Routing in wireless sensor networks,” Proc. 8th IEEE Communications Society Conf. on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), June 2011.

[40] D.C. Dhanapala and A.P. Jayasumana, "Anchor selection and topology preserving maps in wsns - a Directional Virtual Coordinate Based Approach," Proc.36th Annual IEEE Conf. on Local Computer Networks (LCN), 2011.

[41] D.C. Dhanapala and A.P. Jayasumana, "Directional virtual coordinate systems for wireless sensor networks," Proc. IEEE Int. Conf. on Communications (ICC), June 2011.

[42] D.C. Dhanapala, S. Mehta and A.P. Jayasumana, "Boundary detection of sensor and nanonetworks deployed on 2-d and 3-d surfaces," Proc. IEEE Ad-hoc and Sensor Networking Symposium (Globecom), 2011.

[43] B. Deng, G. Huang, "Algorithm of Creating Routing Table for WSNs Wireless Communications," Proc. 5th Int. Conf. Networking and Mobile Computing, 2009, pp. 1-5.

[44] A. Egorova-Forster, and A.L. Murphy, "Exploiting Reinforcement Learning for Multiple Sink Routing in WSNs," Proc. IEEE Int. Conf. Mobile Adhoc and Sensor Systems, Oct. 2007, pp: 1-3.

[45] I. Eyal, I. Keidar, and R. Rom, "Distributed Data Classification In Sensor Networks," Proc. 29th ACM SIGACT-SIGOPS, July 2010, pp: 151-160.

[46] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Angle Regression", ArXiv Mathematics e-prints, arXiv:math/0406456, 2004.

[47] F. Fazel, M. Fazel, and M. Stojanovic, "Random access compressed sensing in underwater sensor networks," Proc. 48th Annual Allerton Conf. on Communication, Control, and Computing, 2010, pp. 768 -774.

[48] C. Feng and S. Valaee, and Z. Tan, "Localization of wireless sensors using compressive sensing for manifold learning," Proc. 20th IEEE Int. Symp. on Personal, Indoor and Mobile Radio Communications, 2009, pp. 2715 -2719.

[49] Q. Fang, J. Gao and L.J. Guibas, "Locating and bypassing routing holes in sensor networks," Proc. IEEE INFOCOM, pp. 2458–68, 2004.

[50] R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler and S. Shenker, and I. Stoica, "Beacon vector routing: Scalable point-to-point routing in wireless sensor networks," Proc. 2nd Symposium on Networked Systems Design and Implementation, pp. 329–342, 2005.

[51] S.P. Fekete, M. Kaufmann, A. Kroller, and N. Lehmann, "A new approach for boundary recognition in geometric sensor networks," Proc. 17th Canadian Conf. on Computational Geometry, 2005, pp. 82-85.

[52] S. Funke, “Topological hole detection in wireless sensor networks and its applications,” Proc. 3rd ACM/SIGMOBILE Int. Workshop on foundations of Mobile Computing, DIAL-M-POMC, 2005.

[53] T. Goldstein and S. Osher, “The split Bergman method for L1-regularized problems,” SIAM Journal on Imaging Sciences, 2009, pp. 323-343.

[54] A.M. Khedr, W. Osamy and D. P. Agrawal, “Perimeter discovery sensor network,” Journal of Parallel and Distributed Computing, Nov. 2009, pp: 922-929.

[55] A. Kroller, S.P. Fekete, D. Pfisterer, and S. Fischer, “Deterministic boundary recognition and topology extraction for large sensor networks,” Proc. of 17th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA) 2006.

[56] C.F. García-Hernández, P.H. Ibargüengoytia-González, J. García-Hernández, and J.A. Pérez-Díaz, “Wireless sensor networks and applications: a survey,” Int. Journal of Computer Science and Network Security(IJCSNS), Mar. 2007, pp. 264-273.

[57] J. Gao, and L. Guibas, “Geometric algorithms for sensor networks,” Phil. Trans. R. Soc. A 370, 2012, PP. 27–51.

[58] R. Hartley and A. Zisserman, “ Multiple view geometry in computer vision,” Appendix 4, 2nd Edition, Cambridge University Press New York, 2003.

[59] S. Hla, Y.S. Choi, and J. S. Park, “Mobility Enhancement in Nanorobots by Using Particle Swarm Optimization Algorithm”Proc. Int. Conf. Computational Intelligence and Security (CIS), Dec. 2008, pp. 35 – 40.

[60] Implementing Mica2 MOTE sensor network. Available at: http://hamster.foxhollow.ca/TinyOS/Guide/Mica2-Install-Guide.pdf,2011

[61] Imote2. Available at: http://docs.tinyos.net/index.php/Imote2, 2011

[62] IP1 Radar Specification, CASA ERC. Available at : http://server.casa.umass.edu/~zink/ECE597S/Papers/IP5_radar_spec.pdf, 2011

[63] Y. Jiang, “Virtual coordinate based operation in dynamic and mobile networks (tentative)”, MS thesis, Colorado State University, CO,2013.

[64] A.P. Jayasumana, Q. Han, and T. Illangasekare, "Virtual Sensor Networks – A Resource Efficient Approach For Concurrent Applications," Proc. Int. Conf. Information Technology ITNG , Apr. 2007.

[65] W. Jia, T. Wang, G. Wang and M. Guo, “Hole avoiding in advance routing in wireless sensor networks,” Proc. Wireless Communications and Networking Conf. (WCNC), pp. 3519 – 3523, 2007.

[66] G. Jin and S. Nittel, “NED: An efficient noise-tolerant event and event boundary detection algorithm in wireless sensor Networks,” Proc. of 7th Int. Conf. on Mobile Data Management, 2006.

[67] E. Kranakis, H. Singh and J. Urrutia, “Compass routing on geometric networks,” Proc. 11th Canadian Conf. on Computational Geometry, Aug. 1999.

[68] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, “Geometric ad-hoc routing: of theory and practice,” Proc. 22nd ACM Int. Symposium on the Principles of Distributed Computing (PODC), July 2003.

[69] B. Karp and H. T. Kung., “Greedy perimeter stateless routing (GPSR) for wireless networks,” Proc. 6th annual ACM/IEEE Int. Conf. on mobile computing and networking (mobicom), 2000, pp. 243–254.

[70] K. Klues, “Power management in wireless networks,” Report. Available at : http://www1.cse.wustl.edu/~jain/cse574-06/energy_mgmt.htm, 2011

[71] H. Kulah, and K. Najafi, “Energy Scavenging From Low Frequency Vibrations by Using Frequency Up Conversion for Wireless Sensor Applications,” IEEE Sensors Journal, Vol. 8, 2008 , pp: 261 - 268.

[72] M. Kirby, “Geometric data analysis- An empirical approach to dimensionality reduction and the study of patterns,” John Wiley & Sons, 2001.

[73] N.D. Lane, H. Lu, and A.T. Campbell, “Ambient beacon localization: using sensed characteristics of the physical world to localize mobile sensors,” Proc. 4th Int. workshop embedded networked sensors, June 2007, pp: 38-42.

[74] List of wireless sensor nodes. Available at: http://en.wikipedia.org/wiki/List_of_wireless_sensor_nodes

[75] S. Lee, S. Pattem, M. Sathiamoorthy, B. Krishnamachari, and A. Ortega, “Spatially-Localized Compressed Sensing and Routing in Multi-hop Sensor Networks,” Proc. 3rd Int. Conf. on GeoSensor Networks, 2009, pp. 11-20.

[76] S. Lederer, Y. Wang and J. Gao; “Connectivity-Based Localization of Large Scale Sensor Networks with Complex Shape,” Proc. 27th IEEE INFOCOM, 2008 , pp: 789 – 797.

[77] B. Leong, S. Mitra, and B. Liskov, “Path vector face routing: geographic routing with local face information,” Proc. 13th IEEE Int. Conf. on Network Protocols (ICNP), Nov. 2005.

[78] Experimental comparison of RSSI-based localization algorithms for indoor wireless sensor networks,” Proc. workshop on Real-world wireless sensor networks, 2008, pp. 1-5.

[79] O. Liang, Y.A. Sekercioglu, and N. Mani, “A Low-Cost Flooding Algorithm for Wireless Sensor Networks,” IEEE Wireless Communications and Networking Conf., pp. 3495 – 3500, March 2007.

[80] E.A. Lee, “Cyber-Physical Systems - Are Computing Foundations Adequate?,” Position Paper for NSF Workshop On Cyber-Physical Systems: Research Motivation, Techniques and Roadmap, Oct. 2006.

[81] C-H Lin, B-H Liu, H-Y Yang, C-Y Kao, and M-J Tsai, “Virtual-coordinate-based delivery-guaranteed routing protocol in wireless sensor networks with unidirectional links”, Proc. IEEE INFOCOM, 13-18 April 2008, pp. 351-355.

[82] K. Liu and N. Abu-Ghazaleh, “Aligned virtual coordinates for greedy routing in WSNs,” Proc. IEEE Int. Conf. on Mobile Ad-hoc and Sensor Systems, Oct. 2006.

[83] K. Liu and N. Abu-Ghazaleh, “ Virtual coordinate backtracking for void traversal in geographic routing,” Proc. 5th Int. Conf. on Ad-Hoc Networks & Wireless, Aug. 2006.

[84] K. Liu and N. Abu-Ghazaleh, “Stateless and guaranteed geometric routing on virtual coordinate systems,” Proc. 5th IEEE Int. Conf. Mobile Ad Hoc and Sensor Systems (MASS). pp. 340-346, Oct. 2008.

[85] M.C. McCutcheon, H.J. Farahani, J.D. Stednick, G.W. Buchleiter and T.R. Green, "Effect of Soil Water on Apparent Soil Electrical Conductivity and Texture Relationships in a Dryland Field", Journal of Biosystems Engineering, 2006, pp. 19 - 32.

[86] S. Mehta, “Evaluation of topology generation in sensor networks,” MS Project Report, Colorado State University, CO, USA, 2011.

[87] J. Meng, H. Li and Z. Han, “Sparse event detection in wireless sensor networks using compressive sensing,” Proc. 43rd Conf. on Information Sciences and Systems,(CISS), 2009, pp.181 -185.

[88] G. Mainland, D.C. Parkes, and M. Welsh, “Decentralized, Adaptive Resource Allocation for Sensor Networks,” Proc. 2nd Symp. Networked Systems Design & Implementation, 2005, pp: 315-328.

[89] F. Martincic, and L. Schwiebert, “Distributed perimeter detection in wireless sensor networks,” Wayne State University, WSU-CSC-NEWS/03-TR03, July 2004.

[90] S. Olga, S. Robert, G. Matthias, and M. P. José, “On boundary recognition without location information in wireless sensor networks,” Proc. of 7th Int. Conf. on Information Processing in Sensor Networks, April 2008, pp.207-218.

[91] H.A.B.F. Oliveira, E. F. Nakamura, A. A. F. Loureiro, and A. Boukerche, “Error analysis of localization systems for sensor networks,” Proc. 13th ACM Int. Workshop on Geographic Information Systems, New York, USA, 2005

[92] K. Pearson, “On lines and planes of closest fit to systems of points in space,” Philosophical Magazine, 1901, pp: 559-572.

[93] J. Polastre, R. Szewczyk, and D. Culler, “Telos: enabling ultra-low power wireless research,” Proc. 4th Int. Symposium on Information Processing in Sensor Networks (IPSN), April 2005, pp. 364 – 369.

[94] R. Pan, J. Zhao, V. W. Zheng, J.J. Pan, D. Shen, S.J. Pan, and Q. Yang, “Domain-constrained semi-supervised mining of tracking models in sensor networks,” Proc. 13th ACM SIGKDD Int. Conf. knowledge discovery and data mining, 2007, pp: 1023-1027.

[95] L. Peshkin, and V. Savova, “Reinforcement Learning For Adaptive Routing,” Proc. Int. Joint Conf. Neural Networks, 2002, pp: 1825 – 1830.

[96] G. Quer, R. Masiero, D. Munaretto, M. Rossi, J. Widmer, and M. Zorzi, “On the interplay between routing and signal representation for Compressive Sensing in wireless sensor networks,” Proc. Information Theory and Applications Workshop, 2009,pp. 206 -215.

[97] Radio-frequency identification. Available at: http://en.wikipedia.org/wiki/Radio-frequency_identification

[98] S. T. Roweis, and L. K. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” Science, Vol 290, pp: 23239-2326, 22 Dec. 2000.

[99] J. Romberg, L1-Magic, available at: www.acm.caltech.edu/l1magic/, 2011.

[100] H. Rauhut, “Compressive sensing and structured random matrices,” Journal of Theoretical Foundations and Numerical Methods for Sparse Recovery, 2010,edited by Fornasier M.,pp. 1-92.

[101] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica , “Geographic routing without location information,” Proc. 9th Int. conf. on Mobile computing and networking, pp.96 - 108, 2003.

[102] Y. Shang, W. Ruml, Y. Zhang and M. Fromherz, “Localization from connectivity in sensor networks,” IEEE Transactions on Parallel and Distributed Systems, 2004, pp. 961–974.

[103] L. Shu, Y. Zhang, L.T. Yang, Y. Wang and M. Hauswirth, “Geographic Routing in Wireless Multimedia Sensor Networks,” Proc. 2nd Int. Conf. Future Generation Communication & Networking (FGCN), 2008, pp. 68 – 73.

[104] M. Sartipi, and R. Fletcher, “Energy-Efficient Data Acquisition in Wireless Sensor Networks Using Compressed Sensing,” Proc. Data Compression Conf. (DCC),2011, pp223 -232.

[105] H. Shokrzadeh, A.T. Haghighat, F. Tashtarian, A. Nayebi, “Directional rumor routing in wireless sensor networks,” Proc. 3rd IEEE/IFIP Int. Conf. Central Asia on Internet, 2007, pp.1 - 5.

[106] Sensor node. Available at: http://en.wikipedia.org/wiki/Sensor_node

[107] Smart spaces. Available at : http://vismod.media.mit.edu/vismod/demos/smartspaces/,2011

[108] Sensor networks for dummies, Available at: http://www.technologyreview.com/InfoTech/wtr_16607,300,p1.html?a=f&a=f, 2011

[109] M. Steyvers, “Multidimensional scaling,” Encyclopedia of Cognitive Science, 2002.

[110] Scientific Computing, When Cars can Talk, Available at: http://www.scientificcomputing.com/news-DS-When-Cars-can-Talk-032211.aspx?

[111] TelosB mote platform. Available at: http://www.willow.co.uk/html/telosb_mote_platform.html,2011

[112] Y. K. Tan and S. K. Panda, “Review of energy harvesting technologies for sustainable wsn, sustainable wireless sensor network,” Winston Seah (Editor) and Yen Kheng Tan (Editor-in-Chief), Available at: http://www.intechopen.com/articles/show/title/review-of-energy-harvesting-technologies-for-sustainable-wsn,2011

[113] M. J. Tsai, H. Y. Yang, and W. Q. Huang, “Axis based virtual coordinate assignment protocol and delivery guaranteed routing protocol in wireless sensor networks,” Proc. INFOCOM 2007, May 2007, pp: 2234-2242.

[114] M. Tubaishat, and S. Madria, “Sensor networks: an overview,” IEEE Potentials, April-May 2003,pp. 20-23.

[115] B. Tenenbaum, V. de Silva and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction,” Science, Vol 290, Dec. 2000, pp: 2319-2323.

[116] “Ubiquitous sensor networks (USN),” ITU-T Technology Watch Briefing Report Series, No. 4, Feb. 2008, Available at: http://www.itu.int/dms_pub/itu-t/oth/23/01/T23010000040001PDFE.pdf

[117] Y. Wang and H. Wu and N-F. Tzeng, “Cross-Layer Protocol Design and Optimization for Delay/Fault-Tolerant Mobile Sensor Networks,” IEEE journal on Selected Areas in Communications, 2008, pp. 809 -819.

[118] N. A. Weir, D. P. Sierra, and J. F. Jones, “A review of research in the field of nanorobotics,” Technical Report, Sandia National Laboratories, Oct 2005.

[119] Y. Wang, M. Martonosi, and L-S. Peh, “A supervised learning approach for routing optimizations in wireless sensor networks,” Proc. 2nd Int. workshop on Multi-hop ad hoc networks, 2006, pp: 79-86.

[120] E.W. Weisstein, "Frobenius norm," From Math World--A Wolfram Web Resource. http://mathworld.wolfram.com/FrobeniusNorm.html,2011.

[121] E.W. Weisstein, "Adjacency matrix," From Math World--A Wolfram Web Resource. http://mathworld.wolfram.com/AdjacencyMatrix.html,2011.

[122] C.Wang and L. Xiao, “Sensor localization under limited measurement capabilities,” IEEE Network, Vol. 21, May - June 2007, pp: 16-23.

[123] L-C. Wuu, W-B. Li, and W-C. Kuo, “Detour Routing Protocol for Geographic Sensor Networks,” Int. Conf. Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 , pp. 505 - 510 .

[124] L. Xiang, and J. Luo, “Compressed Data Aggregation for Energy Efficient Wireless Sensor Networks,” Proc. 8th IEEE Communications Society Conf. on Sensor and Ad Hoc Communications and Networks (SECON), 2011.

[125] Y. Yu, R. Govindan and D. Estrin, “Geographical and energy aware routing: a recursive data dissemination protocol for WSNs,” UCLA CS Dept Tech. Rept., UCLA/CSD-TR-01-0023, May 2001.

[126] Y. Zhang, D. Zhang, and C. Huang, “On the construction and maintenance of virtual backbone for wireless sensor networks,” Proc. Int. Conf. on Convergence Information Technology (ICCIT) 2007, pp. 1813-1817.

[127] H. Zhou, H. Wu, S. Xia, M. Jin, and N. Ding, “A distributed triangulation algorithm for wireless sensor networks on 2D and 3D surface,” Proc. IEEE INFOCOM, 2011 , pp: 1053 – 1061.

[128] O. Saukh, R. Sauter, M. Gauger, and P. J. Marron, and K. Rothermel, “On boundary recognition without location information in wireless sensor networks,” ACM Transactions on Sensor Networks, June 2010.

[129] C. Zhang, Y. Zhang, and Y. Fang, “Localized algorithms for coverage boundary detection in wireless sensor networks,” Wireless Networks, Feb. 2007.

[130] B. Zhang, X. Cheng, N. Zhang, Y. Cui, Y. Li, and Q. Liang, “Sparse target counting and localization in sensor networks based on compressive sensing ,” Proc. IEEE INFOCOM, 2011,pp. 2255 -2263.

[131] M. Zhang and M. C. Chan and A.L. Ananda, "Connectivity monitoring in wireless sensor networks," Journal of Pervasive and Mobile Computing,2010, pp. 112-127.

[132] H. Zhou, S. Xia, M. Jin, and H. Wu, “Localized Algorithm for Precise Boundary Detection in 3D Wireless Networks.” Proc. IEEE 30th Int. Conf. on Distributed Computing Systems (ICDCS), June 2010, pp: 744 – 753.

APPENDIX A

CSU BENCHMARK NETS AND THEIR VIRTUAL COORDINATE GENERATION

MATLAB codes for four example CSU benchmark networks (a) sparse grid (grid with missing nodes), (b) circle with three circular voids, (c) network in a building, and (d) spiral shaped network followed by VCS generation are given next. Physical coordinates of these networks can be found at http://www.cnrl.colostate.edu/Projects/VCS/.

A.1 Sparse grid

```
% INPUT: Grid size N,number of missing nodes , DEAD is the % symbol of a
missing node.
% OUTPUT: Network node arrangement and the flag of 1 if connected.

function [Grid ConnectedFlag] = gridwithholes(DEAD,N,HOLES)
K=0;
for I= 1: HOLES
    holesTemp=int8((N-3)*rand(1,2))+2;
    if K==0
        K=1;
            holes(2*K)=holesTemp(1);
            holes(2*K-1)=holesTemp(2);
    else
        SameholesFlag=1;
        while (SameholesFlag)
            Flag=0;
            for index=1:K
                if holes(2*index)==holesTemp(1) && holes(2*index-
1)==holesTemp(2)
                    Flag=1;
                    break;
                end
            end

            if Flag==1
                holesTemp=int8((N-1)*rand(1,2))+1;
            else
                SameholesFlag=0;
            end
        end
            K=K+1;
            holes(2*K)=holesTemp(1);
            holes(2*K-1)=holesTemp(2);
    end

end

Grid=ones(N,N);% grid of N by N
for i=1:2:2*HOLES-1
    Grid(holes(i+1),holes(i))=DEAD;
end

if HOLES>0
    ConnectedFlag=connectivity(N,HOLES,holes,Grid,DEAD); % check whether
the resultant network is connected
else
    ConnectedFlag=1;
    holes=0;
end
```

A.2 Circle with voids

```
% INPUT: size of the network.
% OUTPUT: Network node arrangement and the flag of 1 if connected.

function [Grid ConnectedFlag]= circlewithholes(N)

C=N/2;C1=10;C21=22;C22=15;C31=10;C32=20; %Center locations of the circles
Grid=zeros(N,N);
z=4;z1=N/2;z2=6;
for x=1:N;
 for y=1:N;
     if (x-C)^2+(y-C)^2<=z1^2 &&(x-C1)^2+(y-C1)^2>z^2 && (x-C21)^2+(y-
C22)^2>z2^2 && (x-C31)^2+(y-C32)^2>z^2
         Grid(x,y)=1;
     end
 end
end
ConnectedFlag=connectivity(N,HOLES,holes,Grid,DEAD); % check whether the
resultant network is connected
```

A.3 Building network

```
% INPUT: Grid size N ,number of missing nodes , DEAD is the symbol used
for a missing node.
% OUTPUT: Network node arrangement and the flag of 1 if connected.

function [Grid ConnectedFlag]= Building(DEAD,N,HOLES)

z=1;
if HOLES>0
    for x=2:4
        for y=2:10
            holes(2*z)= y;
            holes(2*z-1)= x;
            z=z+1;
```

```
         end
     end
     for x=6:10
         for y=2:10
             holes(2*z)= y;
             holes(2*z-1)= x;
             z=z+1;
         end
     end

     for x=12:16
         for y=12:20
             holes(2*z)= y;
             holes(2*z-1)= x;
             z=z+1;
         end
     end
     for x=18:20
         for y=12:20
             holes(2*z)= y;
             holes(2*z-1)= x;
             z=z+1;
         end
     end
     for x=12:15
         for y=2:5
             holes(2*z)= y;
             holes(2*z-1)= x;
             z=z+1;
         end
     end
         for x=17:20
         for y=2:5
             holes(2*z)= y;
             holes(2*z-1)= x;
             z=z+1;
         end
         end
     for x=22:29
         for y=2:5
             holes(2*z)= y;
             holes(2*z-1)= x;
             z=z+1;
         end
     end
     for x=12:17
         for y=7:10
             holes(2*z)= y;
             holes(2*z-1)= x;
             z=z+1;
         end
     end
    for x=19:23
         for y=7:10
             holes(2*z)= y;
             holes(2*z-1)= x;
             z=z+1;
         end
    end
    for x=25:29
         for y=7:10
             holes(2*z)= y;
             holes(2*z-1)= x;
             z=z+1;
         end
     end
     for x=22:29
         for y=12:14
             holes(2*z)= y;
             holes(2*z-1)= x;
             z=z+1;
         end
     end
     for x=22:29
         for y=20:22
             holes(2*z)= y;
             holes(2*z-1)= x;
             z=z+1;
         end
      end
     for x=22:29
         for y=24:26
             holes(2*z)= y;
             holes(2*z-1)= x;
             z=z+1;
         end
     end
     for x=22:29
         for y=28:29
             holes(2*z)= y;
             holes(2*z-1)= x;
             z=z+1;
         end
     end
     for x=2:6
         for y=12:16
             holes(2*z)= y;
             holes(2*z-1)= x;
             z=z+1;
         end
     end
     for x=2:6
         for y=18:23
             holes(2*z)= y;
             holes(2*z-1)= x;
             z=z+1;
         end
     end
     for x=2:6
         for y=25:29
```

```
         holes(2*z)= y;
         holes(2*z-1)= x;
         z=z+1;
     end
 end
for x=8:10
     for y=12:19
         holes(2*z)= y;
         holes(2*z-1)= x;
         z=z+1;
     end
end
for x=8:10
     for y=21:24
         holes(2*z)= y;
         holes(2*z-1)= x;
         z=z+1;
     end
 end
 for x=8:10
     for y=26:29
         holes(2*z)= y;
         holes(2*z-1)= x;
         z=z+1;
     end
 end
 for x=12:15
     for y=22:29
         holes(2*z)= y;
         holes(2*z-1)= x;
         z=z+1;
     end
 end
 for x=17:20
     for y=22:24
         holes(2*z)= y;
         holes(2*z-1)= x;
         z=z+1;
     end
 end
 for x=17:20
     for y=26:29
         holes(2*z)= y;
         holes(2*z-1)= x;
         z=z+1;
     end
 end

 for x=7
     for y=12:16
         holes(2*z)= y;
         holes(2*z-1)= x;
         z=z+1;
     end
 end
```

```
    for x=11
        for y=7:10
            holes(2*z)= y;
            holes(2*z-1)= x;
            z=z+1;
        end
    end
end

Grid=ones(N,N);% grid of N by N

for i=1:2:2*HOLES-1
    Grid(holes(i+1),holes(i))=DEAD;
end
    % connectivity check
if HOLES>0
    ConnectedFlag=connectivity(N,HOLES,holes,Grid,DEAD)
else
    ConnectedFlag=1;
    holes=0;
end
```

A.4 Spiral network

```
% INPUT: Grid size N
% OUTPUT: Network node arrangement and the flag of 1 if connected.

function [Grid ConnectedFlag]= spiral(N)

t = linspace(0,4*pi,1000);
Grid=zeros(N,N); %grid of N by N
sh=N/2;
for a=0:0.1:3
 x = round((t+a).*cos(t))+sh;
 y = round((t+a).*sin(t))+sh;
 for i=1:length(x)
    Grid(x(i),y(i))=1;
 end
end
ConnectedFlag=connectivity(N,HOLES,holes,Grid,DEAD)
```

A.5 VCS generation

```
% INPUT: Adjacency matrix Net of N nodes. Anchor nodes are given by vector
Anchor
% OUTPUT: VCS

function CC= VCS(Net,N,Anchor)

CoordinateSet=zeros(N,N);
Dummy=1000;% change the diagonal elements to a large value
for i=1:N
    CoordinateSet(i,i)=Dummy;
end
```

```
for x=1:N
    for y=1:N
        if Net(x,y)==0
            CoordinateSet(x,y)=Dummy;
        end
    end
end

flag=0;T=1;
CoordinateSet(Anchor,Anchor)=0;
index=Anchor;
Stack(1)=index;
n=1;p=1;k=1;
PreIndex=Anchor;
Lp=2;a=1;setFlag=0;
PreIndex(1)=index;
num=0;

for j=1:N % colomns
        for i=1:N % rows
            if Net(i,index)==1 % ie there is a connection
                if n>1
                    for x=1:length(Array)
                        if Array(x)==i
                            setFlag=1;
                            break;
                        end
                    end
                    if setFlag ~= 1
                        setFlag=0;
                        Array(n)=i;
                        n=n+1;
                        num=num+1;
                    else
                        setFlag=0;
                    end
                elseif n==1
                    Array(n)=i;
                    n=n+1;
                    num=num+1;
                end
                CoordinateSet(i,index)=CoordinateSet(index,PreIndex(a))+1;
            end
        end
        a=a+1;
        PreIndex(Lp:Lp+num-1)=index;
        Lp=Lp+num;
        num=0;

        if ~isempty(Array) && p <= length(Array)
            while T==1
                for q=1:length(Stack)
                    if Array(p)~=Stack(q)
                        flag=1;
                        T=0;
                        break;
                    elseif Array(p)==Stack(q)
                        a=a+1;
                        p=p+1;
                        break;
                    end
                end
            if flag == 1
                T=0;
            end
            end
            if flag == 1
                index=Array(p);
                Stack(length(Stack)+1)=Array(p);
                p=p+1;
                T=1;
                flag=0;
            end
        else
            clear Array
            n=1;
        end
end
CC=min(CoordinateSet'); % Return VCS
```

APPENDIX B

SIMULATOR FOR TOPOLOGY MAP GENERATION, DIRECTIONAL VIRTUAL COORDINATE GENERATION AND ROUTING ALGORITHMS

MATLAB codes for topological coordinate generation proposed in Chapter 5 (B.1), Directional Virtual Coordinate Generation discussed in Chapter 4 (B.2), Directional Virtual Coordinate Routing discussed in Chapter 4(DVCR, B.3) and Geo-Logical Routing proposed in Chapter 8 (GLR, B.4) are given next.

B.1 Topological coordinates generation

```
% INPUT - virtual coordinate system
% OUTPUT – topological coordinates
[U,S,V]=svd(VCS);
PC=U*S;
Psvd=[PC(2,:), PC(3,:)];
```

B.2 Directional Virtual Coordinates

```
% INPUT-Virtual coordinate system
% OUTPUT - Directional virtual coordinate system
count=0;
for i=1:Anchor_NO-1
      for j=i+1:Anchor_NO
            count=count+1;
            AGrid(count,:)=NewAnchors([VCS(i,:); VCS(j,:)]);
            AGrid(count,:)= AGrid(count,:)./abs(VCS(i,AnchorArray(i))-
VCS(i,AnchorArray(j)))/2;
      end
end

function A= NewAnchors(Grid)
for i=1:max(size(Grid))
    AplusB(i)=Grid(1,i)+Grid(2,i);
    AminusB(i)=Grid(1,i)-Grid(2,i);
    A(i)=AplusB(i)*AminusB(i);
end
```

B.3 Directional Virtual Coordinate Routing - DVCR

```
% INPUT - DVCS, source, destination
% OUTPUT- successfully routed or not and path length
if dest_reached==0 % destination not reached
    indx=1;
    for x=1:N
        if Net(x,currentNode)== 1 && PrevNodeArray(currentNode)~=x  &&
PrevFWDNodeArray(currentNode)~=x
            neighbors_Dest(indx)=Grid_Destanse(x);
            Array(indx)=x;%store the index of the node
            indx=indx+1;
        end
    end
    if indx==1
        i=0;
        for xneigh=1:N
            if Net(xneigh,currentNode)== 1 %
                i=i+1;
                neighCoord1(i)=Grid(1,xneigh);% neighbors coordinate array
                diffArray1(i)=neighCoord1(i)-Grid(1,currentNode);
                neighCoord2(i)=Grid(2,xneigh);% neighbors coordinate array
                diffArray2(i)=neighCoord2(i)-Grid(2,currentNode);
                nei(i)=xneigh;
            end
        end

        a1=min(abs(diffArray1));
        b1=max(abs(diffArray1));
        a2=max(abs(diffArray2));
        b2=min(abs(diffArray2));
        k=1;
        for j=1:length(nei)
            L1=abs(Grid(1,Dest)-Grid(1,nei(j)));
            L2=abs(Grid(2,Dest)-Grid(2,nei(j)));
            h1(k) =(L1*b2 - L2*b1)/(a1*b2 - a2*b1);
            h2(k) =-(L1*a2 - L2*a1)/(a1*b2 - a2*b1);
            k=k+1;
        end

        [~,ID]=min(abs(h1)+abs(h2));
        PrevFWDNodeArray(currentNode)=nei(ID);
        PrevNodeArray(nei(ID))=currentNode;
        currentNode=nei(ID);
    else
        [value1 index1]=min(neighbors_Dest);% get the minimum Distanced
neighbor
        [value2 index2]=max(neighbors_Dest);% get the maximum Distanced
neighbor
        terminator= Grid_Destanse(Dest);


        if (value1-terminator)==0
            %check whether neighbor is the destination
           a=1;

            for c=1:length(neighbors_Dest)
                if value1==neighbors_Dest(c) && c==Dest
```

```
                    dest_reached=1;
                    break;
                end
            end
            if dest_reached==0
                i=0;
                for xneigh=1:N
                    if Net(xneigh,currentNode)== 1  %
                        i=i+1;
                        neighCoord1(i)=Grid(1,xneigh);% neighbors
coordinate array
                        diffArray1(i)=neighCoord1(i)-Grid(1,currentNode);
                        neighCoord2(i)=Grid(2,xneigh);% neighbors
coordinate array
                        diffArray2(i)=neighCoord2(i)-Grid(2,currentNode);
                        nei(i)=xneigh;
                    end
                end

                a1=min(abs(diffArray1));
                b1=max(abs(diffArray1));

                a2=max(abs(diffArray2));
                b2=min(abs(diffArray2));
                k=1;
                for j=1:length(nei)
                    L1=abs(Grid(1,Dest)-Grid(1,nei(j)));
                    L2=abs(Grid(2,Dest)-Grid(2,nei(j)));
                    h1(k) =(L1*b2 - L2*b1)/(a1*b2 - a2*b1);
                    h2(k) =-(L1*a2 - L2*a1)/(a1*b2 - a2*b1);
                    k=k+1;
                end

                [~,ID]=min(abs(h1)+abs(h2));
                PrevFWDNodeArray(currentNode)=nei(ID);
                PrevNodeArray(nei(ID))=currentNode;
                currentNode=nei(ID);

            else
                currentNode=Dest;
                if NoOfHops>TTL
                    success=success+1;
                end
            end

            NoOfHops=NoOfHops+1;
        elseif value1<=value2 %compare min Distance of neighbor with
current Destance
            DestCurrent=Grid_Destanse(currentNode);

            if value1<= DestCurrent % if there exist a neighbor with lower
Destance

                a=1;
                for c=1:length(neighbors_Dest)
                    if value1==neighbors_Dest(c)
                        pool_nodes(a)=Array(c);
                        a=a+1;
                    end
                end

                II=ceil(length(pool_nodes)*rand(1));
                min_neighbors=pool_nodes(II);
                PrevFWDNodeArray(currentNode)=min_neighbors;
                PrevNodeArray(min_neighbors)=currentNode;
                currentNode=min_neighbors;
                NoOfHops=NoOfHops+1;
            else %if there is no neighbor with lower Destance
                i=0;
                for xneigh=1:N
                    if Net(xneigh,currentNode)== 1 &&
PrevNodeArray(currentNode)~=xneigh  &&
PrevFWDNodeArray(currentNode)~=xneigh %
                        i=i+1;
                        neighCoord1(i)=Grid(1,xneigh);% neighbors
coordinate array
                        diffArray1(i)=neighCoord1(i)-Grid(1,currentNode);
                        neighCoord2(i)=Grid(2,xneigh);% neighbors
coordinate array
                        diffArray2(i)=neighCoord2(i)-Grid(2,currentNode);
                        nei(i)=xneigh;
                    end
                end

                a1=min(abs(diffArray1));
                b1=max(abs(diffArray1));

                a2=max(abs(diffArray2));
                b2=min(abs(diffArray2));
                k=1;
                for j=1:length(nei)
                    L1=abs(Grid(1,Dest)-Grid(1,nei(j)));
                    L2=abs(Grid(2,Dest)-Grid(2,nei(j)));
                    h1(k) =(L1*b2 - L2*b1)/(a1*b2 - a2*b1);
                    h2(k) =-(L1*a2 - L2*a1)/(a1*b2 - a2*b1);
                    k=k+1;
                end

                [~,ID]=min(abs(h1)+abs(h2));
                PrevFWDNodeArray(currentNode)=nei(ID);
                PrevNodeArray(nei(ID))=currentNode;
                currentNode=nei(ID);
                NoOfHops=NoOfHops+1;
            end
        elseif value1>value2
            i=0;
            for xneigh=1:N
                if Net(xneigh,currentNode)== 1 %
                    i=i+1;
                    neighCoord1(i)=Grid(1,xneigh);% neighbors coordinate
array
                    diffArray1(i)=neighCoord1(i)-Grid(1,currentNode);
                    neighCoord2(i)=Grid(2,xneigh);% neighbors coordinate
array
                    diffArray2(i)=neighCoord2(i)-Grid(2,currentNode);
                    nei(i)=xneigh;
                end
            end

            a1=min(abs(diffArray1));
            b1=max(abs(diffArray1));

            a2=max(abs(diffArray2));
            b2=min(abs(diffArray2));
            k=1;
            for j=1:length(nei)
                L1=abs(Grid(1,Dest)-Grid(1,nei(j)));
                L2=abs(Grid(2,Dest)-Grid(2,nei(j)));
                h1(k) =(L1*b2 - L2*b1)/(a1*b2 - a2*b1);
                h2(k) =-(L1*a2 - L2*a1)/(a1*b2 - a2*b1);
                k=k+1;
            end

            [~,ID]=min(abs(h1)+abs(h2));
            PrevFWDNodeArray(currentNode)=nei(ID);
            PrevNodeArray(nei(ID))=currentNode;
            currentNode=nei(ID);
            NoOfHops=NoOfHops+1;
        end
    end

clear neighbors_Dest;clear Array;clear pool_nodes;clear neighCoord1;clear
neighCoord2;clear diffArray1;clear diffArray2;clear h1;clear h2;clear nei

else % destination reached

    success=success+1;
    minimaFlag=1;
    break;
end
```

B.4 Geo-Logical Routing - GLR

```
%input- VCS, TCS, source, destination, Output- successfully routed or not
and path length
if dest_reached==0 % destination not reached
    indx=1;Noneighborflag=1;
    for x=1:N
        if Net(x,currentNode)== 1 && PrevNodeArray(currentNode)~=x  &&
PrevFWDNodeArray(currentNode)~=x % %
            if Grid_distanse(x)>=0
                neighbors_dist(indx)=Grid_distanse(x);
                Array(indx)=x;%store the index of the node
                indx=indx+1;
                Noneighborflag=0;
            end
        end
    end

    if Noneighborflag==1
        % send the packet to anchor
        [currentNode
Hops]=GreedyFwdBacktracking(N,currentNode,Ai,Net,DistAnchor);
        NoOfHops=NoOfHops+Hops;
        Grid_distanse=Grid_distanseVC;
    else

    neighbors_dist;
    [value1 index1]=min(neighbors_dist);% get the minimum distanced
neighbor
    [value2 index2]=max(neighbors_dist);% get the maximum distanced
neighbor
    terminator= Grid_distanse(Dest);

    if (value1-terminator)==0
        %check whether neighbour is the destination
        for x=1:N
            if Net(x,Array(index1))== 1 && x==Dest %sum(abs(Grid(:,x)-
Grid(:,Dest)))==0
                dest_reached=1;%destination reached
            end
        end
        if Array(index1)==Dest %sum(abs(Grid(:,x)-Grid(:,Dest)))==0
                dest_reached=1;%destination reached
        end

        for c=1:length(neighbors_dist)
            if value1==neighbors_dist(c)  && c==Dest %sum(abs(Grid(:,c)-
Grid(:,Dest)))==0
                dest_reached=1;
                break;
            end
        end
        if dest_reached==0
            PrevNodeArray=zeros(size(Grid_distanse1));
            PrevFWDNodeArray=zeros(size(Grid_distanse1));
            switch BackTrack
                case 0
                    Grid_distanse=Grid_distanseVC;
                    BackTrack=1;
                case 1
                    Grid_distanse=Grid_distanse1;
                        BackTrack=2;
                case 2
                    a=1;
                    for c=1:length(neighbors_dist)
                        if value1==neighbors_dist(c)
                            pool_nodes(a)=Array(c);
                            a=a+1;
                        end
                    end

                    II=ceil(length(pool_nodes)*rand(1));
                    min_neighbors=pool_nodes(II);
                    PrevFWDNodeArray(currentNode)=min_neighbors;
                    PrevNodeArray(min_neighbors)=currentNode;
                    currentNode=min_neighbors;
                    NoOfHops=NoOfHops+1;
                    BackTrack=3;
                case 3
                    [currentNode
Hops]=GreedyFwdBacktracking(N,currentNode,Ai,Net,DistAnchor);
                    NoOfHops=NoOfHops+Hops;
                    Grid_distanse=Grid_distanseVC;
                    BackTrack=0;
            end
        else
            currentNode=Dest;
            if NoOfHops>TTL
                success=success+1;
            end
            NoOfHops=NoOfHops+1;
        en
    elseif value1<=value2 %compare min distance of neighbor with current
distance
        DistCurrent=Grid_distanse(currentNode);
        if value1<= DistCurrent % if there exist a neighbor with lower
distance
            a=1;
            for c=1:length(neighbors_dist)
                if value1==neighbors_dist(c)
                    pool_nodes(a)=Array(c);
                    a=a+1;
                end
            end
            II=ceil(length(pool_nodes)*rand(1));
            min_neighbors=pool_nodes(II);
            PrevFWDNodeArray(currentNode)=min_neighbors;
            PrevNodeArray(min_neighbors)=currentNode;
            currentNode=min_neighbors;
            DistCurrent=Grid_distanse(currentNode);
            NoOfHops=NoOfHops+1;
        else %if there is no neighbor with lower distance
            PrevNodeArray=zeros(size(Grid_distanse1));
            PrevFWDNodeArray=zeros(size(Grid_distanse1));
            switch BackTrack
                case 0
                    Grid_distanse=Grid_distanseVC;
                    BackTrack=1;
                case 1
                    Grid_distanse=Grid_distanse1;
                        BackTrack=2;
                case 2
                    a=1;
```

```
                        for c=1:length(neighbors_dist)
                            if value1==neighbors_dist(c)
                                pool_nodes(a)=Array(c);
                                a=a+1;
                            end
                        end
                        II=ceil(length(pool_nodes)*rand(1));
                        min_neighbors=pool_nodes(II);
                        PrevFWDNodeArray(currentNode)=min_neighbors;
                        PrevNodeArray(min_neighbors)=currentNode;
                        currentNode=min_neighbors;
                        NoOfHops=NoOfHops+1;
                        BackTrack=3;
                    case 3
                        [currentNode
Hops]=GreedyFwdBacktracking(N,currentNode,Ai,Net,DistAnchor);
                        NoOfHops=NoOfHops+Hops;
                        Grid_distanse=Grid_distanseVC;
                        BackTrack=0;
                end
            end
        elseif value1>value2
            PrevNodeArray=zeros(size(Grid_distanse1));
            PrevFWDNodeArray=zeros(size(Grid_distanse1));
                switch BackTrack
                    case 0
                        Grid_distanse=Grid_distanseVC;
                        BackTrack=1;
                    case 1
                        Grid_distanse=Grid_distanse1;
                            BackTrack=2;
                    case 2
                        a=1;
                        for c=1:length(neighbors_dist)
                            if value1==neighbors_dist(c)
                                pool_nodes(a)=Array(c);
                                a=a+1;
                            end
                        end

                        II=ceil(length(pool_nodes)*rand(1));
                        min_neighbors=pool_nodes(II);
                        PrevFWDNodeArray(currentNode)=min_neighbors;
                        PrevNodeArray(min_neighbors)=currentNode;
                        currentNode=min_neighbors;
                        NoOfHops=NoOfHops+1;
                        BackTrack=3;
                    case 3
                        [currentNode
Hops]=GreedyFwdBacktracking(N,currentNode,Ai,Net,DistAnchor);
                        NoOfHops=NoOfHops+Hops;
                        Grid_distanse=Grid_distanseVC;
                        BackTrack=0;
                end
        end
```

```
    end

  clear neighbors_dist;clear Array;clear pool_nodes;clear
neighbors_XYdist;

else % destination reached

    success=success+1;
    minimaFlag=1;
    break;
end
```