Geographic Routing for Point-to-Point Data Delivery in Wireless Sensor Networks

SHAO TAO

SCHOOL OF COMPUTING
NATIONAL UNIVERSITY OF SINGAPORE

2010

GEOGRAPHIC ROUTING FOR POINT-TO-POINT DATA DELIVERY IN WIRELESS SENSOR NETWORKS

SHAO TAO

A THESIS SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHYLOSOPHY
SCHOOL OF COMPUTING
NATIONAL UNIVERSITY OF SINGAPORE

2010

Acknowledgements

First of all, I would like to thank my supervisors Prof. A. L. Ananda and Prof. Chan Mun Choon for their guidance and patience in the past 5 years. Prof. Ananda is always patient and kind enough to guide me through any issues that I have encountered. I would like to thank him for offering me the opportunity to follow him and giving me the freedom to choose the research topics that I work on. I especially want to thank Prof. Ananda for trusting me to assist him in the courses of CS2105 and CS3103 that he teaches. It has been the most invaluable experience in my entire school life, from which I get to know what I am capable of, have gained my confidence and have made quite a few good friends with the students in the classes. I would like to thank Prof. Chan for providing enormous amount of suggestions on my work and correcting me on the critical issues that I failed to realize all the time. Prof. Chan has a descent personality of being super friendly to all the students and highly tolerant to the errors we made occasionally. His talent of pointing out the flaws without making one feel bad is what I really appreciate every time.

I would also like to thank my parents for respecting the decisions that I made along the way. I have their full support under all circumstances.

Finally, I enjoyed the time to work with my colleagues in the Communication and Internet Research Lab. It has been a great pleasure.

Contents

Abstract

As the design of sensor network applications diversifies, besides flooding and converge-cast, the point-to-point delivery is often required to support more complex communication schemes. Due to the constraints in the current sensor platforms, the traditional ad hoc routing protocols are not scalable. Geographic routing is an attractive option, because its localized packet forwarding procedure obviates the requirement of routing tables.

While wireless networks are generally deployed in three-dimensional environments, most geographic routing protocols are designed and evaluated in a two-dimensional space. Existing face routing protocols rely on a graph planarization procedure, which is not applicable to 3D networks. The greedy forwarding methods are vulnerable to local minimum cases, leading to frequent delivery failures in sparse networks. We present a tree-based routing protocol for 3D wireless networks named *Spherical Coordinate Routing* (SCR), that uses connectivity-based greedy forwarding to obtain efficient routing paths and a spherical coordinate tree to guarantee packet delivery. SCR can deploy multiple recovery trees simultaneously for better routing efficiency and resilience against network dynamics.

Hop count vector-based routing protocols can also be integrated with tree-based routing to improve routing efficiency. However, when the entire hop count vector is used to address each node, the communication and storage overhead in the packets is often too high to be employed in a large scale. We apply a dimension reduction technique, *Principal Component Analysis* (PCA), to reduce the control overhead in data packets. Compared to the original hop count vector, the embedding coordinates generated from the PCA algorithm preserve the network geometry with much lower overhead, making their use more practical on the current sensor platforms. Simulation results show that the coordinates computed by PCA can achieve higher packet delivery ratio, lower hop stretch and shorter flooding range with the same packet overhead.

List of Tables

List of Figures

List of Abbreviations

ACK	Acknowledgement
AODV	Ad hoc On-Demand Distance Vector Routing
BVR	Beacon Vector Routing
CCW	Counter Clock Wise
CLDP	Cross Link Detection Protocol
CTP	Collection Tree Protocol
CW	Clock Wise
DSDV	Destination-Sequenced Distance Vector Routing
DSR	Dynamic Source Routing
ETX	Expected Transmission Count
GFG	Greedy Face Greedy
GFRIS	Greedy Face Routing with Idenfitication Support
GG	Gabriel Graph
GOAFR+	Greedy Other Adaptive Face Routing plus
GPS	Global Positioning System
GPSR	Greedy Perimeter Stateless Routing
GPVFR	Greedy Path Vector Face Routing
LCR	Logical Coordinate Routing
LDT	Localized Delaunay Triangulation Graph
LQI	Link Quality Indicator
MAC	Medium Access Control
NoGeo	Geographic Routing without Location Information
PCA	Principal Component Analysis
PRR	Packet Reception Ratio
RFID	Radio-Frequency IDentification
RNG	Relative Neighbour Graph
RSSI	Received Signal Strength Indicator
RTX	Re-transmission
SCR	Spherical Coordinate Routing
SVD	Singular Value Decomposition
UDG	Unit Disk Graph
VPCR	Virtual Polar Coordinate Routing

List of Publications

1. "Greedy Face Routing with Identification Support for Wireless Networks", Shao Tao, A. L. Ananda and Mun Choon Chan, *Proceedings of the 16 International Conference on Computer Communications and Networks (ICCCN 2007)*, Honolulu, Hawaii, USA, 2007.

2. "Greedy Hop Distance Routing Using Tree Recovery on Wireless Ad Hoc and Sensor Networks", Shao Tao, A. L. Ananda and Mun Choon Chan, *Proceedings of the IEEE International Conference on Communications (ICC 2008)*, Bei Jing, P.R.China, 2008.

3. "Spherical Coordinate Routing for 3D Wireless Ad-hoc and Sensor Networks", Shao Tao, A. L. Ananda and Mun Choon Chan, *The 33rd IEEE Conference on Local Computer Networks (LCN 2008)*, Montreal, Quebec, Canada, 2008.

4. "Practical Connectivity-based Routing in Wireless Sensor Networks with Dimension Reduction", Shao Tao, A. L. Ananda and Mun Choon Chan, *Proceedings of the 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2009)*, Rome, Italy, 2009.

5. "Greedy Face Routing with Face Identification Support in Wireless Networks", Shao Tao, A. L. Ananda and Mun Choon Chan, *accepted for publication in Computer Networks*, 2010.

Chapter 1

Introduction

In the past decade, applications of wireless sensor networks have expanded from ecosystem monitoring [1], intrusion detection [2] to more diverse areas, such as precision agriculture [3], personal health care [4] and road traffic planning [5]. The ubiquitous deployment of wireless sensor networks can be attributed to its unique characteristics compared to the conventional specialized sensor equipment and wired networks. The radio transceivers [6] and sensor modules are compact in size, which allows them to be deployed efficiently without imposing significant impact on the environment or alternating the behavior of target objects. Sensor nodes are equipped with low power radios and processors to operate with low voltage [7]. The energy-efficient design of sensor nodes prolongs the network lifetime, which is critical for applications running in remote or hazardous areas, where it might be difficult or even prohibitive to replenish the power supply. The wireless communication nature in sensor networks obviate the requirement of wired connections for data feedback. The development tool kit provides flexible control of the hardware components through low-level interfaces, such that the embedded programs are highly customizable to fulfill the application requirements.

To collect information from the target area, sensor networks rely on not only the function of the sensors, but also the cooperation of the communication layers. The work presented in this thesis belongs to the category of routing protocols, particularly geographic routing protocols. We target on the problem of providing point-to-point delivery service in 3D wireless sensor networks. We also investigate the techniques to reduce the packet overhead of hop count vector-based routing, in order to improve routing efficiency in 3D networks in terms of delivery ratio, routing latency and path length.

1.1 Necessity of Point-to-Point Communication

Many wireless sensor networks exhibit many-to-one, uni-directional traffic patterns during data collection, where a simple spanning tree-based method will suffice. As a more generic form of communication, we believe that an point-to-point delivery service provides a foundation to diversify the design of sensor applications. Some typical instances that require the point-to-point delivery service include the following.

- For Geographic Hash Table-based data-centric storage [8], each data item is mapped to a location

in the network. The nodes closest to the hashed location will become the storage nodes and the data will be forwarded to the storage nodes through geographic routing methods. The point-to-point communication service allows distributed storage to be implemented across the network in order to avoid hot spots and the single point of failure problem.

- Transmission control algorithms [9] proposed in sensor networks enforce adaptive transmission rates to avoid congestion, which requires feedback messages to be relayed back to the source. For reliable delivery of these feedback messages, point-to-point communication is preferred over flooding, in order to reduce communication overhead at intermediate hops.

- In wireless sensor and actuator networks used for smart farming [10], reactions will be triggered upon the detection of target events. For example, if the soil moisture level is too low, the base station node should inform the pump to deliver more water to the area. The instructions must be sent back to nodes at various locations for the system response to take place, in which case a collection tree will not suffice.

- In sensor networks offering in-network query processing [11], regular nodes are allowed to handle the queries, besides obtaining results from the sink. In such scenarios, the point-to-point communication service allows information to be pulled directly from specific network areas, which is a useful feature especially for network diagnosis during the deployment.

Therefore, the point-to-point routing protocols will allow more sophisticated features to be introduced to the sensor networks and greatly improve its applicability in many aspects.

1.2 Options and Challenges

Despite the similarities between the wireless sensor networks and the traditional 802.11 networks in deployment and radio communication, the constraints from the sensor hardware components and software stack render the conventional ad hoc routing protocols infeasible for sensor networks.

The proactive ad hoc routing protocols, such as DSDV [12], maintain a routing entry for each node in the network to avoid the path discovery process. On sensor networks where data reporting is infrequent, constantly maintaining the routing table will incur redundant control traffic. The on-demand ad hoc routing protocols [13] employ a flooding method for route discovery, which will introduce longer delay and increase the energy consumption at all intermediate hops. The source routing protocols [14] store the entire routing path in the packet header, which is acceptable for the 802.11 networks, where a packet size can reach 1 KBytes. Given that the maximum packet size is significantly smaller (e.g. 128 bytes in TinyOS-2) in sensor networks, the source routing approach will significantly reduce the payload length in each packet and impose a control overhead prohibitive for deployments with a large network diameter.

Geographic routing protocols can be adopted to address the above issues. In geographic routing, each node is identified by its coordinates. The path discovery process is replaced by the node address lookup procedure and packet forwarding is performed by comparing the coordinates at each hop in a localized manner, obviating the requirement for routing tables. As the nodes are addressed by the coordinates, the

packet header will remain constant, regardless of the scale of the deployment or the network diameter. The geometric properties of the underlying topology are utilized to ensure the success of delivery.

Many early sensor network deployments are designed for outdoor monitoring tasks [15], where the nodes were placed on a plane. As indoor sensor networks become popular [16, 17], the deployments often spread over multiple floors of a building, forming a 3D topology. In this thesis, we focus on geographic routing protocols that can provide reliable packet delivery efficiently in both 2D and 3D network topologies. The greedy forwarding algorithms can be applied regardless of the dimensionality of the space; however, greedy forwarding requires accurate coordinates and local minimum cases commonly found in sparse networks will lead to delivery failures.

The face routing protocols [18, 19] complement greedy forwarding with a perimeter mode to route the packets out of the local minimum area using planar graphs. Due to the irregularities of the radio range and the influences from the environmental factors (e.g. obstacles and interference), the correct planarization procedure [20] will require more than local information and often incur high communication overhead. Since the concept of planar graph is not applicable to 3D space, planarization algorithms and face routing protocols cannot be used to provide guaranteed packet delivery in 3D networks.

The tree-based routing protocols [21, 22] create a hierarchical structure over the network, in which the parent nodes contain the aggregated location information for the subtree nodes. By traversing the subtrees, tree-based routing protocols can provide reliable point-to-point delivery in 3D networks.

The existing tree-based routing protocols are often designed for 2D networks and the performance of these protocols has not been studied in 3D networks extensively. The construction of the tree topology requires the coordinate information, that may not be available on some deployments. Localization algorithms can be incorporated to provide virtual coordinates; however, the localization errors may affect the routing performance, which should be carefully evaluated.

Some tree-based protocols (e.g. VPCR) assume that the tree can be built in a balanced manner and the short cut links connecting the sibling nodes can be exploited frequently to reduce the path length and alleviate the hot spot problem. This efficiency is hard to achieve, because a balanced tree is difficult to build for at least two reasons. Firstly, the network connectivity information must be used to determine the placement of the root; however, this information is often not known a priori. Secondly, each node selects the parent independently, which makes it difficult to evenly distribute the nodes among the branches without some form of coordination. Therefore, an alternate greedy forwarding method should be integrated into tree-based routing, since the tree structure alone will fail to provide efficient routing paths.

Greedy forwarding can be applied directly on the node coordinates when the location information of each node is already known. However, due to the cost of GPS devices and limitations in the deployment environment (e.g. indoor space), the location information may not be immediately available. The hop count vector-based routing protocols [23, 24] designate some nodes as beacons and use the vector containing the hop distances to all beacons as high-dimensional coordinates to address the nodes. Even without the actual location information, greedy forwarding algorithms can still be used for point-to-point routing under this configuration.

By using greedy methods, hop count vector-based routing protocols naturally suffer from delivery failures caused by local minimum cases. To improve the delivery ratio, it is a common approach

to increase the number of beacons [25], such that higher resolutions can be obtained from the high-dimensional coordinates. While the delivery performance can be improved, the large set of beacons also impose higher cost in terms of storage space, control traffic and packet overhead.

The intrinsic dimensionality of the sensor network is only two or three; thus, the high-dimensional coordinates contain large amount of redundancies due to correlated hop count values. As sensor network applications run on resource-constrained platforms, such redundancies should be reduced; however, the existing hop count vector-based routing protocols do not have any effective methods to perform this action.

1.3 Thesis Contributions

In this thesis, we present the Spherical Coordinate Routing protocol (SCR), that is designed and evaluated specifically in 3D networks for point-to-point communication. We also introduce a Principal Component Analysis-based dimension reduction algorithm (PCA) to convert the hop count vectors into low-dimensional coordinates, such that greedy forwarding can be applied at lower cost. The contributions of our work are summarized below.

- We provide a 3D routing protocol, named Spherical Coordinate Routing (SCR), that performs point-to-point routing in 3D networks without location information. As a 3D variant of VPCR, SCR employs a 3D localization algorithm derived from NoGeo, which assigns Euclidean coordinates to nodes based on the network connectivity. A spherical coordinate tree is constructed in the network that assign the coordinates based on a node's projection order on the XY and YZ planes and the parent's coordinates. To improve the routing efficiency and overcome the imbalance in the tree structure, SCR applies greedy forwarding based on the hop-count vectors available from the localization step, instead of relying on the scarce and unordered sibling links. Multiple trees can be utilized simultaneously during packet forwarding in SCR to enhance the resilience to clustered node failures. The delivery ratio of SCR is $22\% \sim 63\%$ higher than that of BVR, LCR and NoGeo and the hop stretch of SCR is 13% to 58% lower than that of VPCR at medium to high node densities.

- To reduce the packet overhead in hop count vector-based routing, we apply a network embedding scheme based on Principal Component Analysis (PCA). The PCA algorithm performs dimension reduction on the hop count vectors, such that each node can estimate the most effective coordinates in a distributed manner. The embedding coordinates can preserve the network geometry with much lower overhead, making their use more practical on the current sensor platform. We have implemented the PCA-based dimension reduction algorithm on MICAz and TelosB motes and performed both simulation and experiments on a large-scale sensor network testbed containing 127 nodes. For the experiments, we also developed a hierarchical location service with address caching to examine the reliability and workload distribution of the address queries, which is a fundamental requirement for all geographic routing algorithms.

1.4 Thesis Organization

The rest of the thesis is organized as follows:

- In Chapter 2, we present some background knowledge about geographic routing and give an overview of the existing point-to-point geographic routing protocols.
- In Chapter 3, we introduce *Spherical Coordinate Routing* (SCR). We give details on the 3D localization method, the procedure to allocate the spherical coordinates and the tree recovery algorithm in SCR. We evaluate the performance of SCR in 3D network topologies with clustered node failures.
- Chapter 4 introduces the network embedding scheme based on *Principal Component Analysis* (PCA). We evaluate the efficiency of PCA through both simulations and analysis using the network topology measured from a medium-scale testbed with 48 MICAz nodes.
- Chapter 5 presents the implementation details of the PCA protocol and the performance results of PCA collected from the Indriya testbed with 127 TelosB motes.
- The conclusion and future works are presented in Chapter 6.
- A survey of sensor applications is given in Appendix A. The challenges in sensor networks and active research areas are discussed in Appendix B. The link measurement results of the Indriya testbed are presented in Appendix C for reference.

Chapter 2

Background and Related Work

In this chapter, we introduce the definition of geographic routing, the metrics and tools commonly used for performance evaluation, followed by the related work.

2.1 Definition of Geographic Routing

Geographic routing protocols perform packet forwarding through node coordinates obtained from GPS devices or localization methods. The mapping between the nodes and the coordinates is provided by the location service. Unlike the conventional ad hoc routing protocols in 802.11 networks, the source node does not need to determine the entire path through a path discovery procedure. It will obtain the destination's address from the location server and allow routing decisions to be made at each hop based on local information. Instead of having a routing table to store all possible destinations, in geographic routing, each node has a neighbor table containing the coordinates of the neighboring nodes. The next hop is selected from the neighbors to minimize the distance to the destination.

Due to the network structure or the anomalies in the coordinates [26], a local minimum case will occur when none of the neighbors is closer to the destination than the current hop. Some geographic routing protocols route the packets out of the dead ends through recovery methods, such as scoped flooding, face routing or tree traversal. Therefore, geographic routing protocols can be categorized according to the coordinate system, the distance function and the recovery measure. Table 2.1 summarizes five types of geographic routing protocols commonly referenced in the literature.

Table 2.1: Comparison of Point-to-Point Geographic Routing Protocols

Type	*Coordinate System*	*Distance Function*	*Recovery Measure*
Greedy Forwarding e.g.: MFR [27], DREAM [28], Compass [29]	physical coordinates	Euclidean, projected distance, angular distance	(none)
Face Routing e.g.: GFG [30], GPSR [18], GOAFR+ [19]	physical, virtual coordinates	Euclidean distance	face traversal
Tree-based Routing e.g.: VPCR [21], GDSTR [22]	physical, virtual, polar coordinates	Euclidean, angular distance	tree traversal
Hopcount Vector-based Routing e.g.: LCR [31], HopID [32], BVR [24]	hop count vectors	Euclidean, Manhattan distance	scoped flooding, fall back towards beacon
Randomized Algorithms e.g.: RGR [33]	physical coordinates	Euclidean distance	random selection

2.1.1 Performance Metrics

The performance of geographic routing can be evaluated by various metrics, such as the *packet delivery ratio*, the *hop stretch*, the *flooding range*, the *number of transmissions per delivery*, the *delivery latency* and the *maximum storage cost*. The packet delivery ratio is computed as the number of packets received by the destinations divided by the number of packets sent by the sources. For protocols that apply greedy forwarding as the primary routing procedure, the packet delivery ratio reflects the efficiency of the coordinate system and the distance function at the given node densities. The hop stretch is the ratio between the length of the path discovered by the protocol and the length of the shortest path. In an ideal network with perfect links, a lower hop stretch means that the routing protocol provides more efficient routing paths with lower delay and lower energy consumption. The number of transmissions per delivery represents the total number of times a packet is forwarded before it reaches the destination. When scoped flooding is utilized in the recovery step, the flooding range will indicate the amount of redundancy caused by the duplicate packets. The delivery latency is the delay between the moment when a packet is sent by the source and the moment when it is received by the destination. The maximum storage cost represents the amount of short-term and long-term routing states that need be maintained at each node during the routing process, which is critical for memory-constrained sensor platforms. A routing protocol with high storage demand may sacrifice the performance or function of other software stacks running concurrently.

2.1.2 Evaluation Tools

Some commonly used tools for evaluating geographic routing protocols include MATLAB, OMNet++ [34], *ns-2* [35], TinyOS [36] and TOSSIM. MATLAB is normally used for simulations where the link quality variations and MAC layer operations can be ignored. OMNet++ and *ns-2* provide various physical layer models and MAC layer protocols to simulate the effects of network bandwidth, network contention and environmental noise. TinyOS is the de facto standard development tool kit for implementing sensor network protocols and applications on the Mica and Telos series motes. The TinyOS Simulator (TOSSIM) simulates the TinyOS network stack at the bit/packet level and the simulation code can be directly compiled to the binary executable with minimum modifications. Due to the level of details involved, simulations using TOSSIM may take much longer to complete for complex algorithms (e.g. prioritized packet transmission) running on large-scale networks (e.g. 1000 nodes), compared to MATLAB or customized simulators.

2.2 Geographic Routing Protocols

We discuss five types of point-to-point geographic routing protocols in this section.

2.2.1 Greedy Forwarding

Takagi et al. [27] used the Most Forwarding within R (MFR) routing algorithm to analyze the optimal transmission radius in a wireless network. In MFR routing, the next hop is chosen as the neighbor that

can make the most progress towards the destination when projected on the source and destination line segment.

Finn [37] proposed the Cartesian routing algorithm for the Internet, which associates each gateway with a Cartesian location represented by the longitude and latitude. The routing decision is made at the intermediate gateways to minimize the geographical distance between the next hop and the destination, in order to reduce path length and facilitate node mobility across the network boundary.

The DREAM protocol [28] was designed to perform geographic routing in mobile networks. Each mobile node periodically broadcasts its current location and speed to the entire network. The source node will determine a forwarding region based on the last reported position and velocity of the target. A copy of the packet will be forwarded to all neighbors in the forwarding region to improve the delivery ratio. Simulation results show that the DREAM protocol can deliver 80% of the packets without any recovery methods.

A primitive version of Compass Routing [29] selects the neighbor that forms the smallest angle along the source-destination line as the next hop. Geometric analysis shows that this method will not fail on Delaunay graphs; however, it may create routing loops on other planar graphs.

Due to the simplicity and near-optimal hop stretch performance, geographic routing protocols will often adopt greedy forwarding as part of the routing procedure. However, despite their effectiveness in dense networks, the distance metrics used by the greedy algorithms may be prone to local minimum cases in sparse networks, leading to frequent delivery failures. To route packets out of the local minimum nodes, various types of recovery algorithms have been proposed, such as scoped flooding, face routing and tree-based routing.

2.2.2 Face Routing

Face routing is performed on a planar graph generated by removing the cross links on the network connectivity graph. On the planar topology, a *face* refers to a void area bounded by a series of links. The line drawn from the local minimum node to the destination intersects a group of adjacent faces, which can be traversed by using the left/right hand rule. With the right hand rule, the packet will sequentially visit the connecting links on a face in the *counterclockwise* direction. The *clockwise* direction is applied for the left hand rule similarly. When a link crossing the local minimum-destination line is detected during face traversal, a face switch will be triggered if the intersection is closer to the destination than the current local minimum node. With the face switch procedure, the packet will move to the adjacent face and gradually approach the destination by applying face traversal and face switch in an alternating manner. If the packet traverses the entire face without resuming greedy forwarding or triggering a face switch, it means the destination is disconnected and the packet will be dropped.

The GFG [30] protocol is one of the earliest geographic routing protocols that combine greedy forwarding with face routing to achieve guaranteed packet delivery. The similar idea was also used by GPSR [18], which applied the face routing method in mobile ad hoc networks and obtained a significant performance improvement over DSR in simulations. As both GFG and GPSR use the simple right/left hand rule to traverse the faces, packets may be diverted to the longer side of a face before the next face switch, even if a more efficient path exists in the opposite direction.

The GOAFR+ [19] protocol was proposed to improve the worst case performance by doing a bounded search on each face. The bounded search procedure explores a face in both directions to find a node closer to the destination within an adaptively increasing bounded area, providing the optimal worst case performance of $O(c^2)$, where c is the cost of the shortest path in terms of hop count or distance.

The GPVFR [38] protocol enables each node to store information for multi-hop neighbors residing on the same face, which allows better guessing of correct forwarding directions. The face routing method can be further improved by performing face traversal on a connected dominating set of the topology and exploiting the shortcut paths [39] in high node density regions. A detailed survey on the geographic face routing protocols is presented in [40].

Kim et al. [20] showed that certain static face switch and face traversal rules may actually fail to deliver packets on a connected planar graph. Frey et al. [41] conducted a further examination on this problem and proposed a more proactive face switch algorithm that performs a face switch at every progressive intersection point. It forwards the packet to the side closer to the destination across the intersecting link, and guarantees packet delivery on all planar graphs with both left and right hand rules for face traversal.

We conducted preliminary research on face switch algorithms to provide guaranteed packet delivery on general planar graphs. We found that the correct face switch is possible if each face can be uniquely identified by a face ID. The resulted face routing protocol is named *Greedy Face Routing with Face Identification Support* (GFRIS) and the summary of the algorithm is given as follows.

According to the face routing classification method described in [41], GFRIS is a *continuative strategy* which uses both left and right hand rules for face traversal. We use S to denote the source node or the current local minimum node where face routing is initiated and use D to represent the destination node. Face switch occurs only if a link intersects the $\overline{SD}$ line at a location closer to destination D than S and the face IDs of both regions on two sides of the crossing link are different. The *after crossing* variant is used during face switch, where the packet will step through the crossing link to enter the next face and the face traversal direction will be switched from the left hand rule to the right hand rule or vice versa. GFRIS switches to greedy forwarding when there is a neighbor closer to the destination than the current local minimum node. A GFRIS face switch example is provided in Fig. 2.1. Assume a packet p arrives at node N from the previous node P. Node N has a link intersecting line $\overline{SD}$ at location x closer to destination *D* than S. The packet traverses the links by *dir* rule, where *dir* is either the *left* or *right* hand rule. The face switch procedure at node *N* is summarized in Algorithm 1.

Node N has four neighboring nodes A, B, C and P, which constitute four angles: $\angle$*BNA*, $\angle$*ANP*, $\angle$*PNC* and $\angle$*CNB*. We assume the face IDs for these angles are F_1, F_2, F_3 and F_4 respectively. Node N receives a packet p from node P. If the face traversal direction for packet p is *dir* = *left*, the incoming angle for packet p is $\angle$*PNC*, whose face ID is F_3. Otherwise, the incoming angle is $\angle$*ANP* with face ID F_2. Once the incoming face ID for the packet p is determined, the outgoing face ID can be retrieved by computing the outgoing angle for p according to the coordinates of destination D and the neighboring nodes of N. Assuming that the destination D is located within the area bounded by $\angle$*CNB*, the outgoing angle for packet p is $\angle$*CNB* with face ID F_4. Since GFRIS follows the *after crossing* variant, the outgoing link will be the intersecting link. The face traversal direction in the new face depends on the intersecting link. In Fig. 2.1, if line $\overline{SD}$ intersects the *ccw_start* link *NC* of outgoing angle $\angle$*CNB*, the next hop will be C and the face traversal direction will follow the *right* hand rule. If line $\overline{SD}$ intersects the *ccw_end*

Algorithm 1 Face Switch Algorithm of GFRIS

Let $\alpha(\beta)$ be the incoming(outgoing) angle of packet p
if *p.dir = left*, **then** $\alpha.ccw\_start = P$
else $\alpha.ccw\_end = P$ ▷ α is determined
Compute β by location of N, D and neighbors of N.
Let face ID of incoming angle α be $F_\alpha = \alpha.faceid$
Let face ID of outgoing angle β be $F_\beta = \beta.faceid$
if $F_\alpha \neq F_\beta$ **then**
    if $K = \beta.ccw\_start$, **then** set $dir = right$
    else set $dir = left$
    Send packet p to next hop K. □
else ▷ skip face switch on equal face IDs
    Forward packet p based on original dir. □
end if

link *NB* of angle $\angle$*CNB*, the next hop will be B and the face traversal direction will follow the *left* hand rule.

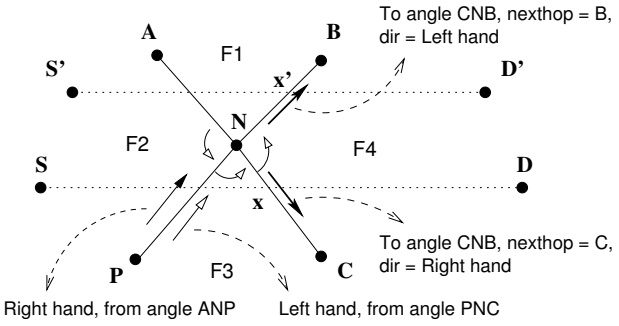


Figure 2.1: Face Switch Rules in GFRIS

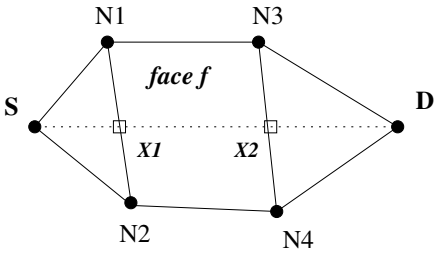
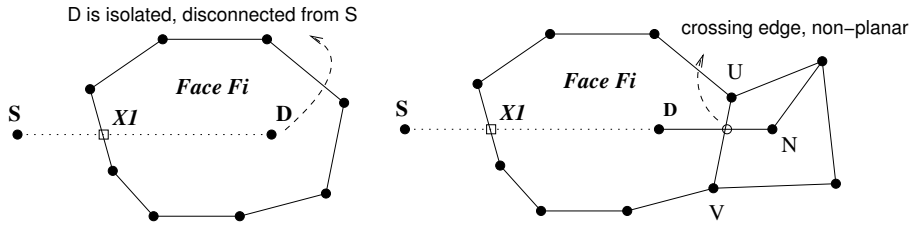


Figure 2.2: Face f intersects $\overline{SD}$ at ≥ 2 locations

The purpose for performing selective face switch is to avoid oscillating face traversal while ensuring that the packet is passed to a face closer to the destination. To show that the face switch procedure in GFRIS guarantees packet delivery on planar graphs, we need to prove the following Lemma 1.

Lemma 1. *Assume S and D are two connected vertices on a planar graph. If a link N_1N_2 on face f intersects line $\overline{SD}$ for the first time at location X_1, where $X_1 \neq N_1$ or N_2, face f will have at least one more intersection X_2 with $\overline{X_1D}$, where $dist(X_2, D) < dist(X_1, D)$*(as shown in Fig. 2.2)*. If destination D resides on face f, the second intersection $X2$ will be D.*

Proof: We prove the lemma by contradiction. If face f intersects line $\overline{SD}$ at X_1, and line segment $\overline{X_1D}$ is partially contained in face f but does not intersect f at a second location, destination D will be located inside face f. There can be two possible scenarios: D is isolated within f as shown in Fig. 2.3(a) or D is connected through a link or a series of links crossing the boundary of face f as shown in Fig. 2.3(b). Since the graph is planarized before face routing is applied and the source S and destination D should remain connected on the planarized topology, neither case will happen. From this contradiction, we can see that Lemma 1 is correct. □



(a) D is isolated in face F_i (b) the graph is not planar

Figure 2.3: Two cases if line segment $\overline{SD}$ intersects face F_i only once

In general, for a pair of connected nodes S and D on a planar graph, if the boundary of a face f intersects line segment $\overline{SD}$, there will be *multiple* intersections $X_1, X_2, \ldots, X_n (n \geq 2)$. We sort the intersection points based on their positions such that $|X_1D| > |X_2D| > \ldots > |X_nD|$. By traversing along the links on face f, the packet can travel from X_1 to X_n, approaching the destination D. Next, we briefly prove that the GFRIS face switch procedure will guarantee the delivery based on Lemma 1.

Theorem 1. *GFRIS guarantees delivery for two connected vertices S and D on a planar graph.*

Proof: Assume the line segment $\overline{SD}$ crosses a sequence of regions f_1, f_2, ..., f_k separated by the intersecting links L_1, L_2, ..., L_j as shown in Fig. 2.4. If two consecutive regions f_i and f_{i+1} belong to the same face, they will have the same face ID and GFRIS will not initiate face switch, the packet will follow the same traversal direction and ultimately reach f_{i+1} from f_i. Without losing generality, we assume f_i and f_j belong to different faces if $i \neq j$, which means a face switch will occur at each intersection X_i along the $\overline{SD}$ line. After each face switch, the packet will traverse a face closer to the destination. On a planar graph with a limited number of nodes, there are only a limited number of faces to traverse [42]; therefore, the packet will arrive at the last face f_k, which is the closest face to D. From Lemma 1, the second intersection X_{k+1} of f_k and $\overline{SD}$ is closer than the first intersection X_k. As there is no more face after the intersecting link at X_{k+1}, the last intersection between face f_k and line segment $\overline{SD}$ is the destination D, i.e. $X_{k+1} = D$, implying that the destination node D is located on face f_k. By traversing the links on the last face f_k, the packet will reach destination D. □

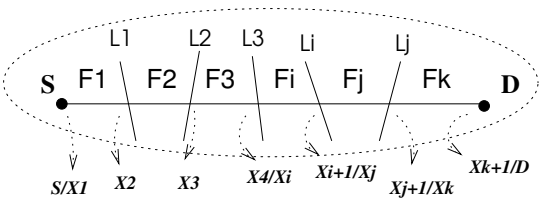


Figure 2.4: Face switch along the sequence of F_1, F_2, ..., F_k

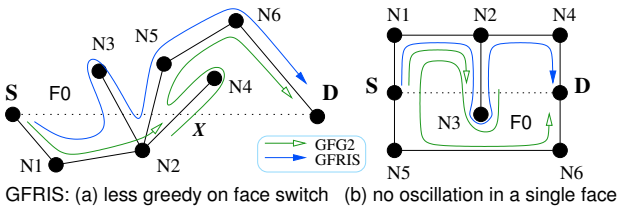


Figure 2.5: Face Switch Difference: GFG2 and GFRIS

The differences between the face switch procedures in GFG2 and GFRIS are illustrated in Fig. 2.5. The GFG2 protocol initiates face switch whenever a link intersects line $\overline{SD}$ at a closer location towards D. GFRIS further requires that the face IDs of the regions on two sides of the crossing link must be different. While GFRIS is less greedy at the face switch step compared to GFG2, GFRIS can reduce *face*

traversal oscillation within a single face, since the face switch will not be activated and the packet will not traverse the same face in alternating directions during normal face traversal.

The face routing algorithms work on a planar graph created from the full connectivity graph, such that no two links intersect each other and the reachability among the nodes remain intact. The commonly used planar graphs include the Relative Neighbor Graph (RNG) [43] and the Gabriel Graph (GG) [44], both of which can be efficiently constructed in a decentralized manner. The Localized Delaunay Triangulation Graph (LDT) [45] is a *t-spanner* of the original connectivity graph. The shortest path that can be found in a LDT graph for a node pair is no more than a constant t times longer than the shortest path in the original graph. The RNG, GG and LDT planarization techniques assume that the node communication range is constant, which is invalid in a real network deployment due to link asymmetries, interference and obstacles [46]. The Cross Link Detection Protocol (CLDP) [20, 47] probes the crossing links before removal, which can avoid the network partition problem and facilitate face routing on any 2D topologies. However, this planarization method sends probes to traverse each face and potentially even the entire network, which is not a localized operation. Even with a planarized topology, it may not be easy to choose a good face traversal direction and doing face change correctly is either expensive or hard. Combined with temporal variations in the network links, these difficulties will render face routing a less practical option for a real deployment.

2.2.3 Tree-based Routing

To support indoor sensor network applications, many network deployments naturally grow from two-dimensional to three-dimensional [16, 17], as the nodes are placed at multiple levels of a building. It is crucial to ensure the correctness and efficiency of routing protocols in such 3D environments. Due to the requirement for planar graphs, face routing algorithms generally cannot be applied to 3D networks. The tree-based routing protocols provide a more promising alternative, as the correctness of the forwarding operation is not affected by the dimensionality of the network space.

The tree-based routing protocols build a spanning tree in the network and store the aggregated location information for the subtrees at the parent nodes in each level. A packet is forwarded towards the subtree containing the destination address. When none of the child nodes contains a subtree covering the destination location, the packet will be forwarded towards the root. If the destination node is not found after traversing all the possible subtrees, it means the destination node is disconnected and the packet will be dropped.

Newsome et al. proposed the Virtual Polar Coordinate Routing protocol (VPCR) [21] that assigns polar coordinates to the nodes. VPCR assumes that all nodes can obtain their geographic coordinates from GPS devices or localization methods and the spanning tree is built in a balanced way such that nodes close to one another will become siblings. The parent nodes will collect the subtree size from each child and report this information towards the root. The polar coordinates of a node N are represented by an angle range, proportional to the size of the subtree rooted at N. The packet forwarding direction is determined by the relation between the polar coordinates of the neighbors and the destination. The routing efficiency of VPCR is provided by the shortcut links across the sibling nodes. However, due to localization errors and the link selection procedure, the spanning tree can be highly skewed with few

shortcut links to exploit, leading to routes $50\% \sim 80\%$ longer than optimal.

Instead of using polar coordinates, the GDSTR protocol [22] works on the geographic coordinate system. Each parent node will construct a convex hull covering all nodes in the subtree. According to the destination's position, greedy forwarding is applied to minimize the Euclidean distance along the intermediate hops. When a local minimum node is encountered, GDSTR will resort to tree forwarding mode. As the convex hull is represented by polygons, two convex hulls may overlap with each other. All subtrees with convex hulls that may potentially contain the destination node will be traversed using a depth-first search algorithm.

Tree-based routing protocols are applicable to both 2D and 3D networks; however, the routing performance of tree-based protocols in 3D topologies has not be studied extensively. Zhou et al. proposed a 3D version of GDSTR (GDSTR-3D) [48] that uses two 2D convex hulls to approximate a 3D convex hull and applies greedy forwarding based on two-hop neighborhood information for better hop stretch performance in 3D networks.

Unlike GDSTR-3D, the Spherical Coordinate Routing protocol (SCR) presented in this thesis is a 3D variant of VPCR, which builds a spherical coordinate system based on the nodes' coordinates. The original VPCR protocol assumes the availability of the nodes' positions, while SCR applies a 3D version of NoGeo to derive the position information. In the greedy forwarding step, VPCR relies on the shortcut links between the sibling nodes to avoid routing packets through the root. We evaluated two alternatives for greedy forwarding in SCR. The first method is to use 3D virtual coordinates computed by the localization algorithm. Due to the localization errors, the greedy forwarding step fails constantly, leading to long routing paths. The second alternative directly uses the hop count vectors to the perimeter nodes for packet forwarding and achieves nearly 40% lower hop stretch than VPCR in both 2D and 3D network topologies.

2.2.4 Hop Count Vector-based Routing

The coordinate systems used by tree-based point-to-point routing protocols can be expensive to update in terms of computational overhead during network dynamics. The hop count vector-based routing protocols use the hop distances to a set of beacons to address the nodes, which are easier to update and incur lower maintenance cost during fluctuations of network connectivities.

In hop count vector-based routing, some nodes are selected as *beacons* (also called *landmarks* or *anchors*), while other nodes will measure the hop distance to each beacon and record these distances in the hop count vectors, which represent high-dimensional node coordinates in the form of singleton Lipschitz embedding [49]. Greedy forwarding is applied based on the inter-node distance computed from the hop count vectors. Some hop count vector-based routing protocols apply fall back mechanisms to complement the greedy forwarding method, in order to improve the delivery ratio. The differences of various hop count vector based routing protocols lie in the beacon selection method, the distance function and the fall back mechanism.

The Logical Coordinate Routing protocol (LCR) [23] employs an iterative voting approach to select candidate beacons based on its distance to the existing beacons. The candidate with the largest distance is selected at each round. It uses the L^2 (L^4 for 3D) norm Euclidean distance function to determine the

next hop. The recovery step of LCR requires each node to remember the packet it has forwarded recently and the list of neighbors selected as the next hop for that packet. When the memory is not constrained, the packet will be able to reach any connected node by using a depth first search method.

The Hop-ID based Routing protocol (HIR) [32] proposed a centralized landmark selection algorithm. HIR constructs a shortest path tree from the controller node and subtree nodes will report their node IDs to the controller through the upstream nodes. After all node IDs have been collected, the controller randomly selects the required number of landmarks and informs the candidates through a message broadcast on the tree. Based on empirical results, HIR uses L^{10} norm of the Euclidean distance for greedy forwarding. To resolve the dead-end problem, HIR uses both a landmark-guided routing method and scoped flooding. Landmark-guided routing forwards the packet towards the landmark with the minimum hop distance to the destination. When a node closer to the destination than the local minimum node is discovered, HIR will switch back to greedy forwarding. Scoped flooding is used for the remaining cases that cannot be resolved by landmark-guided routing.

The Beacon Vector Routing protocol (BVR) [24] uses a random beacon selection method. However, instead of using the Euclidean distance function, it uses a weighted Manhattan distance function for greedy forwarding. It gives higher preference to the beacons closer to the destination than the current hop. The packet is forwarded towards the high-priority beacons and away from the low-priority ones. The recovery procedure of BVR is similar to that of HIR, which allows the packet to fall back to the beacon closest to the destination and uses scoped flooding as the last resort. Simulation results show that BVR can achieve a delivery ratio over 95% with 10 routing beacons in large networks (over 800 nodes), when 2% of the nodes are candidate beacons. In the experiments, it employed $5\% \sim 10\%$ of the nodes as beacons for medium-sized networks with $40 \sim 70$ nodes.

The packet forwarding procedure of the Small State and Small Stretch routing protocol (S4) [50] combines a localized version of DSDV with the fall back mechanism used in HIR and BVR. Once the hop count vector is built, every node will extract the hop distance d to the closest beacon and apply incremental updates to create a cluster with a diameter of d hops. All neighbors within this cluster, including the closest beacon, will add a routing entry for the cluster head, such that the shortest path can be taken to route packets within the cluster. For inter-cluster routing, S4 will send the packet towards the beacon closest to the destination, until the packet enters the cluster of the destination. To achieve a balance between performance (in terms of delivery ratio and hop stretch) and memory cost, S4 requires $\sqrt{N}$ beacons, where N is the total number of nodes.

The placement of beacons is crucial for hop count vector-based routing. Clustered beacons with low path diversity will generate highly correlated hop count distances and become less effective to differentiate nodes at different locations. The beacon selection problem has been studied in the context of network distance estimation [51]. Zhang et al. [52] proposed a hierarchical beacon selection approach that groups the candidates into clusters. Beacon nodes are selected from both nearby and distant clusters, in order to improve the granularity of the embedding and capture the global network connectivity. Srinivasan et al. [25] conducted a performance comparison of different beacon selection methods based on randomization, clustering, hierarchical structure and min/max inter-beacon distance. The result shows that the heuristics-based beacon selection algorithms provide nearly identical performance as random selection.

Given a network topology, it is difficult to find the optimal beacon placement and the critical number

of beacons required for the routing performance. For many algorithms, utilizing more beacons becomes a common approach to achieve higher delivery ratio and lower hop stretch. As the intrinsic dimensionality of the network deployment is low (2D or 3D), the high-dimensional hop count vectors caused by a large number of beacons will contain significant amount of redundancies. The existing hop count vector-based routing protocols can apply heuristics to avoid some less efficient beacons; however, they cannot fundamentally eliminate such redundancy. In this thesis, we apply the Principal Component Analysis-based dimension reduction algorithm (PCA) to tackle this problem. The PCA algorithm can compress the high-dimensional coordinates into low-dimensional ones, while preserving the routing performance at lower packet overhead.

2.2.5 Randomized Algorithm

Flury et al. proposed the Random-Greedy-Random (GRG) algorithm [33] for routing packets in 3D networks [53, 54]. They proved that the path length of any memoryless, localized geographic routing protocols is bounded by $\Omega(r^3)$, where r is the shortest path length. The packet forwarding is performed on a dual graph filled with virtual cubes. Similar to the GOAFR+ protocol, the GRG protocol combines greedy forwarding with a bounded search algorithm which visits the dual vertices using random walk within an exponentially increasing area. The authors used simulation to show that GRG can be more efficient than pure flooding for large-scale networks. However, due to the potentially long path length and high delivery latency, it is not a practical choice for real network deployments.

2.3 Other Related Areas

To provide reliable point-to-point delivery, geographic routing protocols also require support from other services, such as network embedding, link quality estimation and location service.

2.3.1 Network Embedding

The network embedding algorithms derive virtual coordinates based on the network connectivity, such that the distance computed from the coordinates of two nodes can be used to approximate the actual distance in the network. One popular application of network embedding is delay estimation over the Internet, where measuring the latency between every pair of nodes is prohibitive due to the network scale. In [55], the distance between two nodes is estimated by iteratively minimizing a potential energy function. Mao et al. [56] applied matrix factorization to obtain an incoming and an outgoing vector for each node, such that the network distance between two nodes is the dot product of the two vectors. The network can be embedded in a hyperbolic or an Euclidean space and prior studies [57] have shown that they have nearly identical performance for network latency prediction.

Network embedding can also provide the address information for geographic routing, when the actual node coordinates are not available. The NoGeo [58] protocol selects the perimeter nodes from the network boundary and allows each node to compute and refine its coordinates through iterative updates with its neighbors. Both Vivaldi [59] and GSpring [60] treat the network as a spring system, where each link has a normalized length. By applying the contraction and expansion rules according to the node

distance, the network system converges to a stabilized state, where the node location can be determined independently.

Principal Component Analysis (PCA) [61] is a common technique used in data analysis [62] and image processing [63], where the PCA-based dimension reduction algorithms can remove the information redundancies by transforming the high-dimensional data into a low dimensional space. The dimension reduction procedure can be treated an alternate method for network embedding, apart from the iterative approach seen in NoGeo and Vivaldi. ICS [64] and Virtual Landmarks [65] are two PCA-based algorithms that embed each node based on the delay to a set of landmarks and estimate the mutual network latency from the derived coordinates.

While the network latency may violate the triangle inequality property [66] assumed by the Euclidean embedding methods, the geodesic distance measured in hop counts generally conform to the triangle inequality and symmetry properties. Instead of predicting the network latency, in this thesis, we apply the PCA-based dimension reduction technique to hop count vector-based routing in resource-constrained sensor networks. The resulted virtual coordinates can retain the network connectivity with lower overhead, while maintaining the routing efficiency. Our approach is practical in sensor networks and we are able to implement a distributed version of the PCA algorithm using TinyOS-2. The performance result collected from a large-scale testbed shows that, using the same amount of packet overhead, the coordinates generated from the PCA algorithm can achieve higher delivery ratio and lower path length than the unprocessed hop count vectors.

2.3.2 Link Quality Estimation

Empirical studies [46] have shown that the wireless link connectivity is not a binary notion but a likelihood of successful transmission. The communication area around a wireless node contains a *transitional region* (also known as *grey area* [67]), where link asymmetries are more prominent [68] and link quality fluctuates with high variance. Without considering the varying link qualities, multi-hop packet forwarding will experience poor performance. A reliable routing protocol should incorporate a link quality metric adaptive to changes in the connectivities.

The hop count metric is not suitable for wireless networks. Minimizing the hop count will maximize the per hop distance, which will cause a reduced signal strength at the receiver and increase the packet loss ratio. Couto et al. proposed a metric named Expected Transmission Count (ETX) [69] for path selection. The ETX metric measures bi-directional packet delivery ratios of a link and computes the average number of transmissions required for each delivery. For network deployments with link quality variations, ETX can significantly improve the delivery success rate over multi-hop paths than the hop count metric.

Cerpa et al. studied the temporal properties [70] of the wireless links with MICA-1 and MICA-2 nodes. They found that the packet losses did not always follow random uniform distribution. They proposed the Required Number of Packets (RNP) metric that takes into account not only the packet delivery ratio but also the distribution of packet losses. Given two links with equal packet delivery ratios, the RNP metric will penalize the link with more consecutive packet losses, as the link with continuous losses will fail more retry attempts when link layer retransmission is enabled.

The link layer usually allows only limited number of retries, which is often neglected by link quality metrics in order to simplify the computation of point-to-point cost. Jakllari et al. proposed a routing metric named ETOP [71] to take this into account. ETOP considers not only the number of links and link quality along the path, but also the link position. The poor links close to the destination are considered to be more harmful than the poor links close to the sender, as transmissions at all previous hops must be redone given a limited number of retries in the link layer. Experiments on a 25 nodes mesh network show that ETOP can achieve higher the TCP throughput than ETX with a longer average path length. However, finding a path with the minimum ETOP requires the complete connectivity graph, which makes it more complex to compute than ETX.

Srinivasan et al. [72] evaluated two link quality estimators provided by the CC2420 radio chip: RSSI and LQI [6]. RSSI refers to the received signal strength and LQI indicates the bit error rate when decoding the first 8 symbols of the received packet. The LQI value over a link shows a large variation; therefore, the *average* LQI over a window of at least 120 packets is a more accurate estimator for the packet reception rate. However, due to the requirement for a large number of samples, the average LQI may be slow to adapt to link quality changes. RSSI has a small variation and the RSSI value obtained from a single packet can reflect the network condition. Hence, RSSI can serve as a coarse indicator to tell if the link quality is good (RSSI > -87 dBm) or bad. As the experiments were conducted on the CC2420 radio, their results cannot be simply generalized to other radio platforms.

Gnawali et al. [73] studied the effects of link layer retransmission, blacklisting and ETX on the Directed Diffusion routing protocol in a network with MICA-2 motes. The result shows that link layer retransmission can greatly improve the packet delivery ratio even with a small number (e.g. 3) of retries. Blacklisting is effective for high node density networks, but it is not stable for low density networks due to the fact that discarding the low quality links may cause network partition. Combined with link layer retransmission, ETX is a very reliable metric for multi-hop packet forwarding.

Liu et al. [74] examined the performance of the Collection Tree Protocol with different link quality metrics, including ETX, RNP and 4B [75]. Compared to RNP and 4B, ETX achieved very good performance in terms of packet delivery ratio, path quality, path length and total number of transmissions. However, ETX had higher number of route changes compared to 4B and RNP. The route changes triggered more frequent exchange of beacons but also alleviated the hop spot problem. The blacklisting strategy did not exhibit significant effects on ETX and RNP, while it improved the performance of 4B.

In the design of our routing protocols, we assume the link losses are handled by the link layer protocols. For our implementation, we used periodic beacons to estimate the packet reception ratio and employed ETX as the cost metric to construct the routing topology, since ETX is very reliable in various network settings [73, 74] and can be computed efficiently.

2.3.3 Location Service

The Grid Location Service (GLS) [76] divides the network into grids and covers the network with a hierarchy of squares containing an increasing number of grids. Each node N forwards its location update message to the squares at every level. The node with the minimum ID greater than N in a square will become a location server for N. The location look-up procedure is performed by following a chain of

nodes with a ID larger than the target, until the query reaches the target node. The reply is forwarded directly from the target node back to the source. By allowing the query packets to follow different paths, GLS can distribute the workload of address look-up across the network and improve resilience to node failures.

The Geographic Hash Table (GHT) system [8] originally designed for data-centric storage can be employed for location service as well. GHT hashes keys into geographic coordinates and stores the key-value pair in the node closest to the hashed position. For location service, the key can be set to the unique node ID and the value is the node's location. A source node S uses geographic routing to send its location to a node (the location server for S) residing on the hashed position. The address query is forwarded in a similar manner, until it reaches the location server, which will send back the result. GHT also allows the location information to be replicated at nodes near the server, such that the information will remain available after the original server fails. With a distributed design, GHT imposes lower storage requirement on the servers and alleviate the problem of single point of failure.

Ortiz et al. proposed the Beacon Location Service (BLS) [77] that works on top of BVR. Once the beacon vectors have been selected, each node will use a hash function to choose one of the beacons as its location server. The node address can be uploaded to the server by following the spanning tree. The memory cost required to store the ID-address mappings will be shared by the beacons. The node that initiates a query will apply the same hash function to the target node ID and obtain the location server's ID. The query is forwarded on the spanning tree and the reply can be sent back by BVR. BLS allows intermediate nodes to eavesdrop the address update packets in order to alleviate the congested path problem caused by the look-up traffic.

As the focus of our work is on routing protocols, the routing algorithms presented in the following chapters assume that a reliable location service is available. For the experiments performed on the Indriya testbed, we implemented a centralized location service using the VPCR protocol to route the address queries and replies.

Chapter 3

Spherical Coordinate Routing for 3D Networks

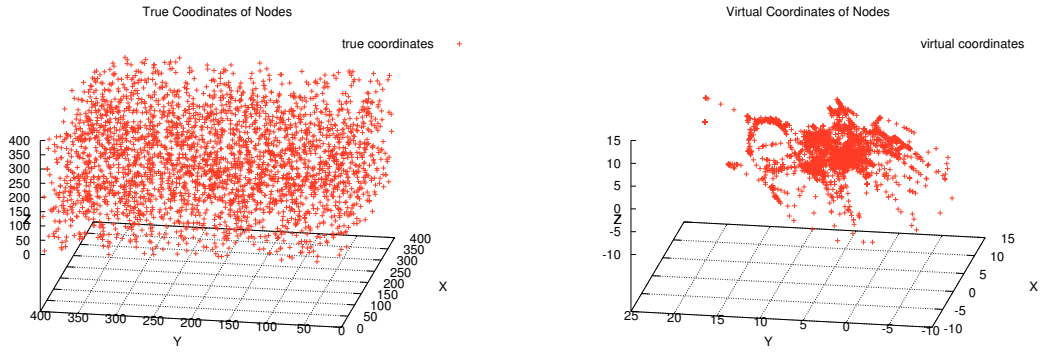
3.1 Introduction

The deployment of wireless nodes is often conducted in a three-dimensional space; however, few routing protocols have focused on providing efficient data delivery in a 3D environment.

Traditional ad-hoc routing requires each node to perform path discovery and maintain separate routing entries for different destinations, which is not scalable for a large network. Geographic face routing protocols [41] are proposed to provide guaranteed delivery by routing the packets around voids; however, their two-dimensional planarization procedure is expensive and not applicable for three-dimensional networks. The connectivity-based routing protocols such as Beacon Vector Routing (BVR) [24] and Logical Coordinate Routing (LCR) [31] provide alternatives for routing on a network with higher dimensions. Local minimum failures may occur in these protocols because of localization errors or insufficient resolution in the coordinates. A flooding-based recovery procedure can be applied; however, it may also incur high network overhead.

Inspired by NoGeo [58], Virtual Polar Coordinate Routing (VPCR) [21] and BVR, we introduce the Spherical Coordinate Routing (SCR) protocol that guarantees packet delivery in the three-dimensional space without flooding the data packets. SCR adopts the NoGeo's approach for localization and extends the method from 2D to 3D. In terms of routing, it integrates connectivity-based greedy forwarding to achieve routing efficiency and a 3D extension of VPCR's tree-based routing procedure to ensure the packet delivery. By complementing greedy forwarding with tree traversal, SCR substantially improves the packet delivery ratio with low communication overhead.

The rest of the chapter is organized as follows. In Section 3.2, the network localization, spherical coordinates assignment and the routing procedures of SCR are explained in detail. In Section 3.3, we analyze the control overhead of SCR and discuss the selection of routing metrics through empirical measurements. The performance comparison between SCR and other 3D-compatible geographic routing protocols is provided in Section 3.4. The summary is presented in Section 3.5.



(a) True Coordinates

(b) Virtual Coordinates

Figure 3.1: 3D localization of 3000 nodes with 100 iterations. density: 5.3; distance unit: (a) meter, (b) hop count.

3.2 Protocol Design

The SCR protocol assigns spherical coordinates to each node based on the node position estimated from the localization step. The spherical coordinates consist of the hop count to the root of the recovery tree, the projected angle θ on the XY plane and the projected angle φ on the YZ plane as shown in Fig. 3.2. In this section, we present the details of the localization method, the spherical coordinates assignment procedure and the routing algorithm.

3.2.1 Localization Algorithm in 3D Networks

The SCR protocol applies the trilateration method for network localization with four anchor nodes in the three-dimensional space. A 3D expansion to NoGeo [58] is employed, which randomly selects a bootstrap node to broadcast an initial *bootstrap* message to the whole network. Upon receiving the bootstrap message, each node checks if it is the furthermost node away from the bootstrap node within a 2-hop neighborhood. If a node is the 2-hop local maximum, it declares to be a *perimeter node* by broadcasting a *hello* message to the network after a random delay. Each perimeter node will suppress other candidates within an t-hop ($t \in [5, 8]$) neighborhood. Once all perimeter nodes have declared their identities, each node in the network will have a hop count vector indicating the distance to the perimeter group. Following that, all perimeter nodes will exchange their hop count vectors, such that every perimeter node will know the mutual hop distance between one another. The four anchor nodes are selected from the perimeter group.

The first three anchor nodes are selected in the way that the triangle formed by the three nodes covers the maximum area. If node A, B and C are the three initial anchors and the hop count distances are $dist(A, B) = a$, $dist(B, C) = b$ and $dist(A, C) = c$, the triangle area computed by Formula 3.1(Heron's Formula) is the maximum among all perimeter nodes.

$$Area = \sqrt{S(S-a)(S-b)(S-c)}, S = \frac{a+b+c}{2} \tag{3.1}$$

Once the three anchor nodes are identified, the relative position of other nodes can be calculated from their hop count distance to these references. The fourth anchor node is selected such that volume of the tetrahedron (V) formed by the four anchors, $A_i(x_i, y_i, z_i), i \in [1, 4]$, has the maximum volume as computed by Formula 3.2, where *det* represents the determinant value of a matrix.

$$V = \frac{1}{6} \times det \begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{bmatrix} \tag{3.2}$$

After the coordinates of the anchor nodes are determined, the rest of the nodes will extract the distance vector [h_1,h_2,h_3,h_4] to the four anchors. The node location (x, y, z) can be estimated by minimizing the potential energy function presented in Formula 3.3 through a simplex method [78], where $dist(A_i, N)$ denotes the Euclidean distance between anchor A_i and node N.

$$E = \sum_{i=1}^{4} [dist(A_i, N) - h_i]^2 \tag{3.3}$$

As each node in the network has also received the hop distance vectors of all the perimeter nodes, the above procedures can be performed independently by each node. We calculate the center of gravity C for all the perimeter nodes and identify 3 perimeter nodes with the lowest IDs, denoted as P_1, P_2 and P_3. We let CP_1, CP_2 and CP_3 represent the positive X^+, Y^+ and Z^+ axes in the virtual coordinate system and normalize each node position according to the new axes. We denote the vector of $P1$ in the 3D space as $\overrightarrow{P_1} = (P_1.x, P_1.y, P_1.z)$, where $(P_1.x, P_1.y, P_1.z)$ represents the coordinates of P_1. For any normal node V, we obtain its corresponding 3D vector $\overrightarrow{V}$. The projection V_{P_i} of $\overrightarrow{V}$ on the 3 axes $\overrightarrow{P_1}$, $\overrightarrow{P_2}$ and $\overrightarrow{P_3}$ can be computed from the *dot product* of $\overrightarrow{V}$ and $\overrightarrow{P_i}(i \in \{1, 2, 3\})$, as given in Formula 3.4, where α is the angle between $\overrightarrow{V}$ and $\overrightarrow{P_i}$ and V_{P_i} is the project of $\overrightarrow{V}$ on $\overrightarrow{P_i}$ and $i = 1, 2, 3$.

$$\begin{aligned} \overrightarrow{V} \cdot \overrightarrow{P_i} &= \|\overrightarrow{V}\| \|\overrightarrow{P_i}\| \cos\alpha \\ \Rightarrow V_{P_i} &= \|\overrightarrow{V}\| \cos\alpha = \frac{\overrightarrow{V} \cdot \overrightarrow{P_i}}{\|\overrightarrow{P_i}\|} \end{aligned} \tag{3.4}$$

After the normalization step, the coordinates of each node $(V.x, V.y, V.z)$ are transformed into the projected coordinates $(V_{P_1}, V_{P_2}, V_{P_3})$. Finally, we compute the average distance r from all perimeter nodes to C and project all perimeter nodes onto a sphere centered at C with radius r. In the position relaxation stage, the perimeter nodes remain static, while the normal nodes iteratively take the average neighbor location as the new location at each interval. The refinement procedure is repeated 100 times based on empirical results. A localization example of 3000 nodes is shown in Fig. 3.1.

3.2.2 Spherical Coordinates Assignment

In order to assign the spherical coordinates, a spanning tree is constructed from a designated root node. In a bottom-up manner, the child nodes at each level report the subtree sizes to the parent nodes until the

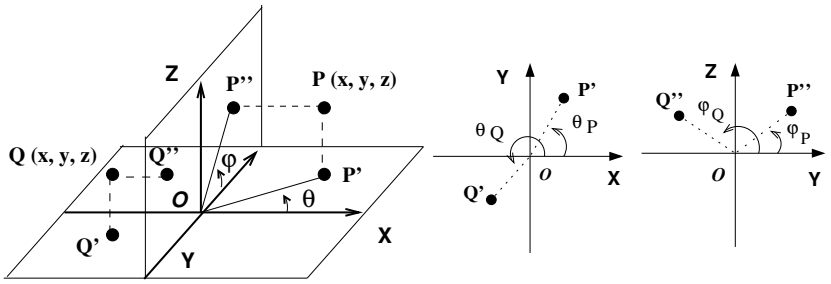


Figure 3.2: Spherical Coordinate Angles: θ and φ

subtree size information propagates back to the root. The spherical coordinates are then assigned to the child nodes level by level according to their alignment in the tree and their virtual coordinates computed by the localization method. Assume a node O has k children N_1, N_2, ..., and N_k whose subtree sizes are s_1, s_2, ..., and s_k. The initial angle ranges for node O are θ range $[\theta_1, \theta_2]$ and φ range $[\varphi_1, \varphi_2]$. From the virtual coordinates of N_1, N_2, ... and N_k, we can retrieve the order of their projections on the XY plane and the YZ plane in the counter-clockwise direction. For example, in Fig. 3.2, the projection order of child nodes P and Q on the XY plane and YZ plane are $[P', Q']$ and $[P'', Q'']$. Based on the projection order on the XY plane, the θ range of the child node N_i (i is the index in the projection order) can be computed by Formula 3.5. The formula can be applied to calculate the φ range similarly.

$$
\begin{aligned}
N_i.\theta_1 &= \frac{\sum_{j=1}^{i-1} s_j}{\sum_{j=1}^{k} s_j} \times (O.\theta_2 - O.\theta_1) + O.\theta_1 \\
N_i.\theta_2 &= N_i.\theta_1 + \frac{s_i}{\sum_{j=1}^{k} s_j} \times (O.\theta_2 - O.\theta_1)
\end{aligned} \tag{3.5}
$$

Once the spherical coordinates are assigned, each node has a hop count value h and two angle ranges, denoted as θ range $[\theta_1, \theta_2]$ and φ range $[\varphi_1, \varphi_2]$. Hop count h represents the distance to the root and the θ and φ are *subsets* of the angle ranges in the parent node. An example is shown in Fig. 3.3, where node A has a hop count of $h = 6$. The θ range of A is $[100, 240]$ and its φ range is $[50, 330]$. Node A has two child nodes B and C, whose subtree sizes are $s_b = 4$ and $s_c = 3$. Assuming the projection orders of B and C on XY plane and YZ plane are both $\{B, C\}$, the θ range of B is calculated as $B.\theta_1 = A.\theta_1 = 100$, $B.\theta_2 = \frac{s_b}{s_b+s_c} \times (A.\theta_2 - A.\theta_1) + A.\theta_1 = \frac{4}{7} \times (240 - 100) + 100 = 180$. The φ range can be calculated similarly; thus, the spherical coordinates of B is $\{7, [100, 180], [50, 210]\}$.

The relation between the θ(or φ) angle range of two nodes A and B can be classified as one of the following:

1. *Contained*: if $B.\theta_1 \in [A.\theta_1, A.\theta_2]$ and $B.\theta_2 \in [A.\theta_1, A.\theta_2]$, or vice versa
2. *Disjoint*: if $A.\theta_1 > B.\theta_2$ or $B.\theta_1 > A.\theta_2$
3. *Overlapped*: if $A.\theta_1 \in [B.\theta_1, B.\theta_2]$ and $A.\theta_2 > B.\theta_2$, or vice versa

By performing angle range assignment in the tree structure, each child receives a unique segment from the parent's angular space; thus, no two nodes should have *overlapped* angle ranges. The angle range

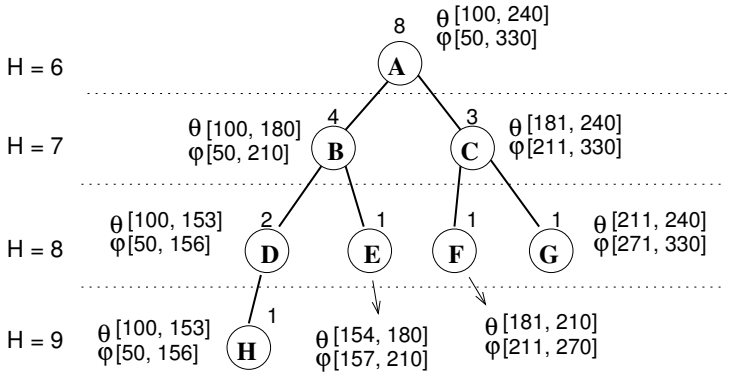


Figure 3.3: Example of Angle Range Assignment

relation of two nodes in this spherical coordinate tree is either *Contained* or *Disjoint*. We denote the angle range relation between two nodes N_1 and N_2 as $Rel(N_1, N_2)$. If $Rel(N_1, N_2)$ = $Contained$, it means the angle range of one node is contained in the other. Assuming the angle range of N_2 is a subset of N_1, node N_2 must be a descendant of N_1 in the tree. If $Rel(N_1, N_2)$ = $Disjoint$, they share some common ancestors.

The distance between two nodes A and B are calculated differently according to their angle range relation. Assuming the spherical coordinates of $A(B)$ are $\{h_A(h_B), [\theta_1, \theta_2], [\varphi_1, \varphi_2]\}$, the node distance $dist(A, B)$ is computed by Formula 3.6, where $dist_\theta = max(A.\theta_1 - B.\theta_2, B.\theta_1 - A.\theta_2)$ and $dist_\varphi = max(A.\varphi_1 - B.\varphi_2, B.\varphi_1 - A.\varphi_2)$.

$$dist(A, B) = \begin{cases} |h_A - h_B|, & \text{if A,B } \textit{Contained} \\ dist_\theta + dist_\varphi, & \text{if A,B } \textit{Disjoint} \\ -1, & \text{otherwise} \end{cases} \tag{3.6}$$

Note that when distance computed is -1, localization error has occurred and the value will be discarded. In general, if the angle range relation of two nodes A and B is *Contained*, the distance between A and B is equal to the difference between their hop counts to the root. If the relation is *Disjoint*, the distance is equal to the gap enclosed by the angle ranges. For example, in Fig. 3.3, $dist(A, D)$ is 2, because $Rel(A, D)$ = $Contained$ and $dist(A, D) = |h_A - h_D| = |6 - 8| = 2$. For node E and G, since $Rel(E, G)$ = $Disjoint$, we have $dist(E, G) = (G.\theta_1 - E.\theta_2) + (G.\varphi_1 - E.\varphi_2) = 31 + 61 = 92$.

3.2.3 Routing Algorithm

Basic routing using angle-based greedy forwarding

Suppose that a source node S has a packet p to destination T and node S has k neighbors $N_i, i \in [1, k]$. If there is a neighbor N_i, such that the angle range relation $Rel(N_i, T)$ is *Contained*, we call N_i a *CT* neighbor. If $Rel(N_i, T)$ is *Disjoint*, we call N_i an *AR* neighbor. Depending on the angle range relation between the forwarding nodes and the destination, routing can be performed in *CT* mode, *AR* mode or *BACK* mode.

If a *CT* neighbor is available, node S will set the routing mode to *CT* mode. If there is no *CT*

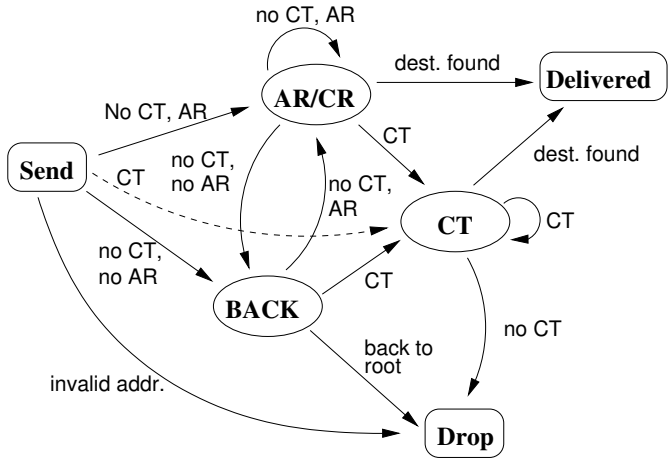


Figure 3.4: Routing State Transition Diagram

neighbor, but an *AR* neighbor with a closer angle range distance to the destination, the routing mode will be set to *AR* mode. If neither a *CT* neighbor or a closer *AR* neighbor is found, node A will use the *BACK* routing mode and select its parent in the tree as the next hop. The routing state transition diagram is depicted in Fig. 3.4, indicating that a *CT* neighbor is preferred over an AR neighbor. This is because a *CT* neighbor is either an ancestor or descendant of the destination, from where the destination can be reached by following the tree branches directly. The *AR* neighbors may provide a shortcut path to route packets along the sibling connections, while the *BACK* mode will pass the packet to upper levels of the tree if both *CT* and *AR* modes fail. As the nodes at the top of the tree consist of the ancestors of all subtree nodes in the lower levels, during the backtracking stage we can ultimately find the ancestor nodes of the destination, from where the packet can be forwarded down the tree towards the target again.

The data packet header carries the following fields: the hop count h between the destination and the root, the θ and φ angle ranges of the destination, the current routing mode M, the current minimum *AR* distance d_{ar} and the minimum *CT* distance d_{ct}. The detailed routing algorithm is illustrated in Algorithm. 2. For example, node E in Fig. 3.3 has a packet for node G, and E has three neighbors B, D and F. Since the angle ranges of B, D and F are all disjoint from G, no *CT* neighbor can be found and the routing mode is set to *AR*. The angular distance from E to G is $dist(E,G) = dist_\theta(E,G) + dist_\varphi(E,G) = 31 + 61 = 92$ and we have $dist(B,G) = 92$, $dist(F,G) = 2$, $dist(D,G) = 167$. From the angular distance, node F is a closer *AR* neighbor and will be selected as the next hop in *AR* mode. If node F is not a neighbor of E, both *CT* and *AR* modes will fail and E will switch to *BACK* mode and pass the packet to its parent node B. Node B will remain in the *BACK* mode and continue to pass the packet to node A. As the angle range of A is a superset of G's angle range, the routing mode will switch to *CT* at node A and the packet will go directly from A to C and reach G by comparing the hop count distance.

Routing using connectivity-based greedy forwarding

The angle-based greedy forwarding (also used in VPCR) does not perform well due to the difficulty in assigning angle ranges consecutively across the adjacent sibling nodes. SCR replaces the basic angle-

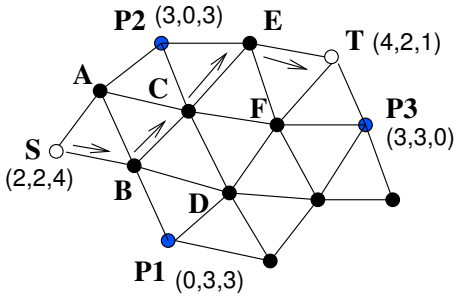
Algorithm 2 Routing Algo. with Spherical Coordinates

Let p be the packet and T be the current hop
The k neighbors of node T are $N_i, i \in [1, k]$.
$\forall N \in N_i$, compute the minimum AR distance to $p.dst$ denoted as $dist_{ar}$ and the minimum CT distance denoted as $dist_{ct}$.
The corresponding neighbors are N_{ct} and N_{ar}
if $p.M = CT$ **then** ▷ routing mode is *CT*
  if $0 \leq dist_{ct} \leq p.d_{ct}$ **then**
    $p.d_{ct} = dist_{ct}$, nexthop = N_{ct}
  else
    drop the packet p
  end if
else if $p.M = AR$ **then** ▷ routing mode is *AR*
  if $dist_{ct} \geq 0$ **then**
    $p.M = CT, p.d_{ct} = dist_{ct}$, nexthop = N_{ct}
  else if $0 \leq dist_{ar} \leq p.d_{ar}$ **then**
    $p.d_{ar} = dist_{ar}$, nexthop = N_{ar}
  else
    $p.M = BACK$, nexthop = *parent.*
  end if
else ▷ routing mode is *BACK*
  if $dist_{ct} \geq 0$ **then**
    $p.M = CT, p.d_{ct} = dist_{ct}$, nexthop = N_{ct}
  else if $0 \leq dist_{ar} \leq p.d_{ar}$ **then**
    $p.M = AR, p.d_{ar} = dist_{ar}$, nexthop = N_{ar}
  else
    nexthop = $parent$
  end if
end if

based greedy forwarding (*AR*) with connectivity-based greedy forwarding(*CR*) based on the Euclidean distance computed from the hop count vectors to the perimeter nodes. For a set of p perimeter nodes A_1, A_2, ..., A_p used in the localization step, each node in the network will maintain a hop count vector indicating the distance to each perimeter node $A_i (i \in [1, p])$. For two nodes M and N, we denote their hop count vector as $[m_1, m_2, \ldots, m_p]$ and $[n_1, n_2, \ldots, n_p]$. The distance D_{hop} between M and N can be computed as $D_{hop}(M, N) = \sqrt{\sum_{i=1}^{p} (m_i - n_i)^2}$. To send a packet to destination T, node P will search among all its k neighbors $N_i (i \in [1, k])$, such that the neighbor N' with $D_{hop}(N', T) = min\{D_{hop}(N_i, T)\}$ and $D_{hop}(N', T) < D_{hop}(P, T)$ is chosen as the next hop. The example network shown in Fig. 3.5 has 3 perimeter nodes P_1, P_2 and P_3. The distance between node $S(2, 2, 4)$ and node $T(4, 2, 1)$ is $dist(S, T) = \sqrt{(4-2)^2 + (2-2)^2 + (1-4)^2} = \sqrt{13}$. By using the Euclidean distance metric, the greedy forwarding path from S to T can be determined. If a local minimum is encountered, the routing procedure will switch to *BACK* mode and the tree-based recovery method will be applied until the greedy mode or *CT* mode can be activated again. We have integrated the connectivity-based greedy forwarding into SCR and find that it can provide improved routing performance highly resilient to localization errors. The results in this chapter are obtained from this improved version.

3.2.4 Tree Reparation for Network Dynamics

The tree reparation procedure is designed to cope with network changes, such as the introduction of new nodes and node failures due to power outage or physical damage. When a new node joins the

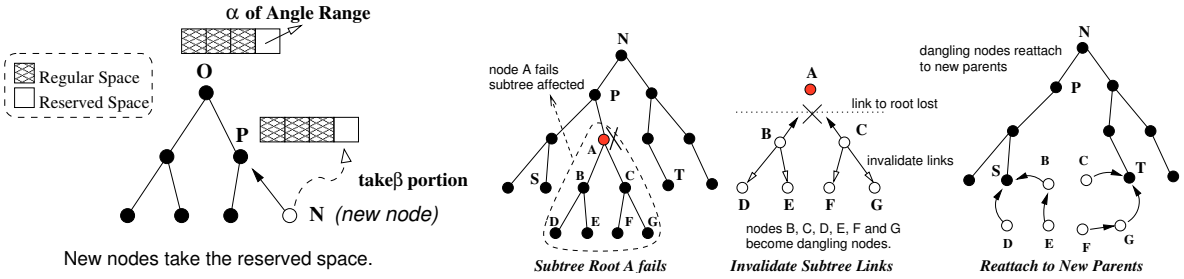


Hopcount vector and distance to node T:

A:(2,1,3), dist $= \sqrt{9}$
B:(1,2,3), dist $= \sqrt{8}$
C:(2,1,2), dist $= \sqrt{6}$
D:(1,2,2), dist $= \sqrt{10}$
E:(3,1,2), dist $= \sqrt{3}$
F:(2,2,1), dist $= \sqrt{4}$

$dist(S,T) = \sqrt{13}$, Path: $S \rightarrow B \rightarrow C \rightarrow E \rightarrow T$

Figure 3.5: Connectivity-based Greedy Forwarding



(a) Node Insertion (b) Node Failure

Figure 3.6: Tree Recovery for Node Insertion and Node Failure

network, it will attach to a parent node whose angular coordinates have already been configured. In order to accommodate the new child, the parent node will reclaim part of the angle range from its current children. The angle range reclamation has to be performed recursively down to the leaf level. An alternate option is that during network initialization, each node reserves part of the assigned angle space to handle future node insertions as illustrated in Fig. 3.6(a). For a node P with θ angle range $[\theta_1, \theta_2]$, before assigning angle range to the subtree nodes, it will reserve α portion of the angular space (θ_2 - θ_1). After the network setup is completed, it has $\alpha(\theta_2$ - $\theta_1)$ angle range available. The φ angle range can be managed accordingly. For each new child node N attaching to P, node P will assign β part of the remaining reserved angle range to N. The values of α and β control the shape of the recovered tree branches. For infrequent node insertions, we set the $\alpha = \beta = 0.25$.

A node failure will render the subtree rooted from the failed node invalid. If the parent beacons are not received after a threshold interval, the child nodes will declare that the parent is gone and send *invalidate* messages to their descendants. This *invalidate* messages will be disseminated downwards through all the subtree links. The descendant nodes receiving the *invalidate* message will discard the current spherical coordinates and become dangling nodes. As shown in Fig. 3.6(b), the dangling nodes will try to re-attach to a new parent by overhearing the beacons, which contain the neighbor's ID, the hop count distance to the root and the angle range information. By sending re-attach requests, a dangling node can be adopted by a neighbor with valid coordinates. Frequent node insertions and node removals will cause the angle range to be acquired and reclaimed repeatedly, creating a large number of non-contiguous angle fragments, which are hard to manage. The fragmentation problem can be alleviated by reconstructing the subtree, triggered once the number of fragments at a node grows above the threshold.

If the root area of the backbone tree fails, the tree topology will become logically disconnected. In such a case, nodes detecting the failure must invalidate the obsolete tree structure and initiate a new root election process. The tree reconstruction procedure will be performed as described in Section 3.2.2. To alleviate the single point of failure problem and hot spot effects of using a single recovery tree, multiple trees can be deployed, whose roots are scattered around the whole network. With multiple trees established, each node will have several routing options available for a single destination. Therefore, clustered node failures occurring at a single area will not break the entire routing infrastructure. For large-scale networks, the candidate roots can be selected by a randomization method. For a small-scale network deployment, the root nodes can be configured manually and placed around the network, such that the path diversity provided by these recovery trees can be maximized.

3.3 SCR Overhead and Routing Metrics

3.3.1 Control Overhead of SCR

The control overhead of SCR mainly comes from the localization method and the spherical coordinate assignment procedure. In the localization step, the bootstrap node broadcasts a beacon to the whole network of n nodes for discovering perimeter nodes. The beacon propagates hop by hop, which takes n transmissions. The nodes that have the local maximum hop count distance to the bootstrap node within a suppression range of $t = 5$ hops will become the perimeter nodes. The total number of perimeter nodes is bounded to the number of nodes required to cover the surface area of the 3D network deployment space. For networks with *moderate* densities, the number of perimeter nodes is bounded by $O(n^{\frac{2}{3}})$. For a *high* density network of n nodes deployed in a cubic space with side length of d, the estimated number of perimeter nodes is $\frac{6d^2}{\pi(5r)^2}$, where r is the node communication radius. For example, in a *dense* network with $d = 400$ and $r = 30$, the average number of perimeter nodes is $\frac{6\times400^2}{\pi(5\times30)^2} \approx 14$. The suppression range allows the selection procedure to scale as the network size increases. Each perimeter node will broadcast 3 times to the network for building the hop count vectors, exchanging the hop count vectors with other nodes and announcing the initial virtual coordinates. This step will cost $3n \times O(n^{\frac{2}{3}})$ transmissions. The iterative refinement for the coordinates can be done through regularly exchanged beacons, which will not generate extra packets. The completion time of the refinement step will depend on the number of rounds and the beacon interval.

The spherical coordinate assignment takes two rounds to complete, one round to collect the subtree size from each child and one round to inform the angle range assignment in a localized manner. The number of packets required is $2kn$, where k is the number of trees deployed. The total number of transmitted packets for the initialization step is equal to $n + 3n \times O(n^{\frac{2}{3}}) + 2kn$; each node averagely transmits $1 + 3 \times O(n^{\frac{2}{3}}) + 2k$ packets. For a *dense* network with 3 recovery trees, assuming $d = 400$ and $r = 30$, the number of control packets for each node during initialization is around $1+3\times14+2\times3 = 49$. The initialization overhead of SCR is equivalent to the sum of overhead in NoGeo and BVR. However, this procedure is performed only once and SCR employs mainly local recovery to counter small-scale network dynamics. The significant performance improvement of SCR in Section 3.4 will prove that this one-time initialization overhead is highly beneficial.

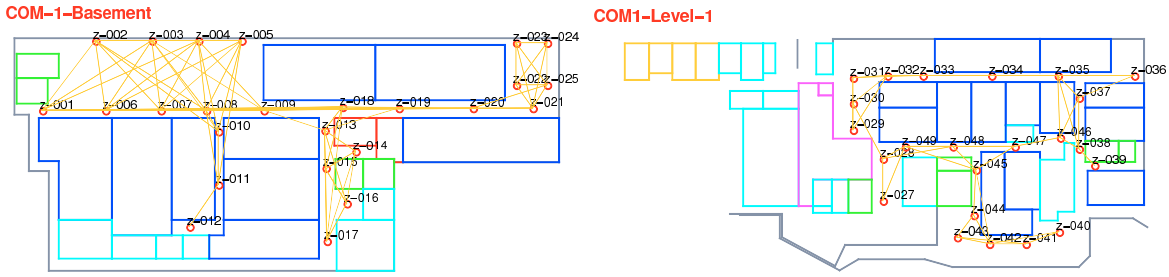
3.3.2 Storage Cost of the Key Components

The storage cost of SCR comes from four key components: the neighbor list, the hop count vector to the perimeter nodes, the matrix containing the inter-node distance among the perimeter nodes and the local spherical coordinates. Assume that both the node ID and the hop count values are of `uint16_t` type (2 bytes). The local spherical coordinates include the *hopcount* to the root, the φ angle range on the XY plane and the θ angle range on the YZ plane. As the *hopcount* is `uint16_t` and each angle range has two `float` (4 bytes) angle values, the local spherical coordinates requires 18 bytes. The neighbor list stores the information of the neighboring nodes within a range of one hop, including the node ID and the neighbors' spherical coordinates. Assuming that the average number of neighbors is k, the size of the neighbor list is $k \times (18 + 2) = 20k$ bytes. As SCR adopts the NoGeo's approach for node localization, for a set of p perimeter nodes, it will maintain a local hop count vector to these p perimeter nodes and a matrix indicating their mutual distance. The local hop count vector consists of the node ID of each perimeter node and the corresponding hop distance, consuming $4p$ bytes. The matrix has p rows, each of which contains the perimeter node ID and a list of p entries. Each entry consists of the destination perimeter node ID and the hop distance. Hence, the matrix consumes $2p + 4p^2$ bytes. The total amount of storage space for all the four components is $4p + 2p + 4p^2 + 20 + 20k$ bytes. Given a cubic network space with a side length of 400m and a communication range of 30m, a very dense network will have approximately $p = 14$ perimeter nodes and the network density can reach $k = 40$ [17]. Therefore, the resulted storage space is around 1688 bytes.

3.3.3 Selection of Routing Metrics

Many geographic routing protocols use *hop count* as the routing cost metric and shorter paths are considered to be superior to the longer ones. The hop count metric indicates the number of hops traversed by the packet. Under uniform transmission power settings, shorter routing paths can reduce energy consumption and delivery latency. However, previous experiments [69, 46] on outdoor wireless networks found that the shortest paths with long range links often provide worse routing performance, compared to longer paths with shorter per-hop distance. The long range links may have considerably lower reception quality due to various reasons, such as signal attenuation or multi-path fading. Some routing metrics (e.g. the number of transmissions and delivery latency) incorporate the packet delivery ratios of each link to improve the communication reliability. They mainly target on the outdoor wireless ad hoc networks, where long range links are possible with low obstruction in the open space and high-power radios (e.g. 17 dBm on Linksys WRT150N Router [79]). Meanwhile, for an indoor low power sensor network environment, the connected nodes usually lie within the line of sight. The radio used in the sensor nodes has much lower transmission power (e.g. ≤ 0 dBm on MICAz nodes [80]) for energy conservation. The long range links are blocked by the complex interior structures such as walls or furniture. This implies that the links in low power indoor sensor networks are more likely to have short ranges and good packet reception rates as observed in [81] and [82].

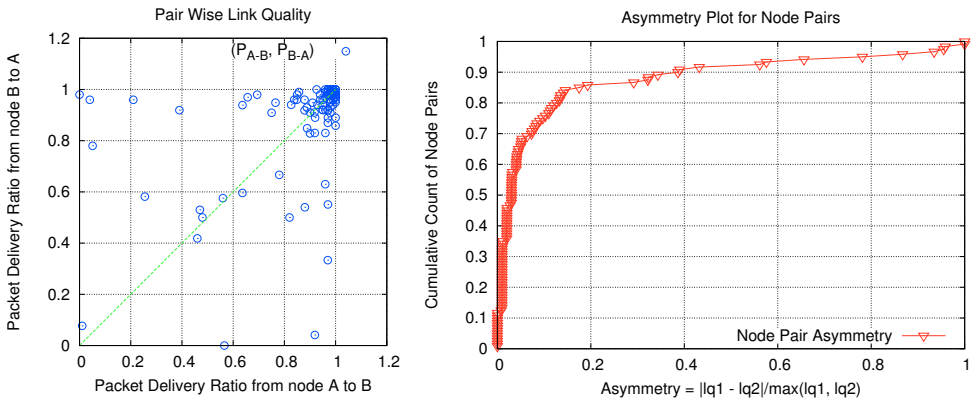
To verify this argument, we deployed 48 MICAz sensor nodes in two floors of the COM-1 office building at the National University of Singapore. The deployment map is given in Fig. 3.7. We measured the link connectivity and packet delivery ratios of each link in both directions p_1 and p_2 by using unicast



(a) COM-1, Basement, 25 nodes: z-001∼z-025 (b) COM-1, Level-1, 23 nodes: z-027∼ z-049

Figure 3.7: Link Connectivity of 48 MICAz Sensor Nodes Deployed at Two Floors of the COM-1 Building (cross-floor links not shown)

probing packets.



(a) Bi-directional Link Quality (b) Node Pair Asymmetry

Figure 3.8: Link Quality and Asymmetry Measurement Results (including cross-floor links)

The quality of the links are plotted in Fig. 3.8(a), which shows that 97 out of the 120 connected node pairs have delivery ratios from 80% to 100% in both directions. This is consistent with the connectivity graph in Fig. 3.7, where most links are short with no obstructions in between. Link asymmetry often occurs at the fading edge of connectivity [81], where the discrepancies in the node transmission power, reception sensitivity or energy levels may impose a significant impact on the packet reception ratio. We compute the level of link quality asymmetry as $\frac{|p_1-p_2|}{\max(p_1,p_2)}$ and plot it in Fig. 3.8(b). It can be observed that over 85% of the links have less than 20% asymmetry, implying that link asymmetry is not a critical issue for our indoor sensor network. This is because the CC2420 radio in MICAz nodes has low hardware miscalibration [72] and the maximum range of most links is constrained by the obstacles before it can expand to the limit. In summary, for scenarios where the majority of the network links exhibit equally good quality and low asymmetry, we believe *hop count* can still serve as a simple and efficient metric to evaluate the routing performance.

3.4 Performance Evaluation

The performance of SCR is compared to 5 routing protocols, namely: Greedy, NoGeo, BVR, LCR and VPCR. All algorithms are implemented in the network simulator *ns-2* and a Java simulator. We use the customized simulator because *ns-2* is not scalable for large network scenarios with over 1000 nodes. We used results generated from the network simulator *ns-2* to verify the output of the Java simulator and find that the Java simulator is highly consistent with *ns-2*. The results presented in this chapter are obtained from the Java simulator. As the original design of NoGeo cannot be applied directly, we extend the algorithm in NoGeo to make it compatible for 3D networks. The LCR protocol uses L^4 norm in the distance function and employs 8 landmarks for routing as suggested in [31]. The BVR protocol allows maximally 5% of the nodes to be the candidate beacons and the top 10 beacons closest to the destination are used for routing. For NoGeo, the third scenario is assumed where the perimeter node election is conducted in a 2-hop neighborhood and each perimeter node can suppress other candidates within 8 hops.

Table 3.1: Input of Simulation Parameters

Parameter	*Value*
Field Size	$400 \times 400 \times 400m^3$
Node Number	1000∼10000, step = 500
Radio Range	30m, unit disk
Topology	50 topos/density, uniform random
Connection	100 random pairs/topology
Routing Proto.	Greedy, NoGeo, BVR, LCR, VPCR, SCR

The configuration of the simulation parameters are listed in Table 3.1. Given n nodes with communication range r, randomly deployed in a cubic area of volume V, the average node density d can be computed as $d = \frac{4\pi r^3 n}{3V}$. The range of node densities thus varies from 1.77 to 17.67. For each node density, we generate 50 random network topologies. On each network topology, 100 node pairs are randomly selected to be the source and destination. We use the six different routing protocols to route data packets for each connection and compute the packet delivery ratio, hop stretch and average overhead per packet delivery for performance comparison. Finally, the performance of the protocols under clustered node failures are also investigated.

3.4.1 Packet Delivery Ratio

The packet delivery ratio shown in Fig. 3.9 indicates the amount of data packets delivered successfully by the primary routing methods of the protocols without flooding the network. The LCR protocol relies on 8 landmark nodes and tries to minimize the Euclidean distance to the destination during the packet forwarding. The BVR protocol relies on the weighted-Manhattan distance computed from the hop counts to the 10 routing beacons.

As expected, both VPCR and SCR are able to guarantee delivery through the tree structure once the network is sufficiently dense. The optimization in SCR allows it to perform slightly better than VPCR for node densities between 3 to 5. For the density range from 6.18 to 17.67, when the network is nearly fully

connected, the packet delivery ratio of VPCR and SCR converges to 100% as described in Section 3.2.3.

On the other hand, the greedy-based routing methods in NoGeo, BVR and LCR clearly fail to provide high delivery ratio at all node densities. At the highest node density of 17.67, the packet delivery ratio of LCR and BVR reaches 57.95% and 80.2%, while NoGeo has the lowest delivery ratio of 18.17%, because of the accumulated errors in the position relaxation steps. The fall back mechanism in BVR will try various number of routing beacons before giving up, which provides better packet delivery performance. In addition, due to the localization and quantization errors in the hop count-based coordinates, even with high node densities above 10, the packet delivery ratios for NoGeo, BVR and LCR are actually lower than Greedy.

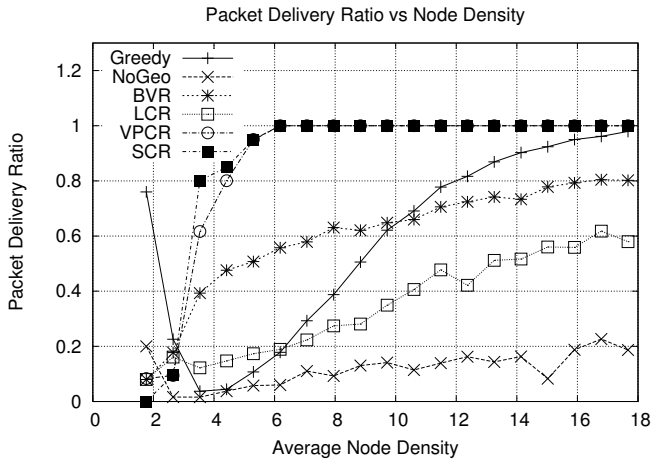


Figure 3.9: Packet Delivery Ratio

3.4.2 Hop Stretch Factor

Hop stretch refers to the ratio between the length of the path discovered by the routing protocols and the length of the shortest path in terms of hop count. A routing protocol with lower hop stretch outputs shorter paths on average and achieves higher routing efficiency.

Fig. 3.10 shows the average hop stretch for the various algorithms, when the number of nodes grows from 1000 to 10000. When node density increases above 8, the hop stretch factor stabilizes for all protocols. It is clear that while VPCR guarantees packet delivery, it has the highest hop stretch of 1.92, because its angle-based greedy forwarding method fails frequently due to the localization errors and the imbalanced tree structure. Two adjacent nodes with non-continuous angular space will force VPCR to fall back towards the root, which will significantly increase the path length. On the other hand, with a single recovery tree, SCR has a stabilized hop stretch value of 1.22, which is much lower than VPCR, NoGeo and BVR, while higher than LCR (1.18) and Greedy. With 3 recovery trees, SCR can achieve the lowest hop stretch of 1.16 in high node density scenarios among all the routing protocols. With 10 recovery trees, the hop stretch of SCR will consistently remain below 1.2 and converge to 1.12 over the entire node density range.

The LCR protocol employs the Euclidean distance-based greedy algorithm as the primary routing method and in case of local minimum, the flooding-based recovery procedure in LCR will help to find

the shortest path to the destination. This mechanism generates lower hop stretch but also introduces exponentially increasing network traffic. The maximum packet delivery ratio of LCR shown in Fig. 3.9 is round 58%; thus, in nearly 42 percent of the test cases, scoped flooding is invoked by LCR to route packets out of the local minimum locations. Due to the high overhead, we believe that the overall hop stretch performance of SCR will be better than LCR in practice.

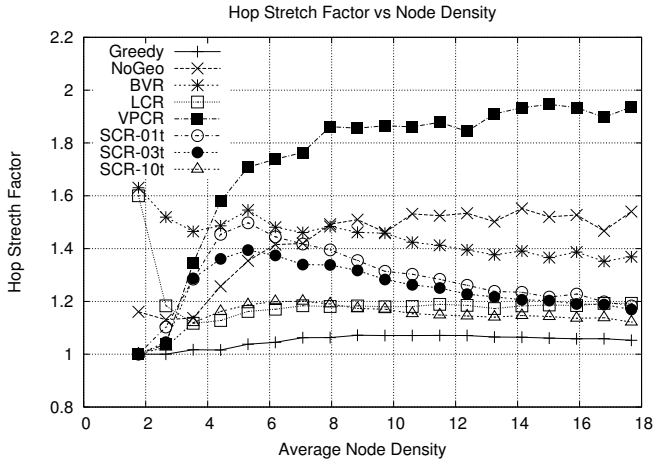


Figure 3.10: Average Hop Stretch Factor

3.4.3 Overhead per Packet Delivery

The *overhead* refers to the average amount of forwarding required to deliver one packet. It is computed as the ratio between the number of packet forwarding performed by the routing protocol to the number of packet forwarding required by routing the packet along the shortest path. The protocol with a lower traffic load involves less intermediate forwarding nodes and generates lower network overhead. Note that this is different from hop stretch since flooding is taken into account.

The overheads of the protocols are plotted in Fig. 3.11. Note that the y-axis is plotted on log scale. Recall that the expand-ring flooding is used by NoGeo, BVR and LCR to recover from greedy forwarding failures. During the flooding-based recovery stage, the number of nodes involved in packet forwarding increases exponentially and the overhead values for NoGeo, BVR and LCR reach 220.9, 9.4 and 70.3 respectively for the largest network evaluated (10,000 nodes).

In comparison, VPCR and SCR are much more efficient since flooding is not performed. Traffic load of VPCR is 1.9. The connectivity-based greedy forwarding in SCR manages to find shorter paths than VPCR and the spherical coordinate tree-based recovery refrains from injecting duplicate packets into the network, which helps SCR to achieve the lowest traffic load of 1.3.

3.4.4 Performance with Topology Changes and Recovery

We designed two models to investigate the performance of SCR under topology changes, named the *Horseshoe* model and the *Hollow Brick* model, as shown in Fig. 3.12. The *Horseshoe* model is used to evaluate the tree reparation procedure and the *Hollow Brick* model is used to evaluate the routing

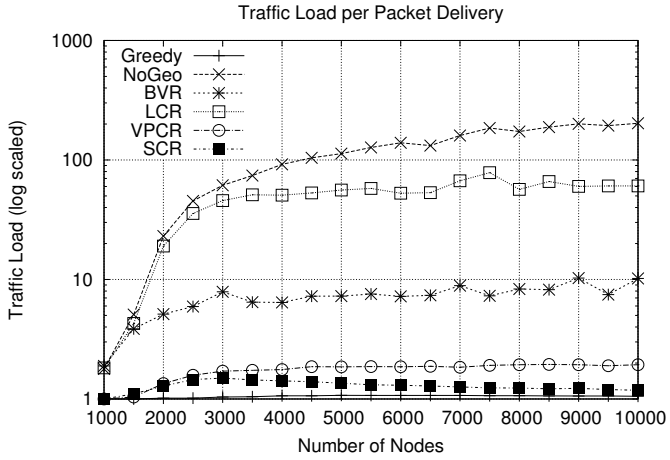
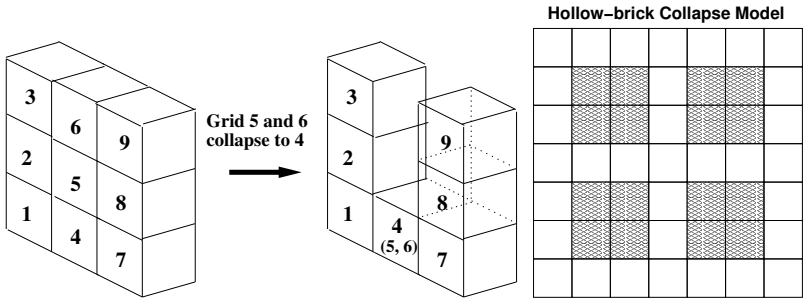


Figure 3.11: Traffic overhead per Packet Delivery



(a) Horseshoe (b) Hollow Brick

Figure 3.12: Modeling Network Topology Dynamics

performance after recovery. In the *Horseshoe* model, the wireless nodes are placed in 9 aligned cubic grids, which resembles the three-dimensional wireless sensor network deployed in high rise buildings. The grids are numbered from 1 to 9, each with a side length of $\frac{400}{3}$ and the root node is located at grid 1. We deploy 1000 nodes randomly into the 9 grids and select two grids, grid 5 in the middle and grid 6 on the boundary, to fail completely, which will affect a large portion of the original tree structure. The scenario imitates the fire damage to certain floors in the building. We count the remaining number of affected nodes after each iteration(one round of beacon exchange) until the the tree structure is fully repaired and recovery performance is given in Fig. 3.13. The recovery result shows that removing two grids from the topology will affect around 670 nodes and the recovery procedure will converge within 36 to 68 iterations, proportional to the relative distance between the recovered nodes and the root grid.

The distributions of subtree size before the grid collapse and after the recovery are shown in Fig. 3.14. The range of tree size varies from 0 to n, where n is the node number. We divide the tree size range evenly into 50 slots and count the number of trees allocated into each slot. Fig. 3.14 reveals that the overall distributions of tree size before the collapse and after the recovery remain similar, which implies that the local recovery method is stable and robust and the performance of the tree-based routing in SCR will not be significantly affected. Compared to the local recovery mechanism in SCR, the NoGeo, LCR and BVR protocols use global recovery - all the anchor nodes must re-broadcast to the whole network to

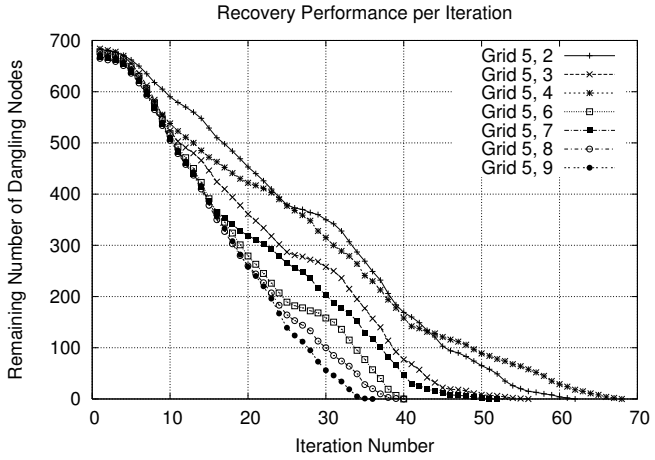


Figure 3.13: Recovery Performance after Two Grids Fail

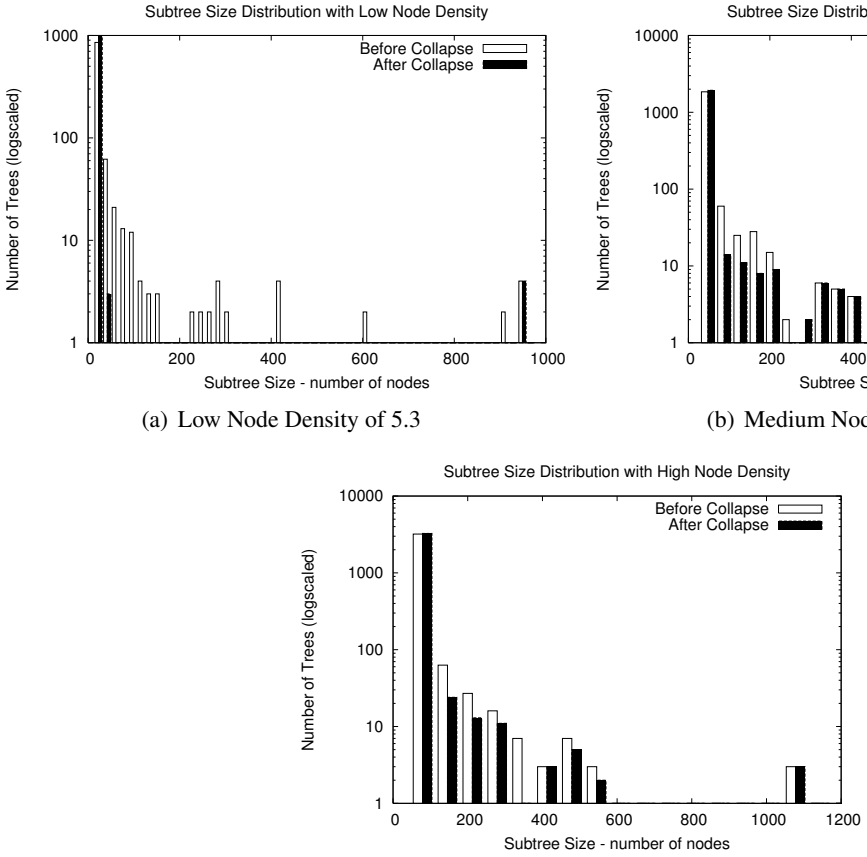
refresh the hop distance vector, which is more expensive.

The *Hollow Brick* model is used to measure the routing performance of SCR under node failures. Assuming that as a result of the disaster event, 4 grids in the center area collapse as shown in Fig. 3.12(b), creating 4 voids in the middle of the network. The affected nodes will initiate the recovery procedure and attach to new parents by using the algorithm described in Section 3.2.4. The source and destination nodes are randomly selected from the remaining grids. The rest of the simulation parameters remain unchanged.

The routing performance of the protocols before and after the topology change is given in Fig. 3.15 $\sim$ Fig. 3.17. As depicted in Fig. 3.15, node failures impose major impact on Greedy and LCR, reducing their delivery ratio by 31.3% and 78.8% respectively. The delivery ratio of BVR is reduced by only 9.4%, as an effect of its fall back scheme. The delivery ratios of VPCR and SCR dropped slightly at the medium density range of $5.3 \sim 10.6$ and remain the highest for the entire density range.

As shown in Fig. 3.16(a) and Fig. 3.16(b), the hop stretch is not significantly affected by the topology change for most of the protocols except BVR, whose hop stretch is increased by 43% from 1.58 to 2.26. Due to the lack of efficient greedy forwarding method, the hop stretch of VPCR remains above 1.8, while LCR and NoGeo has a hop stretch around 1.25. The hop stretch of SCR can be improved by deploying more recovery trees in the routing structure. With just one additional recovery tree, SCR can improve the hop stretch from 1.34 to 1.21 lower than that of LCR, while maintaining a much higher delivery ratio. The detailed hop stretch improvements of SCR in Fig. 3.17 clearly demonstrate that SCR is able to find more efficient routes as more recovery trees are deployed.

The traffic load values plotted in Fig. 3.18(a) and Fig. 3.18(b) show that the load of LCR and BVR are substantially increased by 4.6 and 2.33 times after the topological change, while SCR maintains the lowest traffic load around 1.3 consistently.



(a) Low Node Density of 5.3

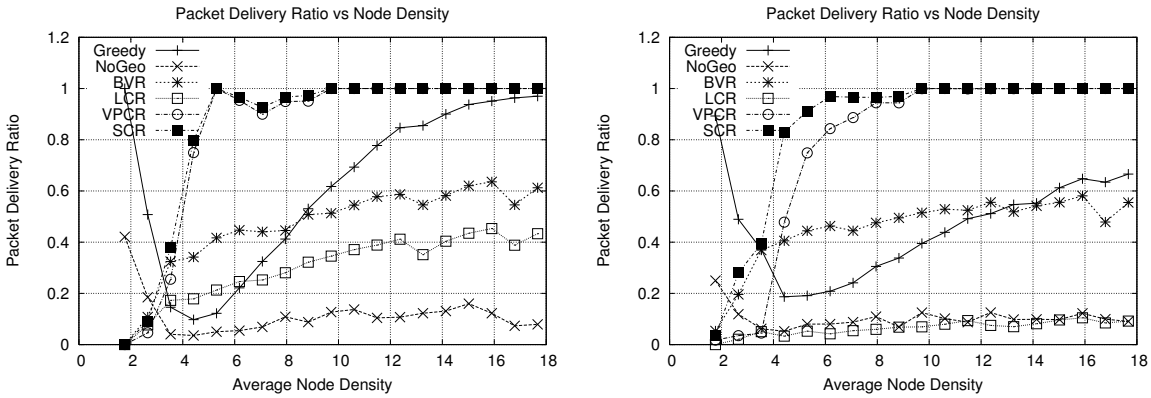
(b) Medium Node Density of 10.6

(c) High Node Density of 17.6

Figure 3.14: Distribution of Subtree Size after Recovery at Various Densities

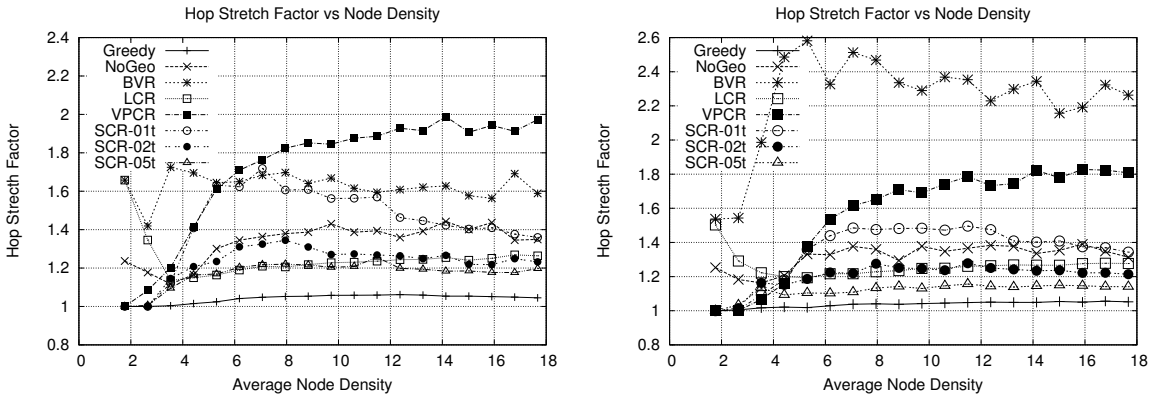
3.5 Summary

In this chapter, we have presented the Spherical Coordinate Routing protocol that provides guaranteed packet delivery in a three-dimensional network topology by constructing a spherical coordinate system over a tree structure. The SCR protocol complements the connectivity-based greedy forwarding with a recovery procedure that can identify the destination subtree according to the spherical coordinates. Compared to other 3D-compatible routing protocols such as NoGeo, BVR, LCR and VPCR, the Spherical Coordinate Routing protocol can achieve more efficient routing paths, maintain higher packet delivery ratio with lower network overhead and remain highly resilient to topology changes due to clustered node failures.



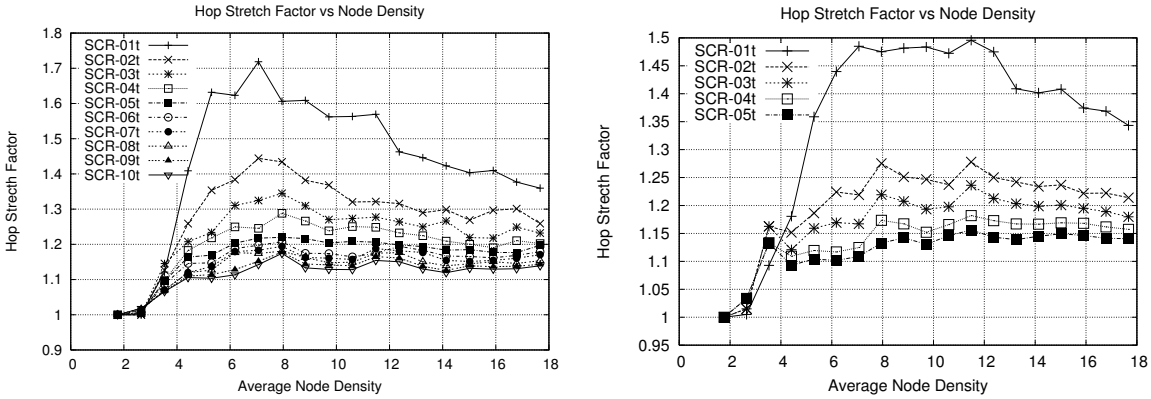
(a) Delivery Ratio before Collapse (b) Delivery Ratio after Collapse

Figure 3.15: Packet Delivery Before and After Collapse



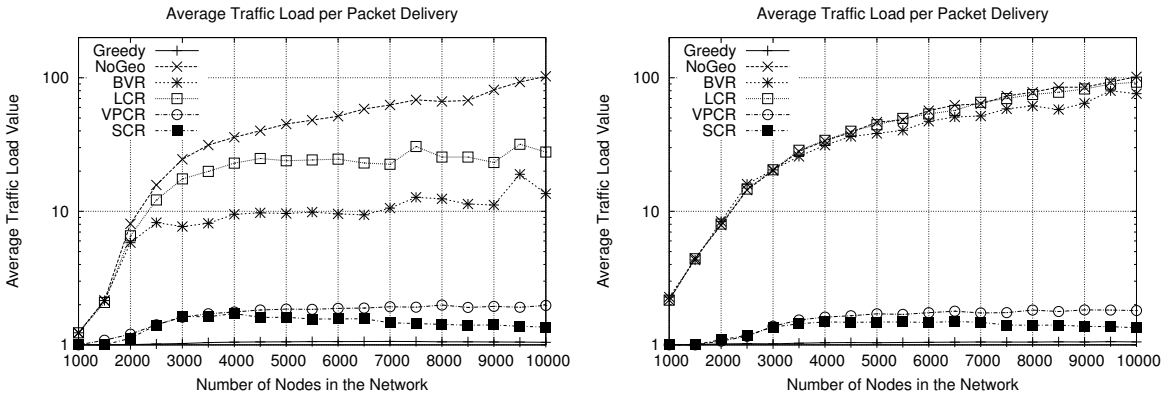
(a) Hop Stretch before Collapse (b) Hop Stretch after Collapse

Figure 3.16: Average Hop Stretch Before and After Collapse



(a) Hop Stretch, Multiple Trees, Before Collapse (b) Hop Stretch, Multiple Trees, After Collapse

Figure 3.17: Hop Stretch Performance with Multiple Recovery Trees



(a) Traffic Load before Collapse

(b) Traffic Load after Collapse

Figure 3.18: Traffic Load per Packet Delivery Before and After Collapse

Chapter 4

Practical Connectivity-based Routing using Dimension Reduction

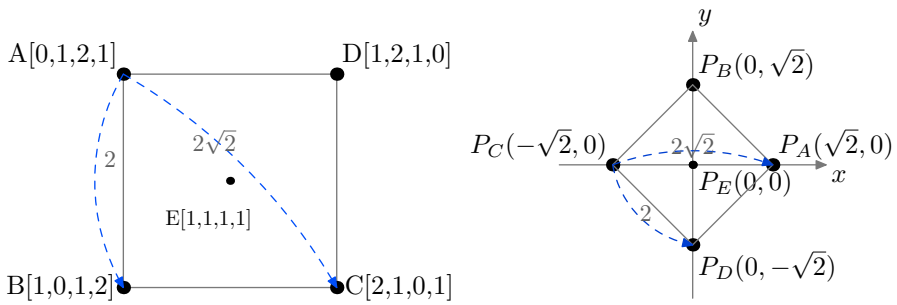
4.1 Introduction

We have seen that the performance of tree-based routing can be improved with greedy forwarding. The conventional greedy algorithms rely on node positions for packet forwarding. However, accurate position information is hard to obtain and violation of the unit disk assumption in real deployments may result in persistent routing failures. Therefore, routing can alternatively be performed based on the network connectivities, such as the hop distance. In hop count vector-based routing, a group of nodes are designated as the *beacons* L_i ($i \in [1, k]$), broadcasting their identities to the network. Each node measures the distance to the k beacons to create a hop count vector $H = [h_1, h_2, \ldots, h_k]$, where h_i is the hop count to beacon L_i. Routing is performed by treating the hop count vector H as the k-dimensional coordinates which can be accessed through a location service [76].

The number of beacons used during packet forwarding is crucial for the routing performance, creating a trade-off between control overhead and routing efficiency. The existing hop count vector-based routing protocols generally require more beacons to attain better performance. The resulted long hop count vectors are very expensive to manage on resource-constrained sensor nodes, making such an approach difficult to apply in practice.

In this work, we exploit the observation that as sensor networks are physically deployed in a 2D or 3D space, the intrinsic dimensionality of the topology is usually much lower than the number of beacons. Therefore, the dimension reduction techniques [83] can be applied to extract the major axises of the connectivity graph and project each node to an Euclidean space with lower dimensionality.

Our approach uses a Principal Component Analysis algorithm. Each node constructs a hop count matrix, describing the pair-wise distance of the beacons. The embedding algorithm apply *singular value decomposition* to the matrix to extract the most significant dimensions. The process returns a transformation matrix to capture the largest variance in the network topology. The inter-node distance is well preserved through the first few components in the coordinates, which are resilient to degenerate beacons. Experiment results show that by compressing the hop count vectors into three-dimensional coordinates, the nodes can maintain 95% of the packet delivery ratio (relative to using full hop count vectors) with a



(a) Connectivity Graph (b) Embedded Topology in $\mathbb{R}^2$

Figure 4.1: An Embedding Example with 4 Beacons: A, B, C and D

hop stretch of only 1.12.

The rest of the chapter is organized as follows. The details of the embedding procedure are explained in Section 4.2. The simulation results on large-scale performance comparison are presented in Section 4.3. The testbed settings and experimental results are detailed in Section 4.4. The summary of the chapter is given in Section 4.5.

4.2 PCA-based Routing Algorithm

In this section, we will present the details of the dimension reduction procedure with dimensionality analysis and highlight some implementation issues.

4.2.1 Embedding with Dimension Reduction

In a network with k beacons, each node measures the minimum hop count distance d_i to each beacon. The hop count vector $(d_1, d_2, \ldots, d_k)$ is utilized to address the nodes. In order to apply PCA, the vector distances between all pairs of beacon nodes are broadcasted to all nodes. Once a matrix containing all these vector distance information is obtained, each node independently performs the PCA-based dimension reduction process. The result provides nearly isometric embedding coordinates and ensures the transformed node distance in the embedded graph will approximate the original value. The new virtual coordinates of each node can be computed from the transformation matrix and the original hop count vector.

As an illustration, assuming four beacon nodes A, B, C and D are connected consecutively as shown in Fig. 4.1(a), the hop count vectors of the 4 beacons are $[0, 1, 2, 1]$, $[1, 0, 1, 2]$, $[2, 1, 0, 1]$ and $[1, 2, 1, 0]$. These vectors form a pair-wise beacon distance matrix M as given in Equation 4.1. We create a matrix M' by normalizing each row of M with a zeroed mean, such that $M'_{ij} = M_{ij} - (\sum_{x=1}^{k} M_{ix})/k$.

$$M = \begin{pmatrix} 0 & 1 & 2 & 1 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 \end{pmatrix}, \; M' = \begin{pmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \tag{4.1}$$

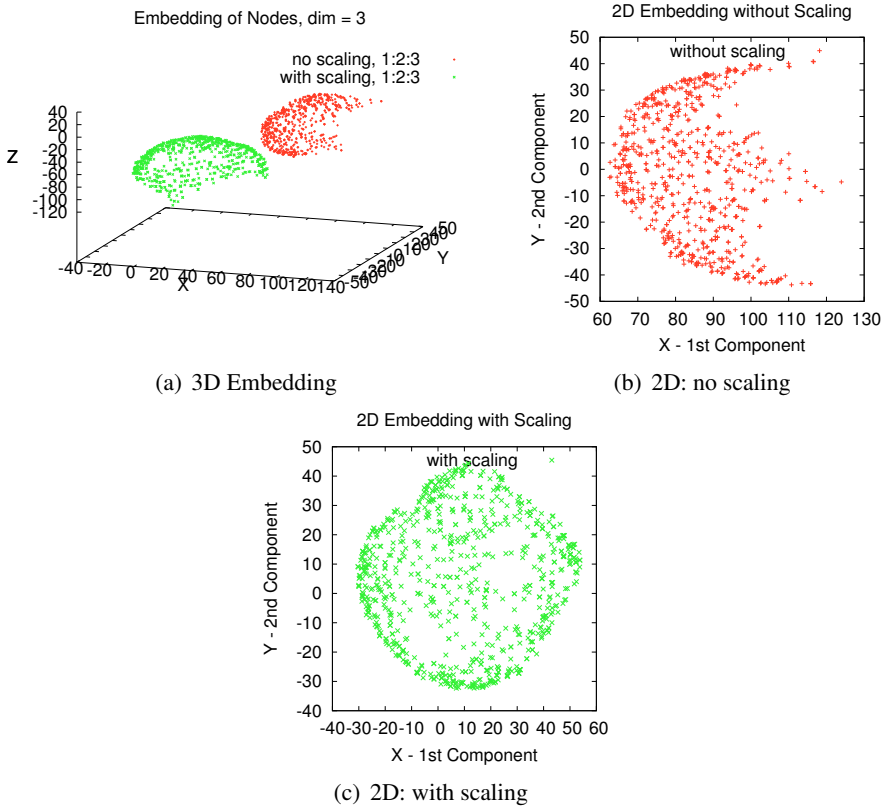
$$
\begin{aligned}
&M' = U \cdot S \cdot V^T, \text{ where} \\
&U = \begin{pmatrix} -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \end{pmatrix}, S = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \\
&V^T = \begin{pmatrix} \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} & 0 \\ 0 & -\frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \end{pmatrix}
\end{aligned} \tag{4.2}
$$

By applying *Singular Value Decomposition* (*SVD*) on the normalized distance matrix M', $[U, S, V^T] = svd(M')$, the result matrixes are computed by Equation 4.2. The diagonal matrix S contains the significance values $\sigma_1, \sigma_2, \ldots, \sigma_n$ for all components in a decreasing order. Matrix U is the transformation matrix for the PCA embedding. The PCA embedding coordinates P_A for beacon A can be computed as in Equation 4.3, where A' is vector A normalized with a zeroed mean. The PCA embedding coordinates for B, C and D are $P_B = [0, \sqrt{2}, 0, 0]$, $P_C = [-\sqrt{2}, 0, 0, 0]$ and $P_D = [0, -\sqrt{2}, 0, 0]$. By taking the first two components in the result vector as the embedding coordinates (x, y) in $\mathbb{R}^2$ space, the four beacon nodes can be plotted on a plane, which resembles their original structure as shown in Fig. 4.1(b). It is clear that the inter-node distance remains unchanged in this isometric embedding, $dist(A, B) = dist(P_A, P_B)$. For a normal node E with a hop count vector of $E = [1, 1, 1, 1]$, its embedding coordinates can be computed from the transformation matrix U as $P_E = E' \cdot U = [0, 0, 0, 0]$. The $\mathbb{R}^2$ coordinates of E is $(0, 0)$, which is consistent with the relative position of E.

$$
\begin{aligned}
P_A &= A' \cdot U = [-1, 0, 1, 0] \cdot \begin{pmatrix} -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \end{pmatrix} \\
&= [\sqrt{2}, 0, 0, 0]
\end{aligned} \tag{4.3}
$$

The zeroed-mean normalization is a necessary step, without which the first component in the output will represent the curvature of the data samples [84]. Incidentally, this normalization step is not performed in ICS. Fig. 4.2 provides an embedding example of 800 nodes. One way to evaluate the PCA generated virtual coordinates is by looking at the projection on the x-y plane which indicates that only the coordinates computed with scaling can correctly reflect the relative node position. As shown in Fig. 4.2(b), the projection without normalization fails to recover the original 2D topology.

An embedding example of a Gabriel graph with 800 nodes is shown in Fig. 4.3. With 70 beacons and a node density of 14.1, the PCA embedding correctly recovers the overall network structure with only minor rotation and stretch on the edge. This verifies that PCA-based dimension reduction is capable of accurately capturing the low-dimensional geometry of the networks by just relying on the connectivity



(a) 3D Embedding

(b) 2D: no scaling

(c) 2D: with scaling

Figure 4.2: An Embedding Example with and without Scaling

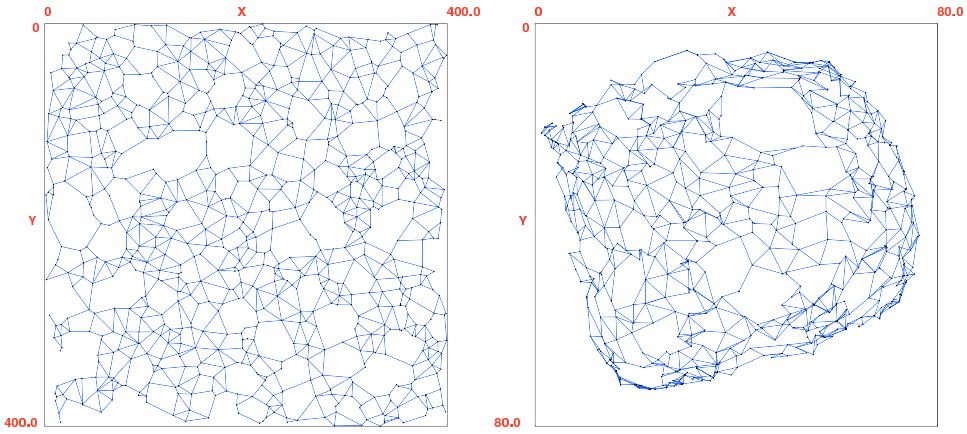
information.

4.2.2 Critical Dimension

A key issue in the dimension reduction procedure is to determine the critical dimensionality required. The rule of thumb [84] is to draw a scree plot of the principal components and select the *90 percentile* – the first k components that contribute to 90% of the total variance. The variance contribution of the i_{th} component is computed as $\sigma_i / \sum_{j=1}^{n} \sigma_j$, where σ_i is a diagonal entry in matrix S. In the example at Fig. 4.1(b), the contributions of the first two components are both 0.5.

We use simulation to examine the critical dimensionality required. Nodes are deployed in a 2D and 3D area with a side length of $400m$ and the communication range is $30m$. The number of nodes and beacons are varied in different scenarios. The distribution of principal components is displayed in the scree plot at Fig. 4.4. In Fig. 4.4(a), when all nodes are placed on a plane, each of first two components in the coordinates contributes to $23\% \sim 25\%$ of the total variance, while each non-intrinsic component contributes less than 7%. In Fig. 4.4(b), when the intrinsic dimensionality becomes three, each of the first three components contributes to $12\% \sim 15\%$ of the variance, while each of the rest contributes less than 5%.

The 90-percentile components are illustrated in Fig. 4.4(c) and Fig. 4.4(d). When 10% and 1% of nodes are randomly selected as the beacons for the 2D and 3D networks, 90% of the total variance comes from the first 20% of the components. Given that the actual network topology has a limited degree of



(a) Original Topology (b) Embedded Topology

Figure 4.3: A 2D Embedding of 800 Nodes with 70 Beacons

freedom, the number of dominant components should remain relatively stable.

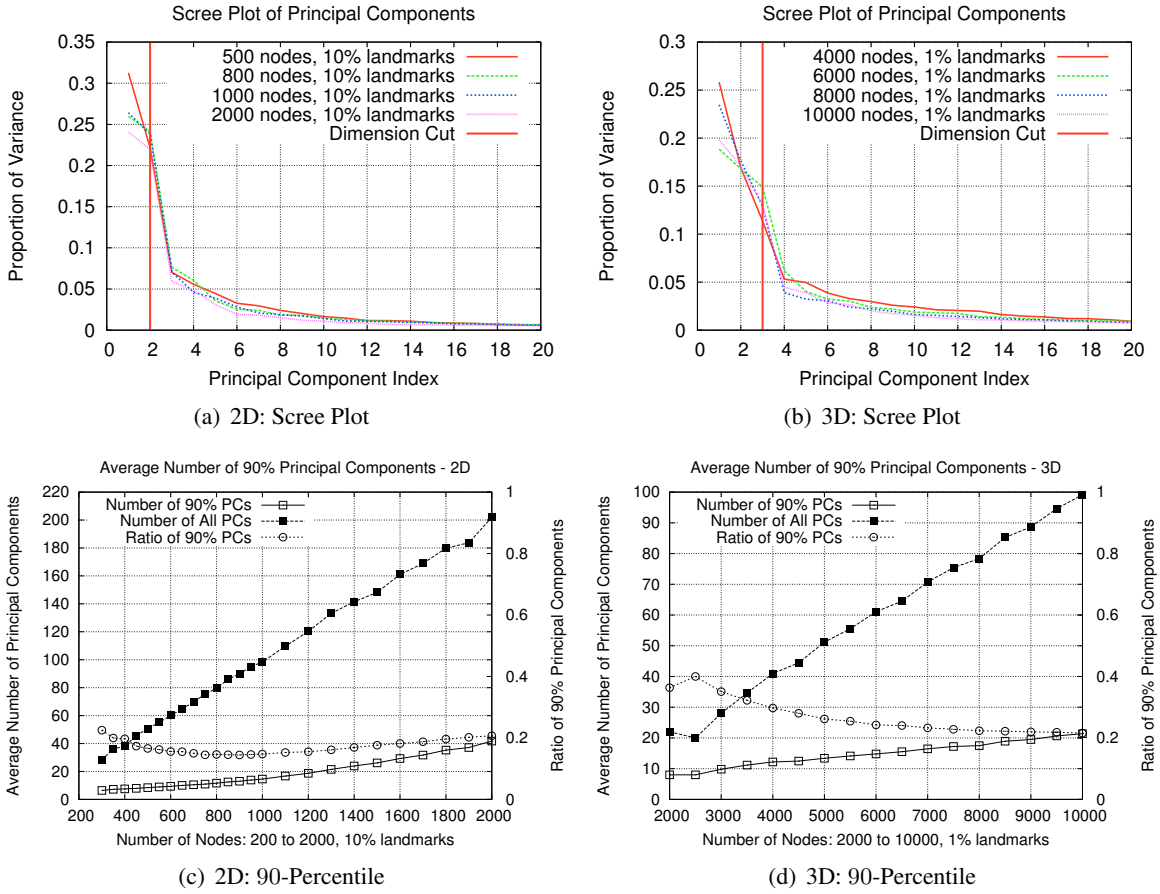
Based on the above results, it would seem that the critical dimension needed is fairly large and the potential for dimension reduction is limited. However, as our application is routing (rather than say data analysis or image processing), the need to include 90% of the total variance may be unnecessarily high. In fact, if the deployment is in a 3D space, as little as 4 principal components may suffice in some deployments.

In practice, we can resort to empirical measurements to assist in the dimension selection. The empirical results show that the critical dimensionality for hop count vector-based routing in 2D and 3D networks is only between 5 to 7 for uniformly random node placement, which is much smaller than the 20 components required for 90% variance coverage in a network with 1000 nodes.

4.2.3 Routing with Fallback

Assuming a source node S with PCA coordinates $[s_1, s_2, \ldots, s_k]$ has a packet for destination T at $[t_1, t_2, \ldots, t_k]$. Node S will search among all its m neighbors to find the next hop that can minimize the distance to T. During the greedy forwarding step, upon reaching a local minimum node, a fall-back mechanism is activated as depicted in Algorithm 3. For each destination T in a k-dimensional Euclidean space, the packet p contains a distance vector $[d_1, d_2, \ldots, d_k]$ to node T, where $d_i (i \in [1, k])$ represents the minimum Euclidean distance encountered from any visited node to T, computed from the first i components of their embedding coordinates. If an intermediate node P has no such a neighbor that can bring the packet closer to T by d_k in the k-dimensional space, P will search in the space of dimension (k-1) by examining distance d_{k-1}. The fall-back procedure continues until the next hop is found or the current node P is a local minimum for all d_i. If the fall-back mechanism fails, the scoped flooding will be used as the last resort.

An example showing how the fall-back technique can mitigate local minimum problems is illustrated in Fig. 4.5. For a dimension of $k = 2$, the Euclidean distance between S and T is $dist_2(S, T) = 60$, smaller than that of node A – $dist_2(A, T) = 63.5$. Upon the failure of $k = 2$, the routing algorithm will fall back to $k = 1$, where $dist_1(S, T) = 60$ and $dist_1(A, T) = 56$. With $dist_1(A, T) < dist(S, T)$,



(a) 2D: Scree Plot

(b) 3D: Scree Plot

(c) 2D: 90-Percentile

(d) 3D: 90-Percentile

Figure 4.4: Distribution of Principal Components in 2D and 3D Networks

node S will use node A as the next hop.

4.2.4 Implementation Issues

Data Structures

As shown in Fig. 4.6, each node maintains three data structures: the hop count vector *local_hv*, the list of neighbors *nbrlist* and the matrix containing the inter-beacon distance *hv_matrix*. *local_hv* stores the hop count entries to all beacons, where *parent_id* refers to the neighbor with the minimum hop count to that beacon. *nbrlist* contains the neighbor records, in which the *last_heard* variable records the time when a neighbor's beacon is received. The PCA coordinates *loc* in a neighbor entry stores the 3D embedding coordinates and the neighbor's hop count vector is stored in *hv*. *hv_matrix* keeps a record of the hop count vectors for all beacon nodes in multiple rows, where each row contains the corresponding *beacon_id*, *parent_id* and its distance to other beacons. The variable *parent_id* identifies the neighbor from which this beacon's hop count vector is received.

Algorithm 3 Routing Algorithm with Fall-back

Let p be the packet from src S to dst T
p arrives at node P with m neighbors $N_i (i \in [1, m])$
Dst T's coordinates $p.t = (t_1, t_2, \ldots, t_k)$ in k-dim space
Distance vector to T is $p.dv = [d_1, d_2, \ldots, d_k]$
// update the minimum distance to T
for $i \leftarrow 1, k$ **do**
    if $dist_i(P, T) < p.dv[i]$ **then**
        $p.dv[i] = dist_i(P, T)$ ▷ update d_i to avoid loop
    end if
end for
// search for the next hop
for $i \leftarrow k, 1$ **do** ▷ for each dimension
    $nexthop = null$
    for $j \leftarrow 1, m$ **do** ▷ for each neighbor
        if $dist_i(N_j, T) < p.dv[i]$ **then**
            $p.dv[i] = dist_i(N_j, T), nexthop = N_j$
        end if
    end for
    if $nexthop \neq null$ **then**
        return $nexthop$ ▷ return the greedy neighbor
    end if
end for
flood the packet p for $dist_k(P, T)$ hops. ▷ local minimum reached

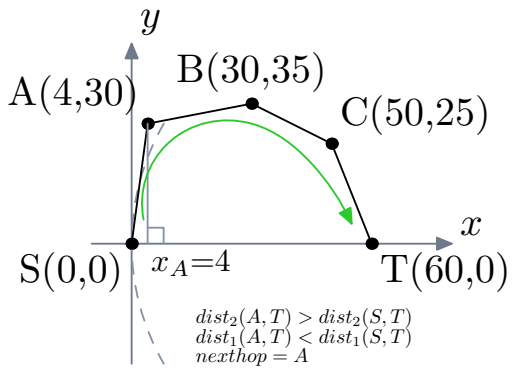


Figure 4.5: Packet Forwarding with Fallback

Memory Requirements

The memory cost for the data structures in Fig. 4.6 is estimated in Table 4.1. The local hop count vector *local_hv* has maximally 9 entries in the format of [beacon_id, hop count, parent_id], which consumes 54 bytes. Each neighbor entry uses 2 bytes for *id*, 4 bytes for *last_heard* timestamp and 12 bytes for PCA coordinates (x,y,z). The hop count vector *hv* in the neighbor entry contains maximally 9 hop count entries, each with a size of 6 bytes. The total number of bytes required for each neighbor is thus 72 bytes. Given a capacity of 20 neighbor entries, the *nbrlist* consumes 1440 bytes. The matrix of beacon hop count vector *hv_matrix* has 9 rows, where each row contains a *beacon_id*, a *parent_id* and 9 hop-count entries of format [beacon_id, hop-count]. The size of *hv_matrix* is 360 bytes. The total memory cost for *nbrlist*, *local_hv* and *hv_matrix* is thus 1440+54+360 = 1854 bytes.

In a network with less than 255 nodes, it is adequate to use `uint8_t` type for node id and hop-count instead of `uint16_t`, which will reduce the total memory cost from 1854 bytes to 1087 bytes

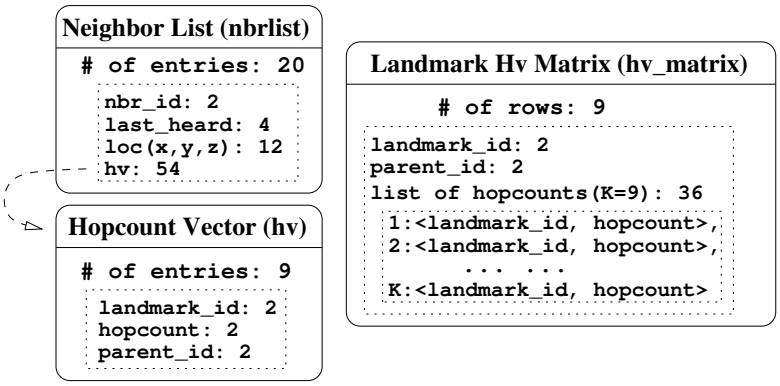


Figure 4.6: Data structures in the nodes: *nbrlist*, *local_hv* and *hv_matrix*

Table 4.1: Memory Cost (bytes): 20 neighbors and 9 beacons

size of *id*, *hop-count*	*nbrlist*	*local_hv*	*hv_matrix*	total
2 Bytes	1440	54	360	1854
1 Byte	880	27	180	1087

and also reduce the control packet size during network initialization. This modification is applied in the experiments. For generic network scenarios where the average number of neighbors is N and the number of beacons is B, the memory cost can be computed by Equation 4.4.

$$\begin{aligned} cost_{2b} &= (18+6B)N + 6B + 4B(B+1), \text{ if 2 bytes for node id and hop count.} \\ cost_{1b} &= (17+3B)N + 3B + 2B(B+1), \text{ if 1 byte for node id and hop count.} \end{aligned} \tag{4.4}$$

Timer-based Operation

The operation of the nodes are controlled by three timers: beacon timer, *nbrlist* refresh timer and the *beacon_hv* timer. The beacon timer allows each node to periodically broadcast a beacon packet containing its nodeid (*id*) and hop count vector (*hv*). Once a beacon is received from a neighbor *nbr*, the node will either insert a *nbr* entry to *nbrlist* if it is from a new neighbor or update the *nbr.last_heard* timestamp for an existing neighbor. For each new beacon learned from that beacon, a new hop count entry will be created in the local hop count vector *local_hv*. For all entries in *local_hv* whose parent is *nbr*, if the corresponding beacon is not found in *nbr*'s beacon, those invalid entries will be erased. To avoid the count to infinity problem, a *clear_entry* message will be sent for each removed entry, such that the stale beacon records in the child nodes will be cleared accordingly. The details are presented in Algorithm 4.

The *nbrlist* refresh timer is used to detect the missing neighbors due to link quality changes or hardware failure. A neighbor *nbr* will be removed if its beacon has not been received for 10 beacon intervals. For each local hop count entry $v \in local\_hv$, if $v.parent\_id = nbr.id$, v will be removed and a *clear_entry* messages will be sent. Similarly, for each row $r \in hv\_matrix$, if $r.parent\_id = nbr.id$, r will be removed from *hv_matrix*.

The beacon nodes use *beacon_hv* timer to broadcast the *beacon_hv* message, which consists of the beacon *id* and *local_hv*. All nodes receiving the *beacon_hv* message will insert the beacon hop count

Algorithm 4 Actions triggered by beacon timer

each node broadcasts beacon every 10s. a beacon from *nbr* is received.
if nbr.id $\notin$ nbr_list **then** ▷ *nbr* is a new neighbor
    add *nbr* to *nbr_list*
else ▷ update timestamp of an existing neighbor
    update *nbr.last_heard* to the current time.
end if
for each $v \in nbr.hv$ **do**
    if v.beacon_id $\notin$ local_hv **then** ▷ *nbr* has a new beacon
        insert [v.beacon_id, v.hopcount, nbr.id] to local_hv
    else if v.beacon_id = w.beacon_id and v.hopcount $<$ w.hopcount - 1, where $w \in local\_hv$ **then** ▷ *nbr* has a lower hop count
        set $w.parent\_id = nbr.id$ and $w.hopcount = v.hopcount + 1$
    end if
end for
for each $w \in local\_hv$ **do** ▷ refresh *local_hv*
    if $w.parent\_id = nbr.id$ **then**
        if $w.beacon\_id \notin nbr.hv$ **then** ▷ *w* is a stale entry
            remove *w* from *local_hv*, forward *clear_entry* msg.
        else if $w.beacon\_id = v.beacon\_id(v \in nbr.hv)$ **then**
            $w.hopcount = v.hopcount + 1$
        end if
    end if
end for

Algorithm 5 Actions triggered by *nbrlist* refresh timer

nbrlist is refreshed every 10 beacon intervals.
for each $n \in nbrlist$ **do** ▷ remove stale neighbors
    if n is a stale neighbor **then**
        remove n from *nbrlist*
        for each $w \in local\_hv$ **do** ▷ refresh *local_hv*
            if $v.parent\_id = n.id$ **then**
                remove v from *nbrlist*, send *clear_entry* msg
            end if
        end for
        for each row $r \in hv\_matrix$ **do** ▷ refresh *hv_matrix*
            if $r.parent\_id = n.id$ **then**
                remove r from *hv_matrix*
            end if
        end for
    end if
end for

vector to *hv_matrix*. In order to reduce the network traffic, each node will only forward the *beacon_hv* messages received from the parent nodes in *local_hv*. Once *hv_matrix* and *local_hv* have obtained compatible entries, each node will be able to calculate its embedding coordinates independently.

Algorithm 6 Actions triggered by *beacon_hv* timer

```
1: beacon nodes broadcast beacon_hv every 30s for 5 minutes.
2: the previous hop is denoted as prevhop
3: if beacon_hv.id ∉ hv_matrix then                          ▷ a new beacon_hv
4:     insert [beacon_id, prevhop.id, beacon_hv] to hv_matrix
5:     forward the beacon_hv
6: else
7:     if ∃ row r ∈ hv_matrix, r.beacon_id = beacon_hv.id then
8:         r.parent_id = prevhop.id, update r.hv with beacon_hv
9:         forward beacon_hv
10:    else                                                  ▷ this beacon_hv is not from parent
11:        ignore the beacon_hv
12:    end if
13: end if
```

4.3 Simulation Results

We first evaluate performance in simulation so results for large network can be obtained. In Section 4.4, evaluation on a 48 nodes MICAz testbed will be presented. In the rest of this chapter, we use *PCA* to denote the routing algorithm leveraging the PCA coordinates. For simulation performance comparison, we implemented Greedy, BVR, LCR and PCA in *ns-2* and the above mentioned protocols with S4 in a Java-based simulator. The output of the Java simulator is verified with *ns-2*. Due to the limited scalability of *ns-2*, the simulation results presented in this section are obtained from the customized simulator. PCA employs the *Jama* [85] library to perform singular value decomposition.

Table 4.2: Simulation Parameters

Name	*Value*
Deployment Space	400m×400m
Number of Nodes	100∼ 2000
Topology	50 topos/density, uniform random
Radio Range	$30m$, unit disk model
Connections	100 connections/topo
Routing Beacons	max = 10
Routing Protocols	Greedy, BVR, LCR, PCA, S4

The simulation parameters are listed in Table 4.2. The simulation area is a square plane with a side length of $400m$. The number of nodes deployed varies from 100 to 2000. We assume that we have an area with a fixed size to monitor; all sensor nodes have a fixed communication range; and the nodes are placed in a uniformly random manner. We want to examine how the delivery performance will change when the number of nodes varies. In this case, the network size (the total number of nodes) and the network density are consistent. With a radio range of $30m$, the node density range is $[1.77, 35.34]$. For each node density, 50 random topologies are generated and within each topology, 100 node pairs are chosen

to be the source and destination. For PCA, 10% of the nodes are selected to be candidate beacons. The maximum number of routing beacons for LCR and BVR is set to 10 as suggested in [23] and [24]. The number of beacons used by S4 is $\sqrt{n}$ by default [50], where n is the number of nodes. We use the five protocols to route packets for each connection and measure the packet delivery ratio, the hop stretch and the average flooding range as the performance metrics.

4.3.1 Packet Delivery Ratio

The packet delivery ratio is measured as the proportion of packets that can be successfully delivered without flooding. The packet delivery ratios of all protocols are given in Fig. 4.7. When the network density grows above 5, S4 can deliver 100% of the packets, as a result of its cluster-based routing scheme. For PCA to be effective, the number of beacons in the distance matrix should be larger than the dimensionality of the deployment space. In a sparse network with a node density less than 5, the number of beacons in a connected component is often below the threshold, leading to a performance inferior to BVR. The PCA coordinates achieve a higher packet delivery ratio, as the node density grows above 5.

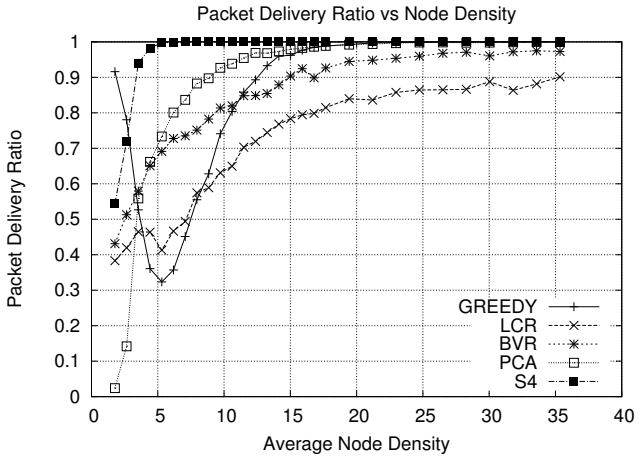


Figure 4.7: Packet Delivery Ratio

At the medium node density range of $5 \sim 15$, the packet delivery ratio of PCA is $73.3\% \sim 97.9\%$, significantly higher than that of LCR. BVR employs a backtrack procedure that diverts the packet to the closest beacon from the local minimum position, obtaining a delivery ratio of 69.1% to 90.4%. At the highest node density of 35, the delivery ratios for LCR and BVR reach 90.1% and 97.3, while PCA has a higher delivery ratio of 99.9%, nearly equivalent to the performance of greedy forwarding with perfect position information.

4.3.2 Hop Stretch Factor

Assuming the routing protocol generates a path of h_p hops and the shortest path has h_s hops, the hop stretch λ is computed as $\lambda = \frac{h_p}{h_s}$. In an ideal network, the protocol with a lower hop stretch λ achieves shorter routing paths and lower delay. The hop stretch performance is shown in Fig. 4.8.

The greedy protocol achieves the lowest hop stretch over the entire density range. The hop stretch of

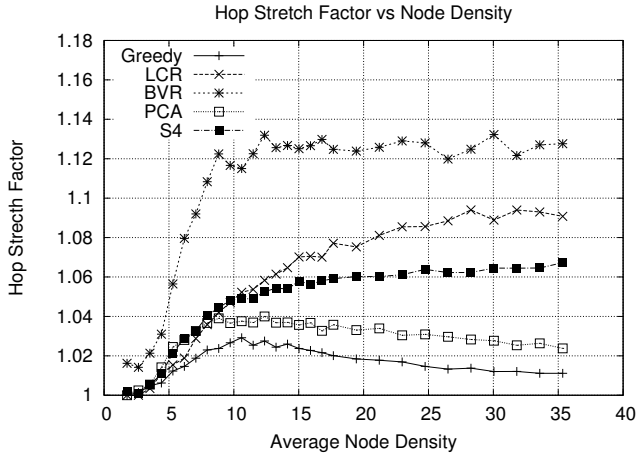


Figure 4.8: Hop Stretch Factor

PCA is lower than that of BVR and comparable to that of LCR and S4 at node densities less than 8. As the node density increases above 8, the hop stretch of BVR, S4 and PCA starts to stabilize, while LCR's hop stretch gradually increases. The converged hop stretch values for BVR, LCR, S4 and PCA are 1.12, 1.09, 1.07 and 1.02. Thus, for high density networks, the routing paths discovered by PCA is $5\% \sim 9\%$ shorter than the rest.

4.3.3 Scoped Flooding Range

For BVR, LCR and PCA, the primary forwarding procedure may fail to deliver a packet due to anomalies in the coordinates, while scoped flooding can be invoked as a recovery step. As the network traffic grows exponentially during flooding, the protocol with a shorter flooding range introduces less duplicate packets to the network.

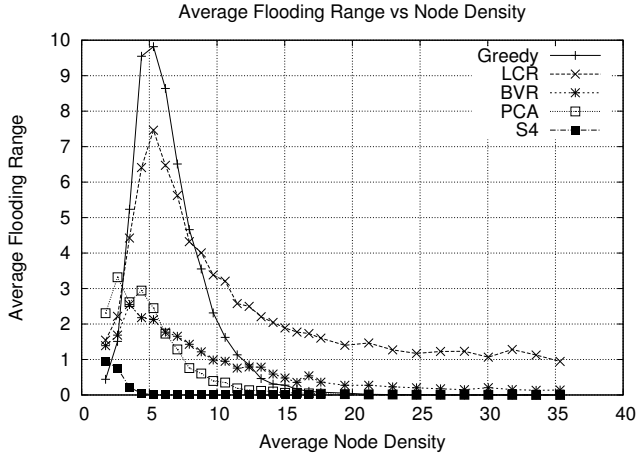


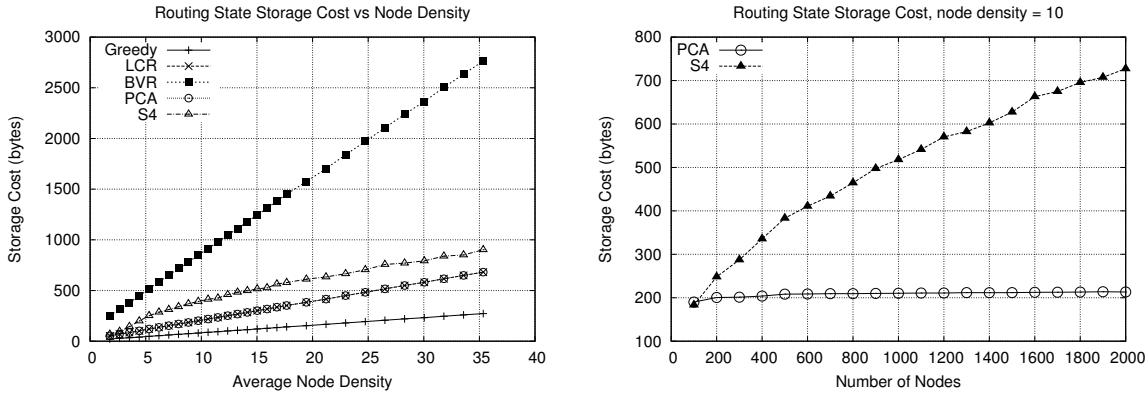
Figure 4.9: Average Flooding Range

In Fig. 4.9, S4 has the lowest flooding range because of its high delivery ratio, while PCA and BVR obtain a lower flooding range than LCR. At the node density around 5.3, the average flooding range for

all protocols except S4 reaches the peak as scoped flooding is frequently triggered. (We ignore the cases when density < 5, as the network is barely connected.) The maximum flooding ranges for Greedy, LCR, BVR and PCA are 9.8, 7.5, 2.1, and 2.4 hops. As the node density increases, all protocols obtain higher delivery ratio and the flooding range starts to decrease. The converged flooding range values for LCR and BVR are 0.95, and 0.14, while the flooding range of PCA approaches the minimum value of 0.

4.3.4 Storage and Transmission Cost

In Fig. 4.10 and Fig. 4.11, we plot the storage and transmission cost for all protocols to maintain the routing states. The storage cost includes the memory space required to keep the local coordinates and all the one-hop neighbors' coordinates. The transmission cost represents the amount of data sent by each node to build the hop count vectors, exchange the coordinates and update the location server.



(a) Storage Cost of Routing States

(b) Storage Cost at a Fixed Density

Figure 4.10: Storage Cost of the Routing States

In Fig. 4.10(a), the storage cost of all protocols grows as the network size increases, as more memory will be used to record the coordinates of the increasing neighborhood. The BVR has the highest storage cost, as its hop count vectors contain the beacon's ID, the next hop ID and the hop count distance. The storage cost of S4 is lower than BVR, but higher than that of LCR and PCA. The memory required in S4 depends on the network density and the number of beacons used. In a given network, a node will consume more memory to store the cluster information when fewer beacons are available, because the radius of the cluster equals the distance to the nearest beacon. LCR and PCA has lower storage cost than BVR and S4, as they only require ten entries in the coordinates, indicating the distances to the ten routing beacons or the top ten components of the dimension reduction results. We plot the storage cost of PCA and S4 for a fixed node density of 10 in Fig. 4.10(b). As the number of nodes increases, the storage cost of S4 increases accordingly, because the total number of beacons does not grow as fast as the network size, creating larger clusters in the networks. As the cluster diameter increases, a node will be covered by more clusters, leading to more entries to be stored in the routing table. The storage cost of PCA remains relatively constant, because the amount of routing states for PCA is determined by the number of neighbors, which is static in this case.

Fig. 4.11 shows that the transmission cost of BVR and LCR is much lower than S4 and PCA, as the

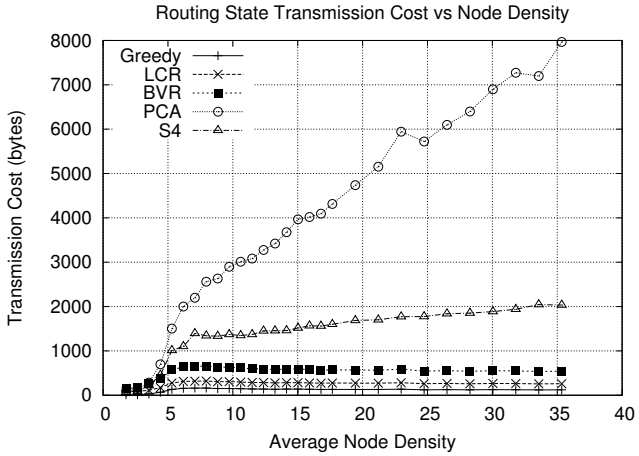


Figure 4.11: Transmission Cost of the Routing States

number of beacons used by both protocols is relatively small. The transmission cost of S4 mainly comes from the exchange of messages to build the clusters, which gradually increases with the network size. PCA sends more data than the other protocols, because PCA employs 10% of the nodes as candidate beacons and collects the hop count vectors from all beacons for the dimension reduction procedure. The number of beacons increases linearly with the network size, leading to a persistent increase in the transmission cost. The maximum cost for PCA is around 8000 bytes, or approximately 80 packets, given that the maximum packet size in TinyOS-2 is 128 bytes. We believe this transmission cost is still acceptable, as the full scale collection of hop count vectors is only invoked once at the network initialization stage.

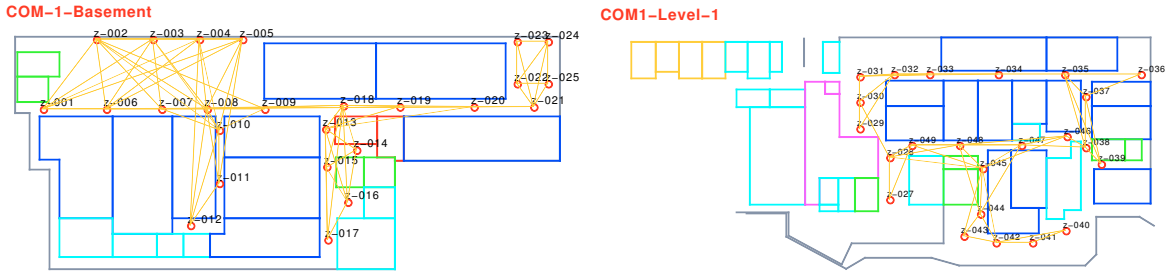
4.3.5 Effects of Dimensionality

While utilizing more components in PCA coordinates provides better routing performance, it also brings higher demands for processing time and memory space in the packet header. Determining the critical dimensionality is important for obtaining a balance between routing performance and overhead. We use empirical results to derive the critical dimensionality value.

Table 4.3: Performance of PCA at Critical Density of 5

Metric	2D Networks				
Dimensionality	2	3	5^*	10	20
Delivery Ratio	52.6%	62.4%	70.5%	73.3%	73.5%
Hop Stretch	1.04	1.03	1.03	1.02	1.02
Flooding Range	4.12	3.28	2.60	2.45	2.44

In general, when node density increases, the need for higher dimension reduces. Therefore, the effect of dimension is more crucial at lower node density. The routing performance of PCA with node density 5.3 under various dimensionalities is summarized in Table 4.3. We choose to highlight node density 5.3 because this is the minimum node density required for any geographical or connectivity-based algorithm to work well. For higher node density, equal or less components will be needed.



(a) Node z-001 to z-025 in Basement

(b) Node z-027 to z-049 in Level-1

Figure 4.12: Deployment Floorplan of the Testbed(inter-floor links not shown). Beacons: 2, 8, 18, 21, 28, 31, 36, 38, 45

As shown in the table, routing performance gradually improves as more dimensions or components are utilized. In this 2D deployment, the incremental improvement diminishes rapidly beyond a dimension of 5. As the components are sorted in their significance order, the subsequent components in the PCA coordinates will have even lower influence on routing performance.

To summarize, for 2D networks, the performance of PCA stabilizes once the dimensionality reaches 5. For 3D networks, the critical dimensionality is around 7.

4.4 Testbed Results

To explore the practicality of the dimension reduction method for real sensor networks, we implemented the PCA algorithm on MICAz [80] motes with 4KB RAM running TinyOS-2 [36]. The singular value decomposition code is adopted from [86]. The experiments were conducted on a testbed deployed on two floors of an office building, as demonstrated in Fig. 4.12. The parameter settings are listed in Table 4.4.

Table 4.4: Parameter Settings of the Testbed

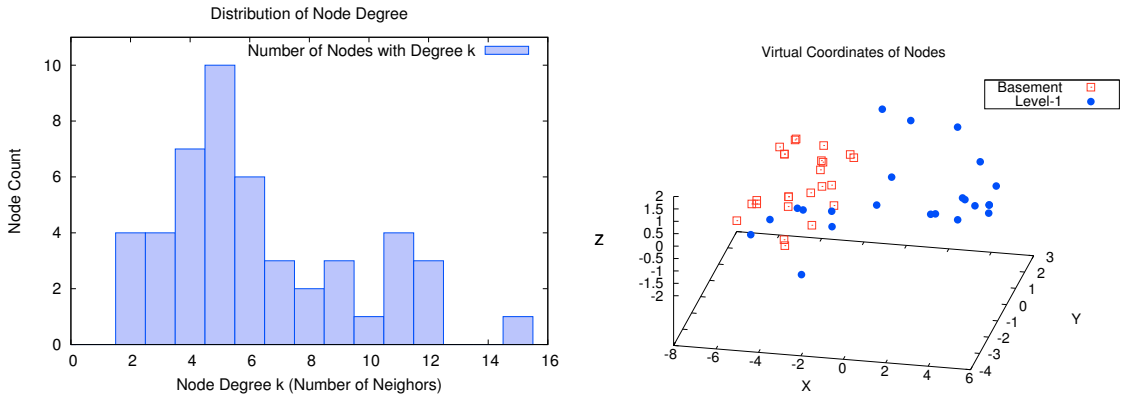
Parameter	*Value*
Network Size	48 MICAz nodes, 303 directional links
Num of Neighbors	avg = 6.31, min = 2, max = 15
Transmission Power	31 (max, 0 dBm)
802.15.4 Channel	26 (default)
Number of Beacons	9 (4 in basement, 5 in level-1)
Intrinsic Dimension	3 (25 in basement, 23 in level-1)

The testbed contains 48 nodes and 303 directional links. All nodes transmit at the maximum power level of 31. We deploy nodes z-001 to z-025 at the *basement* floor and nodes z-027 to z-049 at the floor *level-1*. Nine nodes were chosen as the beacons, four in the basement and five in level-1. Fig. 4.13 gives a snapshot of the deployment. The distribution of node degree is displayed in Fig. 4.14(a), where the average number of neighbors at each node is 6.31.

Network initialization took five minutes for neighbor discovery and hop count propagation. Once the beacon hop count matrix and the hop count vector are established, each node computes its coordinates individually as described in Section 4.2.1. Only the first three components are used for the embedding

(a) z-003 at Basement (b) z-027, z-028 at Level-1

Figure 4.13: Images of Nodes deployed in the Testbed



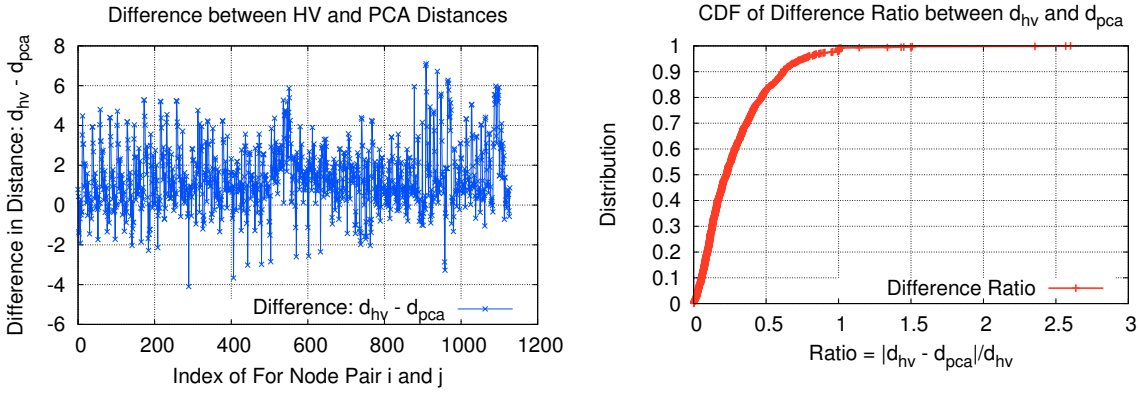
(a) Distribution of Node Degree (b) Network Embedding

Figure 4.14: Node Degree and 3D Embedding Graph of 48 Nodes

in order to reduce the overhead in the packet header. The coordinates plotted in Fig. 4.14(b) show that nodes on different floors clearly form two clusters, implying that the PCA coordinates can successfully retain the locality feature.

4.4.1 Distance Metric Comparison

To compare the Euclidean distance computed from the hop count vectors and the PCA coordinates, we calculate the inter-node distances in both ways. For two nodes A and B with hop count vectors $[a_1, a_2, \ldots, a_n]$ and $[b_1, b_2, \ldots, b_n]$, the hop count vector distance d_{hv} is computed as $d_{hv} = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$. The PCA distance d_{pca} between A and B is the Euclidean distance computed from their PCA coordinates (x_A, y_A, z_A) and (x_B, y_B, z_B). We plot the difference between these two distances $\delta = d_{hv} - d_{pca}$ in Fig. 4.15, where 50% of the distances computed from the PCA coordinates are within a deviation of 0.25 from the corresponding full hop count vector distances.



(a) Difference: $d_{hv} - d_{pca}$

(b) CDF of Difference Ratio

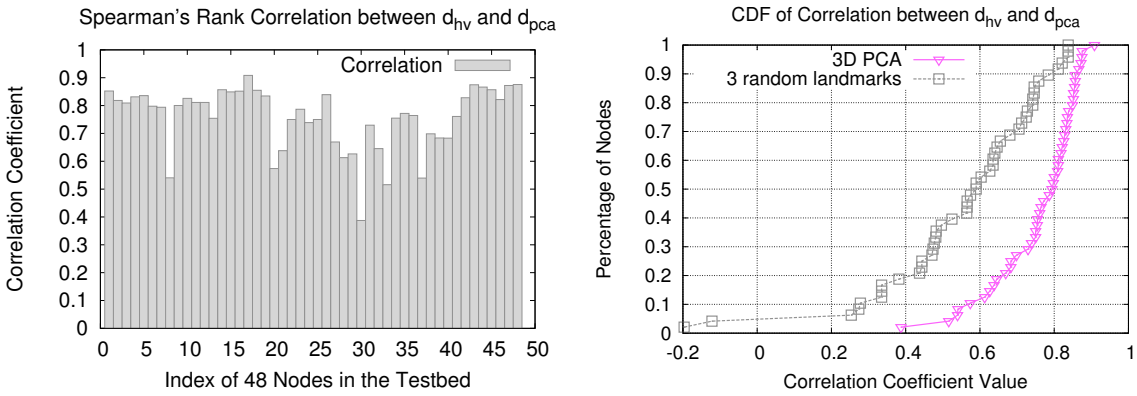
Figure 4.15: Comparison between Distances: d_{hv} and d_{pca}

We use the *Spearman's Rank Correlation* metric to quantify the effectiveness of using PCA coordinates to replace hop count vectors. Given two ranking lists $X = [x_1, x_2, \ldots, x_n]$ and $Y = [y_1, y_2, \ldots, y_n]$, the Spearman's Rank Correlation Coefficient ρ between X and Y can be computed by Formula 4.5. The value of ρ varies in the range of $[-1, 1]$. A higher coefficient value indicates a higher consistency between the two lists.

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)}, \text{ where } d_i = x_i - y_i \tag{4.5}$$

For each node N_i, we compute its hop count vector distance d_{ij} to each node $N_j (j \in [1, n])$ and form a distance vector $D = [d_{i1}, d_{i2}, \ldots, d_{in}]$. According to the distance d_{ij}, we assign a rank $r_{ij} (r_{ij} \in [1, n])$ to node N_j, such that if $d_{ij} < d_{ik}$, $r_{ij} < r_{ik}$. We use $R_{hv}^i = [r_{i1}, r_{i2}, \ldots, r_{in}]$ to denote the ranking list of hop count vectors at node N_i. We compute another ranking list with the PCA coordinates and name it R_{pca}^i. For each node N_i, the correlation ρ_i between R_{hv}^i and R_{pca}^i is computed as in Equation 4.5.

The correlation coefficient values at each node are depicted in Fig. 4.16(a). The correlation lies in a range of $[0.39, 0.91]$, with an average equal to 0.76 and a standard deviation of 0.017. As shown in



(a) Correlation at 48 Nodes

(b) CDF of Correlation

Figure 4.16: Spearman's Rank Correlation between d_{hv} and d_{pca}

Fig. 4.16(b), 80% of the nodes have a correlation above 0.65, indicating that the node locality characteristics estimated from the PCA coordinates is highly consistent to that of the full hop count vectors using all 9 beacons.

For comparison, Fig. 4.16(b) shows the CDF of distance correlation using 3 randomly chosen beacons as is done on LCR (data is averaged of 5 different data sets). It is clear that with the same amount of overhead, the randomly chosen beacons significantly deviates from the geometry of the original graph.

4.4.2 Routing Performance

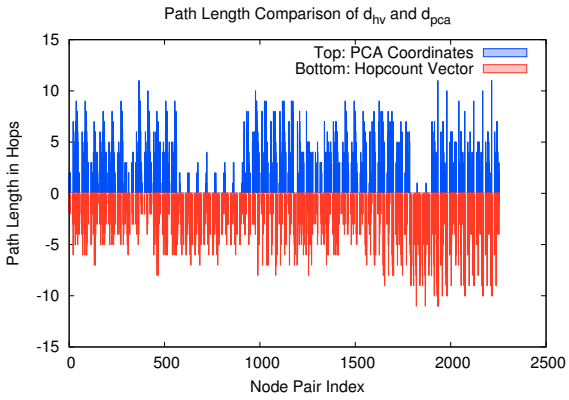
To compare the routing performance, we employ greedy forwarding to find a routing path between two nodes based on d_{hv} and d_{pca} respectively. We measure the number of paths discovered and the path length L_{hv} and L_{pca}. The hop stretch $\lambda = \frac{L_{pca}}{L_{hv}}$ will imply the performance degradation by using the lower-dimensional coordinates instead of the full hop count vector for routing. The results are provided in Table. 4.5.

Table 4.5: Routing Performance Evaluation of d_{hv} and d_{pca}

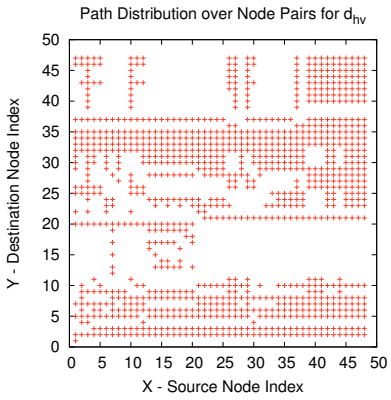
	No. of Paths	Path Length	Hop Stretch over d_{hv}
d_{hv}	581+467=1048	[1, 11], avg = 3.85	1
d_{pca}	581+417=998	[1, 11], avg = 3.94	[0.17, 4], avg = 1.12

With 48 nodes in the network, we can form 2256 pairs of source and destination. The d_{hv} distance metric discovered 1048 paths and the d_{pca} metric discovered 998 paths. Only 581 of the paths are common to both metrics. In terms of packet delivery ratio, d_{pca} found 95.2% of the paths discovered by d_{hv}. The path length of applying d_{hv} and d_{pca} varies in the range of $[1, 11]$, where d_{hv} has an average of 3.85 hops. The average path length of d_{pca} is 3.94, a slight increment of 2%. The hop stretch of d_{pca} over d_{hv} ranges from 0.17 to 4; thus, either metric has found some paths shorter than the ones discovered by the other. The overall hop stretch factor of d_{pca} is 1.12, representing an average increment of just 1.3 hops in the worst case.

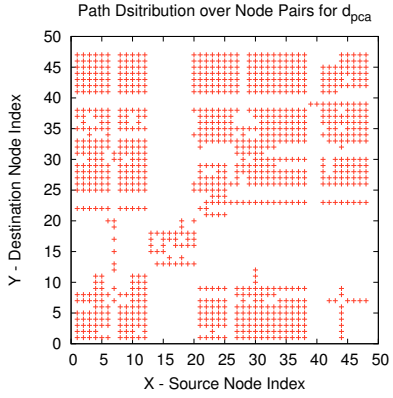
The path lengths are displayed in Fig. 4.17(a) and the routes discovered by d_{hv} and d_{pca} are marked



(a) Path Length for Node Pairs



(b) Paths found by d_{hv}



(c) Paths found by d_{pca}

Figure 4.17: Distribution of paths discovered by d_{hv} and d_{pca}. A + at position (x, y) indicates a path is found from node x to node y.

in Fig. 4.17(b) and Fig. 4.17(c). The route map of d_{pca} in Fig. 4.17(c) clearly indicates that nodes from z-013 to z-020 form a cluster inaccessible from the rest part of the network, causing most routing failures. This is due to the lack of resolution in the input hop count vectors. It is possible to alleviate the problem by applying pair-wise transmission power control [87] to create a small-scale multi-hop topology in these nodes or introducing additional beacon nodes in this cluster. Meanwhile, the route map of d_{hv} in Fig. 4.17(b) is less indicative for network performance diagnosis.

4.4.3 Packet Overhead

The control overhead in the packets mainly comes from the hop count vectors or coordinates used to address the destination. To use the full hop count vector of the destination, each packet must carry the hop count entries to 9 beacons. Assuming that both *nodeid* and *hopcount* are of `uint_16` type, this constitutes 36 bytes. The d_{pca} metric relies only on the 3D embedded coordinates, which cost 12 bytes – just one third of that in d_{hv}. It can be further reduced to 8 bytes, if all nodes are placed on a plane. Although the PCA operation has to maintain a hop count matrix of size n^2 (n is the number of beacons), the memory and computational overhead is only required once during the initialization stage. The memory space can be reclaimed once the procedure completes. Therefore, the computational overhead of the PCA coordinates is transient, while the storage and communication overhead of applying full hop count vectors is persistent throughout the entire network lifetime.

In summary, using dimension reduction method, the routing protocol can work with only 3 components, while maintaining equivalent packet delivery performance with negligible hop stretch overhead.

4.5 Summary

In this chapter, we evaluated the technique of applying dimension reduction method for hop count vector-based routing. By extracting the underlying dimension information from the sampled connectivity graph, the resulted coordinates can significantly improve routing efficiency. We implemented the PCA algorithm on MICAz nodes and conducted experiments on a medium-scale testbed. With a 3D embedding of 9 beacons, the distance computed from the virtual coordinates can closely approximate the distance computed by the full hop count vector. The experiment results show that the dimension reduction algorithm can effectively reduce the packet overhead to make connectivity-based greedy forwarding more applicable for real sensor network deployments, without compromising the routing performance significantly.

Chapter 5

Implementation and Experiments on the Indriya Testbed

Many proposed algorithms are impractical because they cannot fit on practical motes like TelosB. In TOSSIM, the same memory limitations are not present, hence we deploy and evaluate our algorithms on a real testbed. We have implemented PCA, LCR and VPCR in TinyOS 2.1 and perform point-to-point transmission on the Indriya testbed.

The key components of the algorithms incorporate some changes to the variable types and the data structures in order to cope with the software and hardware constraints in the sensor nodes. This will help us to understand the scalability of the algorithm when being deployed in the current sensor platform. The experiments with real link quality measurements will also reveal the actual routing performance of the protocols in a real network environment. The implementation process helps to verify the applicability of the current point-to-point routing design and it also identifies some key issues, that may affect the run time performance of all protocols.

Due to the complexity of the algorithms and the hardware constraints in the MICAz and TelosB sensor platforms available to us, we leave the implementation of SCR for future work.

5.1 Implementation of the PCA-based Routing Algorithm

We compare the performance of PCA with LCR and VPCR, because the latter two represent two main streams of the connectivity-based routing protocols: hop count vector-based routing and tree-based routing. The implementation details of these protocols will be introduced in this section, including link quality estimation, construction of hop count vectors, computation and allocation of the geographic coordinates and the location service.

The procedures to build the VPCR tree and compute the PCA coordinates are summarized in Fig. 5.1. After booting up, all nodes will initiate the neighbor discovery process by exchanging beacons. The periodic beacons are used to estimate the packet reception ratio and ETX (expected number of transmissions) on each link. The anchor nodes for the PCA and LCR routing will announce their identities to the network such that the hop count vectors can be constructed. Meanwhile, VPCR will build a spanning tree from the links with low ETX values and the size of each subtree will be reported to the root. From the

root downwards, the nodes at each level will allocate the polar coordinates to the children proportional to the subtree size. Once the PCA coordinates and the polar coordinates have been assigned, all nodes will upload their addresses to the root to create a lookup directory for the location service. We include the description of location service because it is a fundamental requirement for all geographic routing protocols and it is often ignored in the performance evaluation of previous works. The design of the location service will impose significant impacts on the outcome of geographic routing. The details of each step are given in the following sections.

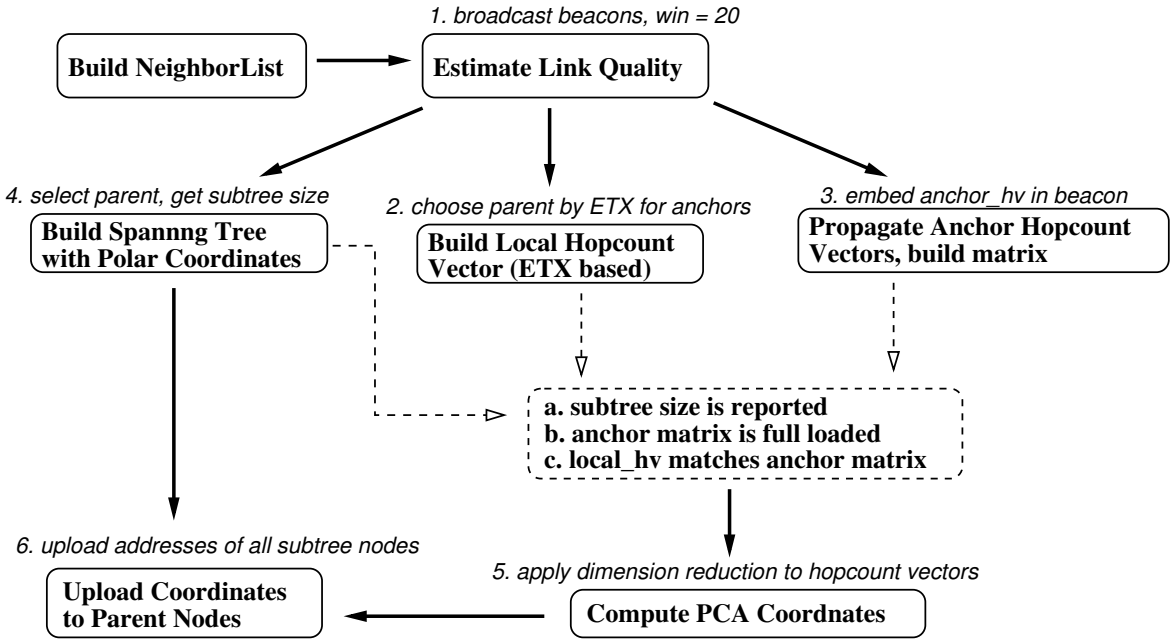


Figure 5.1: Procedures to Compute Coordinates and Update Address Cache

5.1.1 Link Quality Estimation

All nodes monitor the link quality by keeping track of the incoming beacons. As described in Appendix C, the link quality results measured by broadcast probes give a close approximation to the results of the unicast probes. Hence, all neighbors are probed simultaneously which will substantially shorten the probe latency compared to iterative unicast probes, especially in a dense network. Since beacons are commonly used in wireless network protocols to maintain network connectivity, utilizing the existing control traffic will reduce the network overhead and conserve the bandwidth for data traffic.

The receiver can keep track of the lost beacons and derive the packet reception ratio from the sequence number in the beacons, as shown in Algorithm 7. A node keeps two counter r and l for each neighbor N_i, where r is the number of beacons received from N_i and l is the number of lost beacons. When a beacon B arrives, the receiver will compare the sequence number in the beacon $B.seqno$ with the previous sequence number from N_i, $N_i.seqno$. The gap between the two numbers is computed as $gap = B.seqno - N_i.seqno$ and the number of lost beacons is updated as $l = l + (gap - 1)$. As the sequence number is of `uint8_t` type, if it wraps around after 255, we will have $gap < 0$. In this case, we calibrate gap by setting $gap = gap + 255$. The packet reception ratio is updated as $\frac{r}{r+l}$ after every

Algorithm 7 Estimating Incoming PRR from Received Beacons

1: // Assume Beacon B is received from neighbor N_i
2: $Beacon\_Win = 20$ ▷ PRR estimation interval
3: $gap = B.seq - N_i.seq$
4: $N_i.num\_recvd$ += 1
5: **if** $gap < 0$ **then** ▷ beacon sequence number wrapped around
6: gap += 255; ▷ $B.seq$ is `uint8_t`
7: **end if**
8: $N_i.num\_lost$ += $(gap - 1)$
9: **if** $N_i.num\_lost + N_i.num\_recvd \geq Beacon\_Win$ **then** ▷ enough beacons to update PRR
10: $N_i.incoming\_prr = N_i.num\_recvd/(N_i.num\_recvd + N_i.num\_lost)$
11: $N_i.num\_recvd = 0$ ▷ reset the counters
12: $N_i.num\_lost = 0$
13: **end if**
14: $N_i.seq = B.seq$ ▷ update the current sequence number

20 beacon intervals.

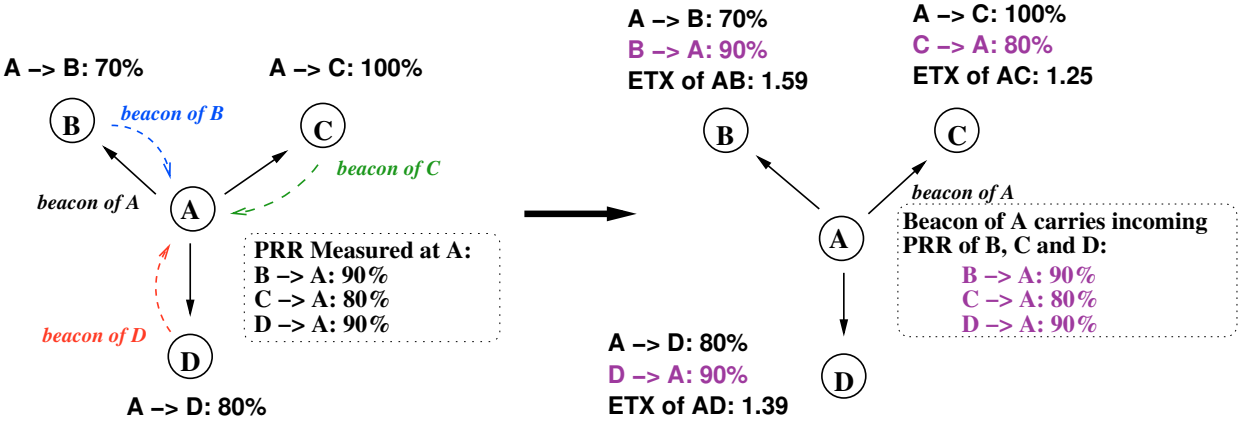


Figure 5.2: Link Quality Estimation and ETX Computation

While the incoming packet reception ratio measured by Algorithm 7 is directional, we use the estimated number of transmissions (ETX) to quantify the link quality. Assuming that p and q represent the incoming and outgoing packet reception ratios, the link ETX is computed as $\frac{1}{pq}$. A node can measure the incoming PRR p by monitoring the beacons received. In order to advertise the corresponding outgoing PRR to the neighbors, each node will iteratively attach the incoming PRRs of 5 neighbors in each beacon message. This procedure is illustrated by an example in Fig. 5.2. For a node with 40 neighbors and a beacon interval of 10 seconds, it will take 8 beacon messages to announce the incoming PRRs from all neighbors. Therefore, it will complete the exchange of link qualities after 80 seconds. In order to maintain a stable topology, the links with low ETX are used to construct the spanning tree and propagate the hop count vectors.

5.1.2 Local Hop count Vector

The anchor nodes are identified by a flag in their beacons. In a network with k anchor nodes, k spanning trees will be built, one from each anchor. Every node will choose a neighbor with the minimum cumulative ETX as the parent to reach the anchor. The hop count metric is not used here, because hop count can

be easily interfered by transient links providing shorter paths but poor connectivity, leading to unstable topologies. The ETX metric can effectively reduce oscillation in the topology based on link qualities estimated through continuous observation. Compared to the blacklisting method, the ETX metric will prevent the network segmentation problem caused by a rigid threshold. Exploiting all links available in the network will also improve the resilience to changes in the nodes' connectivity.

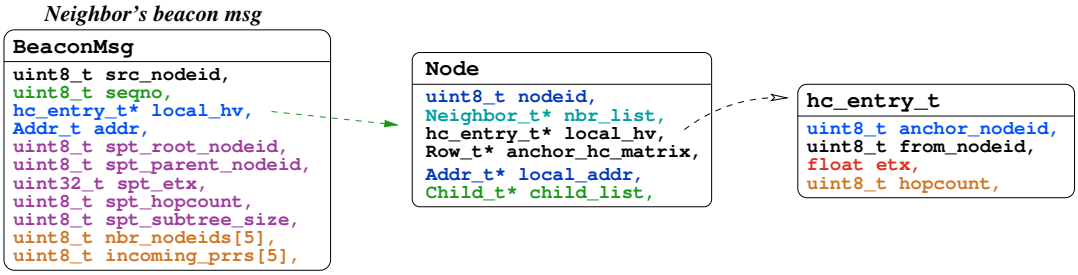


Figure 5.3: Format the Hop count Vector Entries

As depicted in Fig. 5.3, the hop count vector contains a list of entries, corresponding to the anchors in the network. A hop count entry consists of the anchor's node ID, the parent node ID, the cumulative ETX and the hop distance between the node and the anchor. The hop count vector `local_hv` is embedded in the beacons, such that a spanning tree based on minimum ETX can be constructed from each anchor.

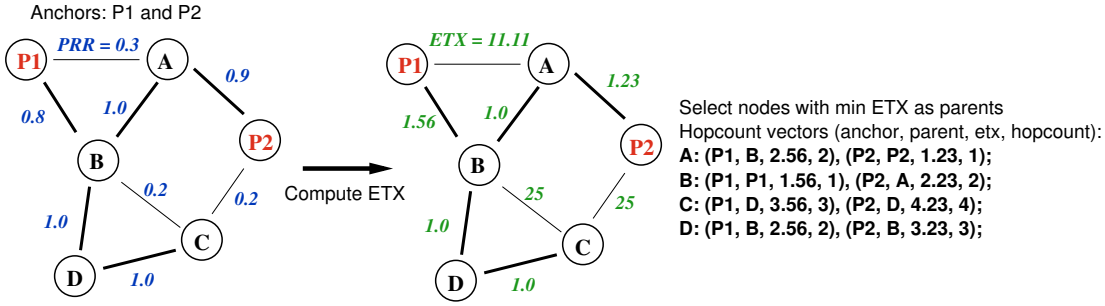


Figure 5.4: Build Hop count Vectors based on Link ETX

The example in Fig. 5.4 shows how the node's hop count vector is obtained, where node P_1 and P_2 are the anchors. Assume all links are symmetric and the figure beside a link refers to the PRR in both directions. To minimize the path ETX, node A selects B as the parent for anchor P_1 and $P2$ as the parent for anchor P_2. The path ETX from A to P_1 is 2.56 with hop count equal to 2. The path from A to P_2 has a hop count of 1 and ETX of 1.23. Therefore, the hop count vector of A is $[(P_1, B, 2.56, 2), (P_2, P_2, 1.23, 1)]$, which can be applied in LCR routing and also be used to calculate PCA coordinates once the anchors' hop count matrix has been propagated.

5.1.3 Anchor Hop count Matrix

To perform dimension reduction, the hop count vectors of all anchors must be propagated to all nodes. These vectors will form a matrix representing the hop distance between each pair of anchors. Besides the conventional *flooding* method, we have also attempted a *hop-by-hop propagation* approach (Algorithm 8)

to advertise the anchors' hop count vectors. In the flooding method, each anchor maintains a timer to periodically broadcast its hop count vector to the neighbors. Upon receiving the update, a node will examine the sender of the message. If the sender is not the parent, the message will be ignored. Otherwise, the node will update the local record and continue the broadcast. The flooding method can dissipate the hop count vector information in a timely manner. However, the chain reactions triggered by flooding may cause severe network contention in dense areas and lead to message loss. Since a node will only accept the copy from the parent and each message is forwarded only once, a weak link on the spanning tree may repeatedly interrupt the message propagation and increase the convergence delay. Due to these drawbacks of flooding, the slower but more reliable hop-by-hop propagation method is used in the current implementation.

Algorithm 8 Hop by Hop Propagation of Anchor Hop count Matrix

```
1: Let max_dim = 7                                   ▷ the maximum number of anchors is 7
2: Let anchor_idx = 0                                ▷ index of anchors
3: Let target_vector = null                          ▷ hop count vector to send
4: Schedule a periodic timer t whose interval is 10 / 60 seconds.
5: if t is fired then
6:     for i = 0 to max_dim − 1 do
7:         anchor_idx = (anchor_idx + 1)%max_dim
8:         hc_vector = anchor_hc_matrix[ancchor_idx]
9:         if hc_vector.anchor_nodeid > 0 then       ▷ the hop count vector entry is valid
10:            target_vector = hc_vector
11:        end if
12:    end for
13:    broadcast target_vector
14: end if
```

The propagation algorithm is given in Algorithm 8. If a hop count vector is sent by the parent, the node will store the received vector in a matrix named `anchor_hc_matrix`. A timer t is scheduled at the interval of 10 seconds during initialization. When t fires, the node will fetch a valid hop count vector entry from the matrix and advertise it to the neighbors. The periodic timer is used to avoid simultaneous packet forwarding triggered by messages from a common parent. Iteratively, it circulates the matrix and broadcasts all entries repeatedly. This will overcome message losses and allow nodes with poor connectivity to capture all information required to compute the PCA coordinates. Once the nodes have obtained the complete `anchor_hc_matrix` and a hop count vector `local_hv`, the PCA coordinates will be calculated as described in Chapter 4. After the PCA protocol converges, the interval of timer t will be raised to 60 seconds to reduce redundant advertisements.

5.1.4 Spanning Tree with Polar Coordinates

The spanning tree for VPCR routing is constructed using Algorithm 9. When a node P discovers a new neighbor N_i, P will insert a new entry into the neighbor list nbr_list. The structure of a neighbor entry is shown in Fig. 5.5. If P is not attached to the spanning tree and the neighbor N_i has joined the tree, P will choose N_i as the parent. If node P is already attached, it will switch to neighbor N_i if N_i provides a path with smaller ETX to the root than its current parent. As the link quality may fluctuate from time to time, a node will switch to a new parent if the path ETX can be reduced by at least 20% in order to

reduce oscillation of the tree topology. The updated parent node ID and the ETX value will be reflected in the beacons and be informed to the nearby neighbors such that the subtree can be updated.

Algorithm 9 Select Parent for the Spanning Tree

1: // Upon receiving a beacon msg from neighbor n, update the neighbor list nbr_list.
2: **for** N_i in nbr_list **do**
3: **if** $N_i.spt\_parent\_nodeid > 0$ **then** ▷ Neighbor N_i is attached to the tree
4: Let p be the incoming PRR from N_i and q be the outgoing PRR to N_i
5: **if** $N_i.spt\_etx + \frac{1}{pq} < spt\_etx$ or $spt\_etx \leq 0$ **then**
6: $spt\_parent\_nodeid = N_i.nodeid$ ▷ use N_i as parent
7: $spt\_hopcount = N_i.spt\_hopcount + 1$
8: $spt\_etx = N_i.spt\_etx + \frac{1}{pq}$
9: **end if**
10: **end if**
11: **end for**

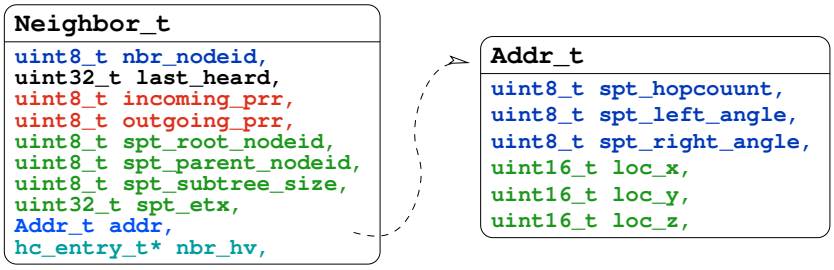


Figure 5.5: Format of the Neighbor Entry and Address Entry

The algorithm to collect the subtree size is given in Algorithm 10. Since each node announces their parent ID in the beacons, a node can determine if it is at the leaf level by overhearing the beacon messages. A leaf node will report back a subtree size of 1 to the parent, once all its neighbors have joined the spanning tree. Subsequently, the intermediate nodes will wait for the report from all children and report the subtree size upwards. The propagation time for the subtree size to reach the root depends on the depth of the spanning tree and the beacon interval.

Algorithm 10 Update the Subtree Size

1: // Upon receiving a beacon message,
2: **for** neighbor N_i in neighbor list nbr_list **do**
3: **if** $N_i.spt\_parent\_nodeid \leq 0$ **then**
4: **stop** ▷ some nodes are not attached
5: **end if**
6: **if** $N_i.spt\_parent\_nodeid = nodeid$ and $N_i.spt\_subtree\_size = 0$ **then**
7: **stop** ▷ some child nodes have not obtained the subtree size yet.
8: **end if**
9: **end for**
10: // all neighbors are attached to the tree and all child nodes have valid subtree tree sizes.
11: **for** neighbor $N_i \in nbr\_list$ **do**
12: **if** $N_i.spt\_parent\_nodeid = nodeid$ **then** ▷ N_i is a child node
13: Let $spt\_subtree\_size$ += $N_i.spt\_subtree\_size$
14: **end if**
15: attach $spt\_subtree\_size$ in the beacon message.
16: **end for**

The root node has the initial polar coordinates and it will compute the polar coordinates for the one

hop neighbors. The angle range in the polar coordinates will be allocated proportional to the subtree size and be recorded in the neighbor list. Upon receiving a neighbor's beacon, if the beacon reveals that the neighbor has not received the assigned polar coordinates, a request message will be sent to inform the neighbor about its coordinates. Once all one hop neighbors will have been updated, they will be in charge of allocating polar coordinates to the nodes two hops away from the root. In a distributed manner, the polar coordinates will be assigned from the root to the leaf level gradually, as summarized in Algorithm 11.

Algorithm 11 Assign Angle Range to Child Nodes

1: Root has $spt\_left\_angel = 0$ and $spt\_right\_angle = 10000$;
2: Initialize spt_angle_mark to spt_left_angle
3: // Upon receiving a beacon from neighbor N_i,
4: **if** $spt\_right\_angle > 0$ and $N_i.spt\_parent\_nodeid = nodeid$ and $N_i.spt\_right\_angle \leq 0$ **then**
5: Let $N_i.spt\_left\_angle = spt\_angle\_mark$
6: Let $range = \frac{N_i.spt\_subtree\_size}{\sum_{n \in nbr\_list} n.spt\_subtree\_size}(spt\_right\_angle - spt\_left\_angle)$
7: Let $N_i.spt\_right\_angle = N_i.spt\_left\_angle + range$
8: $spt\_angle\_mark = N_i.spt\_right\_angle$ ▷ update the free angle range
9: **end if**
10: send polar coordinates to N_i

5.1.5 Centralized Location Service

For geographic routing, a *location service* is required to provide the source node with the destination address. We employ a simple, centralized address lookup service, which stores all addresses at the root and routes the queries and replies using VPCR. We use the VPCR protocol because it provides a reliable point-to-point delivery service for the request and reply messages and the VPCR coordinates are more convenient for debugging the location service records. Once each node has obtained its PCA coordinates and polar coordinates, it will insert a *child* entry containing its own address in the local cache as in Fig. 5.6. Each entry carries an *upload* flag, indicating if the entry has been uploaded to the parent. An address upload timer periodically examines the local cache and reports a new entry to the parent every 2 seconds. This allows all node addresses to be propagated back to the root reliably without causing severe congestion.

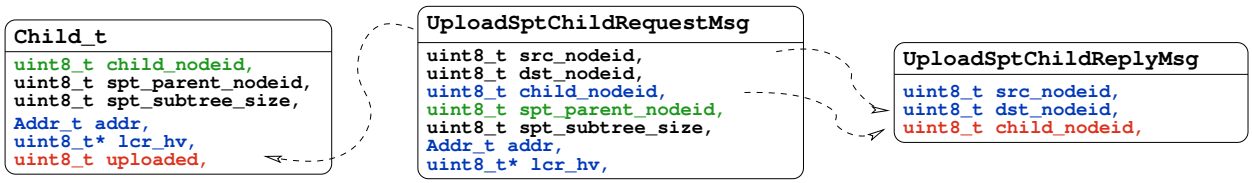
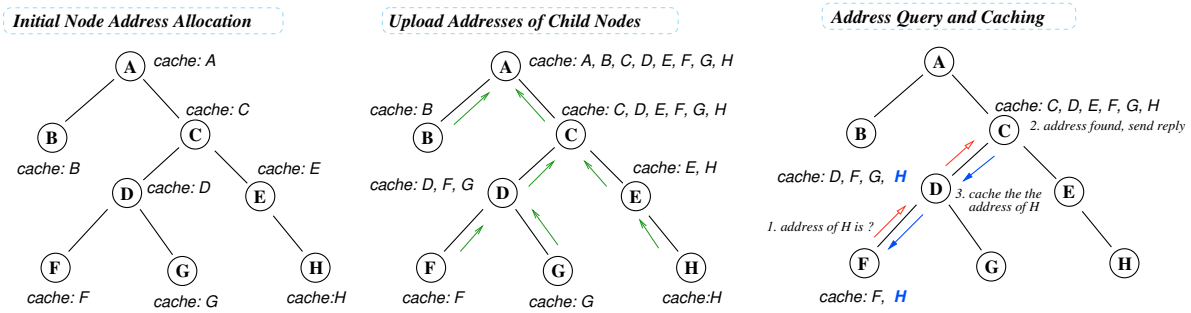


Figure 5.6: Upload Node Address Message

As the addresses are stored in a hierarchical manner, the query for the destination can be resolved at various levels. The lookup process is illustrated in Fig. 5.7. When node F has a data packet for node H, it will first search the local cache for the address of H. If it is not found, F will forward the query to its parent D. The query packet also contains the polar coordinates of F, such that the reply can be sent back using VPCR. As the initial cache of D contains only the addresses of D, F and G, node D will pass

the query further to its parent node C. Node C will use the source address in the query packet to send the reply back to F. When the reply message passes node D before arriving at F, node D will insert this new entry to its cache to accommodate subsequent queries for H. More queries can be resolved at lower levels as more replies are returned. This will effectively reduce the lookup latency for the location service. In a dynamic topology, a expiration interval will be used to eliminate the stale records.



(a) Initial Node Address Cache (b) Upload Subtree Addresses (c) Node Address Query: F to H

Figure 5.7: Management of Address Cache and Node Address Query

5.2 Simulation Results

We first compare the performance of PCA, LCR and VPCR using simulations in TOSSIM. The simulation parameters are selected based on our link measurement results conducted on the Indriya testbed. The Indriya testbed is a wireless sensor network testbed deployed in the COM-1 building of School of Computing at NUS. It contains 127 TelosB nodes installed in 3 floors and uses USB connections for power supply and data logging. It supports TinyOS-2 and uses a web interface ported from MoteLab for job management. We measured the node density, link quality and network topology to choose appropriate values for the simulation parameters, including the beacon interval, the neighbor list size, the buffer size on the radio interface and the placement of the anchor nodes. The link measurement results are presented in Appendix C.

The details of the simulation configuration are shown in Table 5.1. The network topology contains 126 nodes and the link quality data is imported from the link measurement results. The nodes are programmed to send data packets using the following 4 routing algorithms in an interlaced manner: PCA, VPCR, PCA with VPCR and LCR. For each protocol, a node sends 250 data packets to randomly selected destinations at an interval of 1 packet/minute. The data packet size is 51 bytes and each data packet can be retransmitted maximally 7 times with a delay of 10 ms. We manually choose 7 nodes in the network as anchors. As the network topology is three-dimensional, the dimensionality of the PCA coordinates is set to 3 and the LCR protocol selects 3 anchors for routing. The simulation lasts for 20 hours, in which the first hour is the warm up period.

The TOSSIM simulator requires an input network topology with link qualities represented by RSSI. We examine the relation between the packet reception ratio and RSSI in TOSSIM by sending 100 data packets over a link with average RSSI between -103 dBm and -30 dBm. We plot the average delivery ratio at each RSSI level in Fig. 5.8. The input topology is generated by converting the measured PRR

Table 5.1: Configuration of Simulation Parameters

Parameter	Value
Number of Nodes	126
Number Data Packets	250 for each protocol
Data Packet Size	51 Bytes
Data Packet Interval	1 packet/minute
MAC Retransmission	7 times, interval = 10 ms
Address Query Retry	10 times
Routing Protocols	PCA, VPCR, PCA+VPCR, LCR
Number of Anchors	7 (basement: 3, level-1: 1, level-2: 3)
Dimensionality of PCA Coordinates	3
Number of Routing Anchors in LCR	3
Neighbor List Capacity	40
Node Address Cache Size	127, multi-level caching
Simulation Time	20 hours, warm up: 1 hour
Beacon Interval	10 seconds

back to its corresponding RSSI value according to Fig. 5.8.

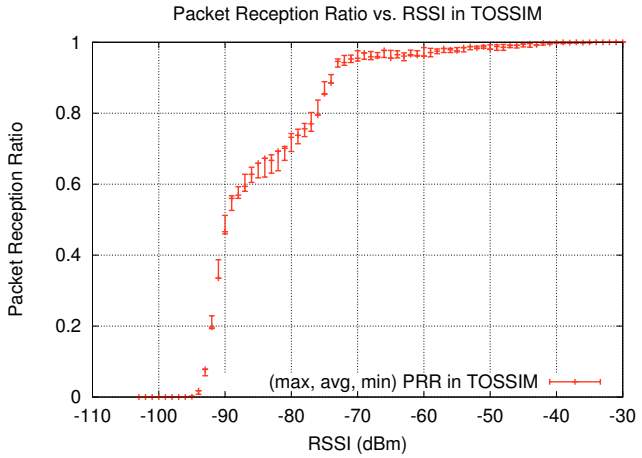


Figure 5.8: Packet Reception Ratio vs. RSSI in TOSSIM, 100 pkts/RSSI, 10 times

5.2.1 Routing Performance of Data Packets

The delivery ratio of the data packets and the address query packets is shown in Fig. 5.9. The address query has an average success ratio of 98.9%, which is computed as the number of queries sent divided by the number of replies received. With maximally 10 times of retry, all queries are resolved successfully. The PCA protocol applies greedy forwarding based on the embedding coordinates and the data packets are dropped at the local minimum nodes, leading to a delivery ratio of 93.1%. In VPCR, every node will be able to identify the next hop for the destination through the polar coordinates. As the simulation uses link qualities measured from the testbed, the delivery ratio drops slightly to 99.7% due to fluctuations in link quality. The VPCR protocol always relies on the spanning tree for routing, resulting in longer paths than LCR and PCA. To achieve high delivery ratio and a short average path length, we combine PCA with VPCR, where greedy forwarding by PCA coordinates is applied until a local minimum is reached and after that, the routing will resort to VPCR. The combined algorithm has a delivery ratio of 98.8%,

comparable to that of VPCR. For connectivity-based routing such as LCR, the number of routing anchors should be larger than the intrinsic dimensionality of the deployment space. However, in order to retain the same routing overhead in the packet as PCA, LCR employs only 3 anchor nodes for routing and obtains the lowest delivery ratio of 60.7%.

In order to investigate the delivery performance of LCR with different number of anchors, we vary the number of anchors used by LCR from 3 to 7 (the maximal number of anchors available) and plot the delivery ratios of LCR in Table 5.2. The result shows that as the number of routing anchors increases, the delivery ratio of LCR slowly grows from 60.7% to 76.0%. The inferior performance of LCR is caused by the discrete nature in the hop count vectors and the limited network size in our testbed. As the hop count distance from the nodes to the anchors are very small (normally less than 5), a slight variation in the hop count value may divert the packets to the wrong direction and cause delivery failure. The PCA algorithm computes the coordinates as floating point values which have finer granularity than discrete values. Each node also applies an iterative method to normalize its coordinates by taking the average of all its neighbors' coordinates, which helps to remove some of the local minimum cases. Therefore, although the PCA coordinates are computed from the hop count vectors, its floating point format and the normalization procedure help it to achieve higher delivery ratio than the hop count vectors.

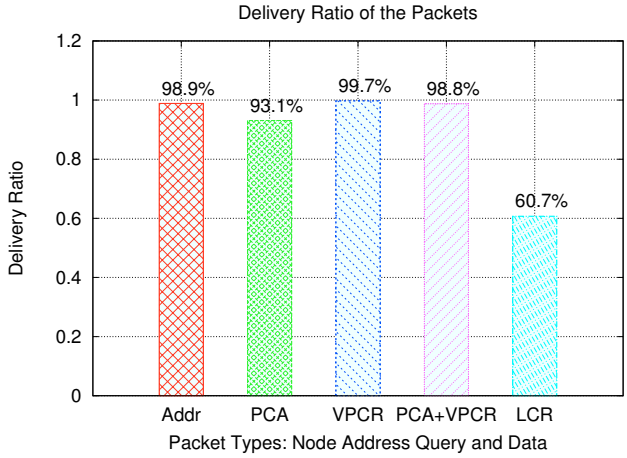
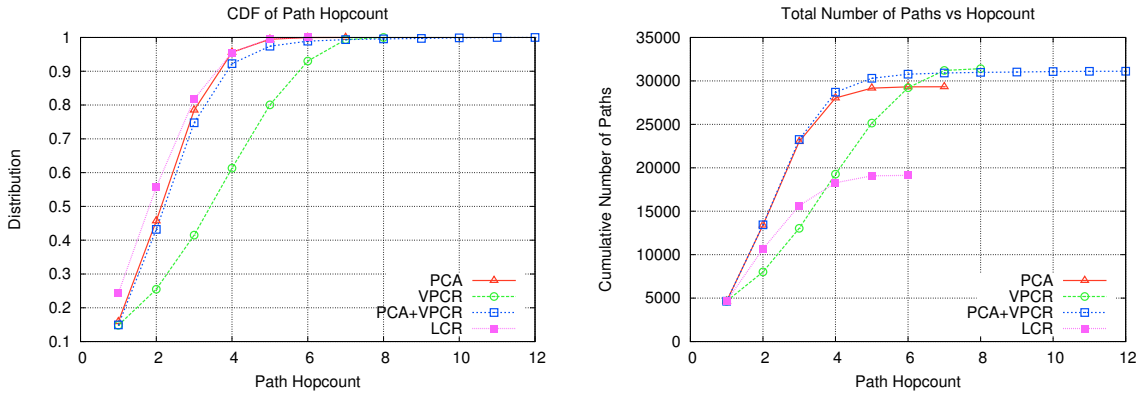


Figure 5.9: Packet Delivery Ratio for Address Queries and Data Packets

Table 5.2: Delivery Ratios of LCR with Various Number of Anchors

Number of Anchors	3	4	5	6	7
Delivery Ratio	60.7%	63.2%	67.2%	75.6%	76.0%

In Fig. 5.10(a), the cumulative distribution of the hop count shows that the average path length of PCA and LCR is smaller than the rest and 95% of the paths are within 4 hops due to the greedy forwarding method employed. The average path length of PCA+VPCR is slightly longer than LCR, although 92.3% of the paths are still ≤ 4 hops long. The VPCR protocol generates the longest routes as packet forwarding follows the spanning tree in Fig. 5.11, even when shorter paths are available. Only 61.3% of the paths found by VPCR are within 4 hops and the average is 3.84 hops, 58% longer than that of LCR and 40% longer than that of PCA and PCA+VPCR.



(a) CDF of Path Hop count

(b) Packet Count vs. Hop count

Figure 5.10: Path Hop Count of the Data Packets

Fig. 5.10(b) shows the number of the paths at the corresponding length. PCA+VPCR and VPCR find more paths than PCA and LCR because the polar coordinates will not create local minimum cases that may appear in the PCA coordinates and the hop count vectors. With the assistance of greedy forwarding, the PCA+VPCR protocol generally returns shorter paths than VPCR, although both protocols return nearly equal number of paths. Despite the fact that LCR achieves the lowest average path length, it finds 39.1% less routes than the tree-based routing algorithms, with the same control overhead. In general, the PCA+VPCR protocol successfully retains high delivery ratio and low average path length by complementing greedy forwarding with polar coordinate routing and outperforms the rest.

The delivery latency of the data packets is given in Fig. 5.12. The distribution of packet delay in Fig. 5.12(a) reveals that all the four protocols achieved similar delivery latency, where the delay of LCR and VPCR is slightly lower than the rest. This is because LCR finds limited number of short routes with less hops to traverse, while VPCR uses entirely the reliable spanning tree links, resulting in less retransmissions. Around 99.5% of packets are delivered within 50 ms for all protocols. Even on the routes longer than 10 hops, the delivery delay is less than 140 ms. Fig. 5.12(b) gives the number of packets received within each interval. Note the log-scale on the Y axis. The number of packets delivered in each interval decreases exponentially as the delay increases. Applying greedy forwarding does not introduce as much benefit to the delivery delay as it does to the hop count. This is because greedy forwarding may risk using unreliable shortcut links to minimize the distance to the destination at each hop, which may cause more frequent MAC layer retransmissions compared to tree-based routing and subsequently increases the routing latency.

5.2.2 Performance of Location Service

The address record consists of the PCA coordinates, the hop count vector for LCR and the polar coordinates for VPCR. A node will report its coordinates to the root in order to initialize the location service. In Fig. 5.13, the hop count value represents the total number of hops travelled by the node address request and reply packets.

When the destination address is not found in the local cache, a node will send an address query to

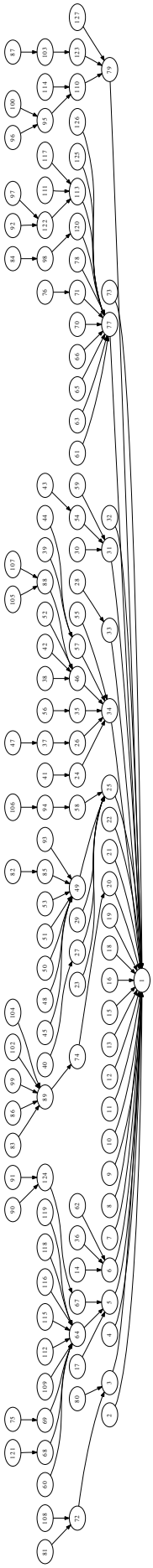
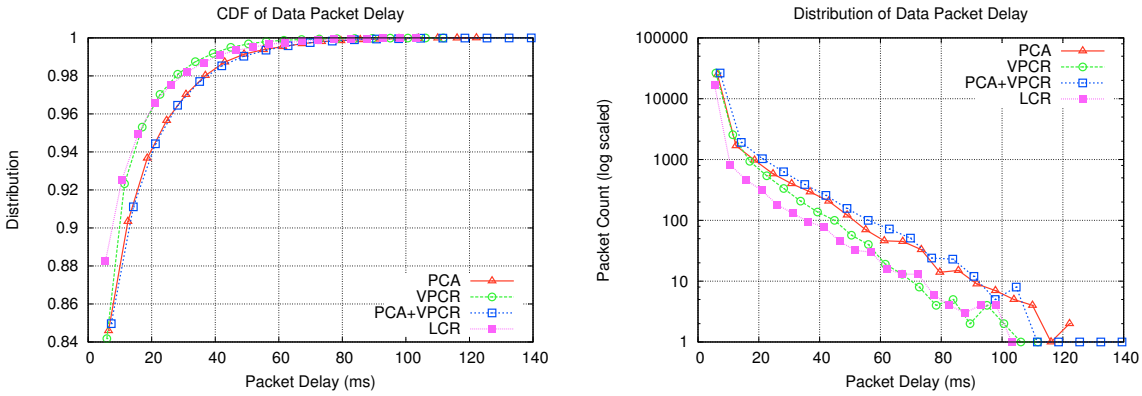
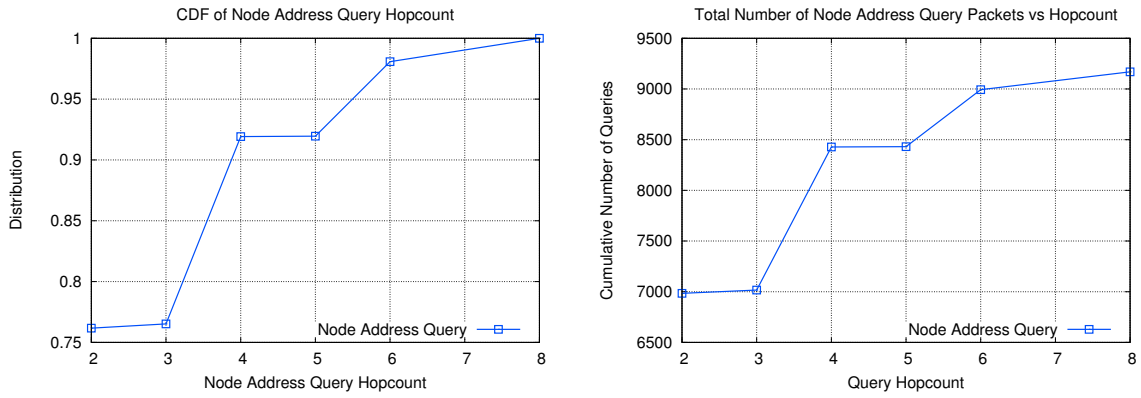


Figure 5.11: The Spanning Tree Topology built from ETX in Simulation



(a) CDF of Packet Delay (b) Packet Count vs. Delay

Figure 5.12: Delay of the Data Packets

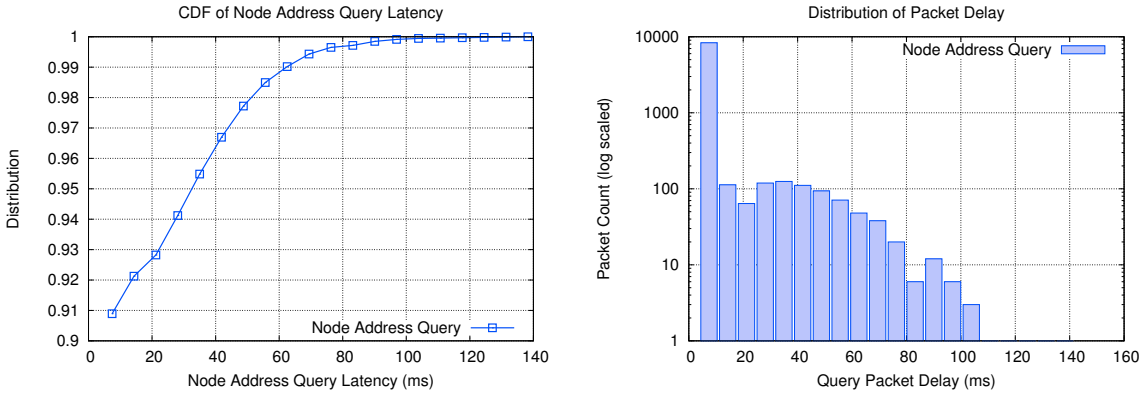


(a) CDF of Path Hop count (b) Packet Count vs. Hop count

Figure 5.13: Path Hop count of the Address Query Packets

its parent. The query is handled in a manner similar to a recursive DNS query, where it is propagated towards the root until the address can be returned. The reply is sent back using VPCR based on the source address attached in the query. The returned address is recorded at each hop on the return path to handle future queries for the same destination.

The spanning tree used for routing the query packets has a maximum depth of 4 hops. In Fig. 5.13(a), the total hop count of the address query and reply packets varies between 2 to 8 hops. With address caching, over 76% of the queries can be answered directly by the parent nodes and 92% of the queries can be processed by the predecessor nodes within 2 hops.



(a) CDF of Address Resolution Delay

(b) Packet Count vs. Delay

Figure 5.14: Delay of the Address Query Packets

The delay of the address query packets is given in Fig. 5.14, where the maximum query latency is 138 ms and 92% queries are answered within 20 ms. In Fig. 5.14(b), as delay increases, the number of query packets in the delay interval decreases exponentially.

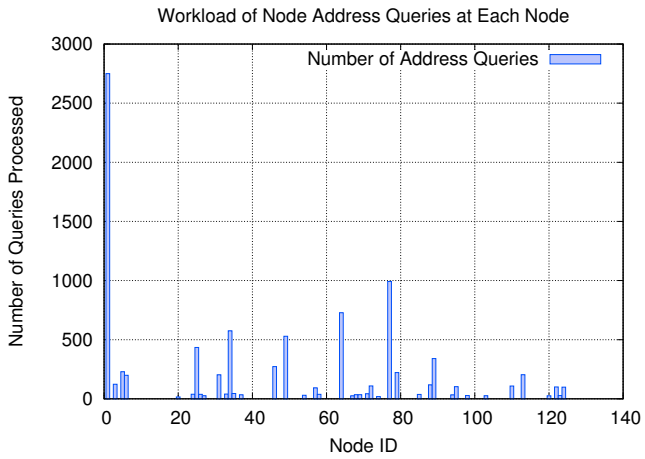


Figure 5.15: Load of Node Address Queries at Each Node

To examine the effectiveness of multi-level address caching, we plot the number of address queries handled by each node in Fig. 5.15. The 9169 address queries are resolved by 41 nodes. Table 5.3 reveals the top 6 nodes that have processed the most number of queries. The root node (node 1) took the highest

load by replying 30% of the queries. Node 77, 64 and 49 handled 10.8%, 7.9% and 6.3% of the queries respectively. The figures indicate that the address caching method can successfully divert the load away from the root and reduce the hot spot effects caused by the query traffic, as 70% of the address queries are resolved by nodes at lower levels.

Table 5.3: Ranking of Address Resolution Workload. Total: 9169 queries

Rank	Node ID	Number of Queries	Ratio	Subtree Size	Level
1	1	2750	30%	126	1
2	77	992	10.8%	20	2
3	64	728	7.9%	12	3
4	34	575	6.3%	19	2
5	49	529	5.8%	9	3
6	25	434	4.7%	17	2

5.3 Experimental Results

As the simulation uses a snapshot of the network topology collected from the testbed, it cannot reflect how the protocols will perform under the temporal variations of the link quality. In order to examine the protocols under real network conditions, we conduct experiments on the Indriya testbed and log the controls messages and data packets through the USB connection for performance analysis. The configuration of the parameters are listed in Table 5.4. Due to hardware issues, the protocols fail to converge on node 76, 101 and 126. Node 1 is the root for VPCR routing and the location service. Every node sends 50 data packets to randomly selected destinations by each protocol. The packet interval is set to 60 seconds to reduce inter-flow interference. We manually select 7 nodes as the anchors to build PCA coordinates. Three of them are used as the routing anchors in LCR, one from each floor in the COM-1 building. All packets are transmitted with the maximum power of 0 dBm. Due to the high node density, the beacon interval is set to 10 seconds to prevent congestion from overhead traffic. A stale neighbor will be removed from the neighbor list if its beacons are not heard for 10 consecutive intervals. The experiment lasts for 6 hours and the first one hour is the initialization period to compute the PCA coordinates and polar coordinates. The program image has a code memory size of 47.8 KBytes and a data memory size of 9.2 KBytes, complying with the memory constraints.

5.3.1 Routing Performance of Data Packets

To avoid the interference from the campus wireless network, experiments were conducted during the night time from 12:00am to 6:00am. The experiments were repeated 3 times, where the performance results show very low variation. We present one set of the results in the following sections for comparison.

In Fig. 5.16, the packet delivery ratios of the spanning tree-based protocols are very close to the simulation results, where VPCR and PCA+VPCR deliver around 95% of the packets. The PCA protocol delivers 86.6% of the packets, where delivery failures mainly result from the local minimum cases (7%) and link losses (6.4%). The LCR protocol achieves the lowest delivery ratio of 49.6% due to insufficient number of routing anchors, as a comprise to the control overhead in the packet header.

Table 5.4: Configuration of Parameters used in Experiments

Parameter	Value
Number of Nodes	124 (failed nodes: 76, 101, 126)
Number of Data Packets	50 for each protocol
Data Packet Size	51 bytes
Packet Interval	60 seconds
Routing Protocols	PCA, VPCR, PCA+VPCR, LCR
Anchors for PCA Coordinates	7 nodes: 15, 31; 52, 58, 72; 107, 113
Dimensionality of PCA Coordinates	3
Routing Anchors for LCR	3 nodes: 15, 58, 113
Neighbor List Capacity	40
Address Cache Size	127
Radio/Serial Buffer Size	3 packets
Beacon Interval	10 seconds, timeout: 10 intervals
Default Transmission Power	max: level 31 (0 dBm)
Experiment Time	6 hours, warm up: 1 hour
Image Size	Program: 47.8 KBytes, Data: 9.2 KBytes

The node address query and reply packets are routed by VPCR and the success ratio for the node address queries is 95.3%. Given the maximum number of retries of 10 times, all queries are resolved correctly.

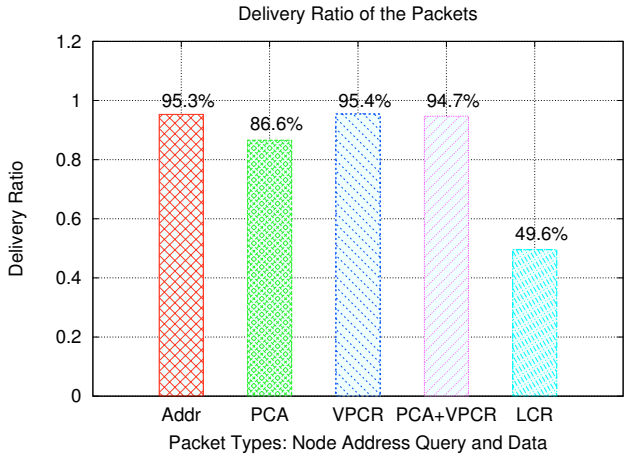
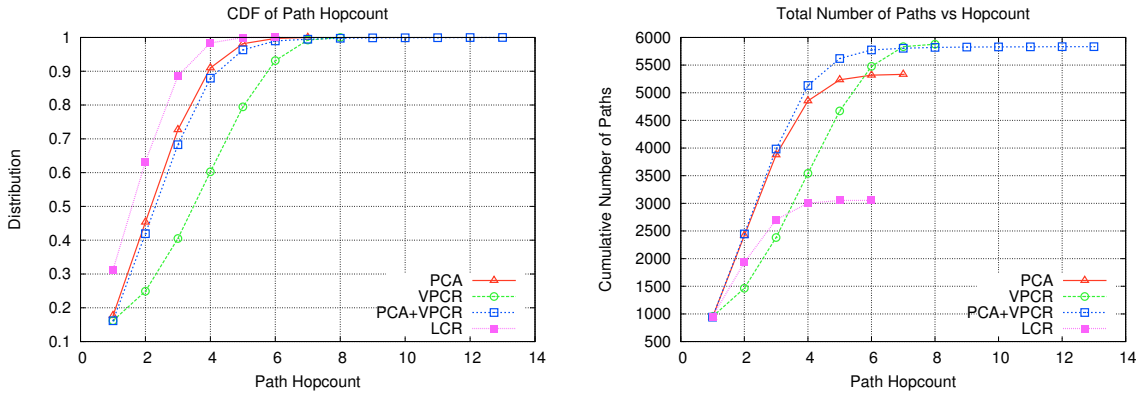


Figure 5.16: Packet Delivery Ratio in the Testbed

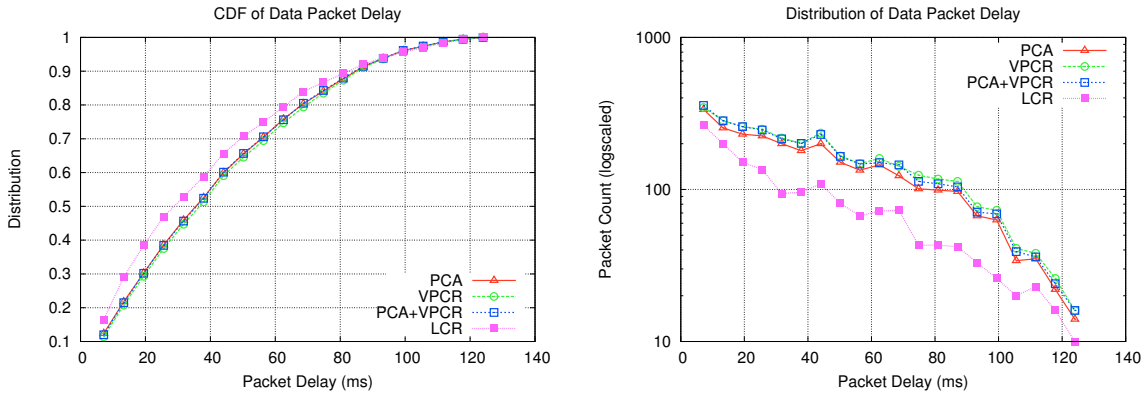
The average path length of the protocols is plotted in Fig. 5.17. The LCR protocol has the lowest hop count, as it discovers only 50% of the paths, 98% of which are within 4 hops. The PCA and PCA+VPCR protocols achieve nearly identical hop count distribution. The average path length for PCA is 2.75 hops and the average path length for PCA+VPCR is 2.94 hops. To escape from the local minimum spots in the PCA coordinates, the PCA+VPCR protocol falls back to the spanning tree, generating the longest path of 13 hops. Although the PCA+VPCR protocol has the largest maximum path length, over 96% of the its paths are ≤ 5 hops, which makes it generally more efficient than VPCR. The spanning tree used for VPCR routing and node address resolution is given in Fig. 5.19, with a depth of 5 hops. The maximum VPCR path length is 8 hops and the average path length is 3.86 hops, 40% longer than PCA and 31% longer than the paths returned by PCA+VPCR.



(a) CDF of Data Path Length (b) Packet Count vs. Hop count

Figure 5.17: Path Hop count of Data Packets

In Fig. 5.18, all protocols exhibit very similar delivery latency for the data packets, despite the variation in the hop count. The difference in the packet delay distribution for PCA, VPCR and PCA+VPCR is negligible. As LCR tends to find shorter but fewer paths, the delivery delay for LCR is slightly lower than the rest. The percentage of data packets delivered within 50 ms is 71% for LCR and 65% for the rest protocols. The average delivery latency for LCR is 39.6 ms, while the rest protocols retain an average of 44.7 ms. The packet count in Fig. 5.18(b) shows that PCA delivers fewer packets compared to VPCR at all delay intervals, as the links attempted by the greedy forwarding step in PCA are more likely to degrade the delivery ratio than the tree links selected by ETX.



(a) CDF of Data Packet Delay (b) Packet Count vs. Delay

Figure 5.18: Delay of Data Packets

5.3.2 Performance of Location Service

The depth of the spanning tree is 5 hops; thus, the address query and reply packets travel maximally 10 hops, While each node selects 50 random destinations to send data packets, the total number of node address queries is 3779. The ratios of queries replied by predecessor nodes 1, 2 and 3 hops away are 59.4%, 26.6% and 11.5%.

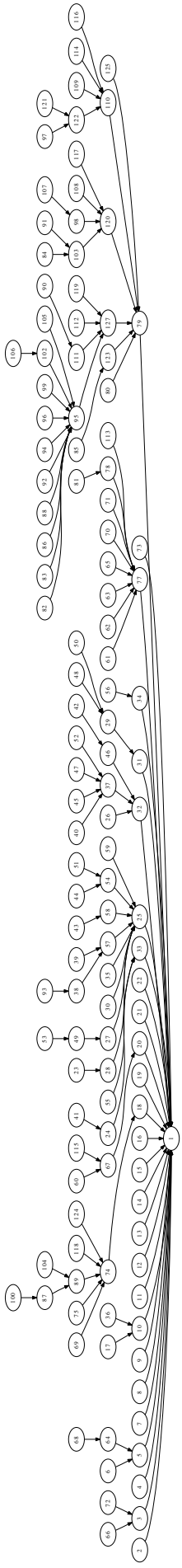
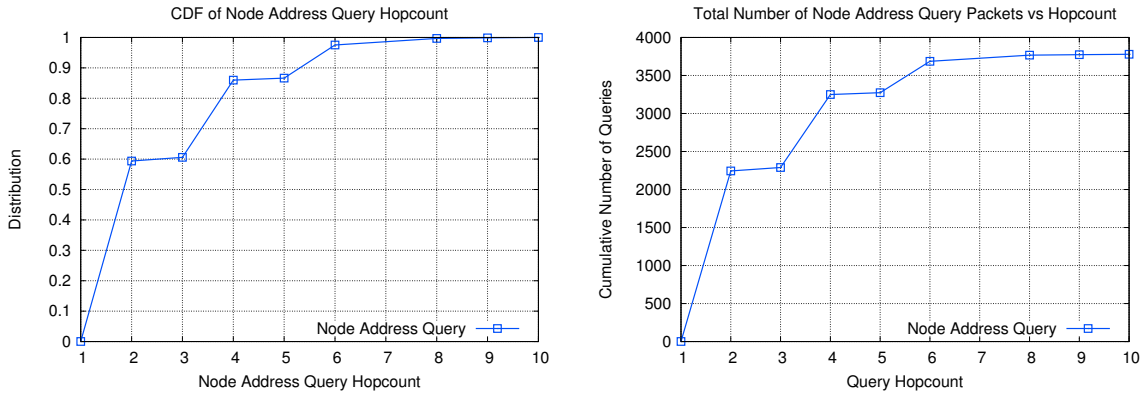


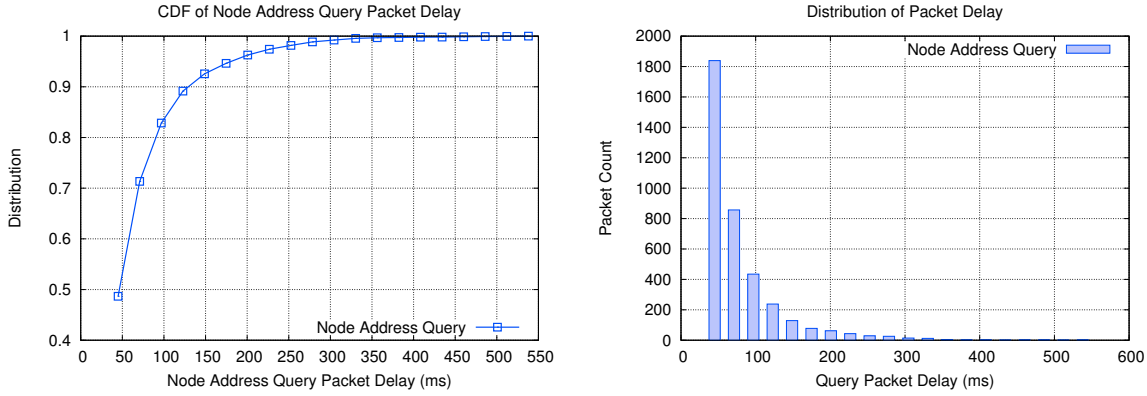
Figure 5.19: The Spanning Tree Topology for Location Service and VPCR in the Indriya Testbed



(a) CDF of Address Path Length (b) Packet Count vs. Hop count

Figure 5.20: Hop count of Address Query Packets

In Fig. 5.21(a), the maximum latency of an address query is 538 ms and the average is 79.4 ms. The maximum address resolution delay is longer than the maximum data packet delay, as it consists of the time for the address query to reach the replier and the time for the reply to return. The 98-percentile of the address resolution delay is 250 ms, slightly over twice of the 98-percentile of data delivery delay, which is 110 ms. The distribution of the query latency is exponential as revealed by Fig. 5.21(b), similar to the delay distribution of the data packets.



(a) CDF of Address Query Packet Delay (b) Packet Count vs. Delay

Figure 5.21: Delay of Node Address Query Packets

Fig. 5.22 shows the number of address queries handled by each node. Out of the 40 non-leaf nodes involved in address resolution, the root node shares 43% of the workload, higher than the simulation result. This is because each node in the experiment only sends one fifth of the data packets compared to those sent in the simulation. The number of address queries generated is only 3779, significantly less than the 9169 queries in simulation. The node address records have not yet been fully populated down the tree, causing more queries to be sent to the root. The top 6 nodes with the highest query workload are listed in Table 5.5, indicating that the address resolution workload is generally proportional to the subtree size before the address cache converges.

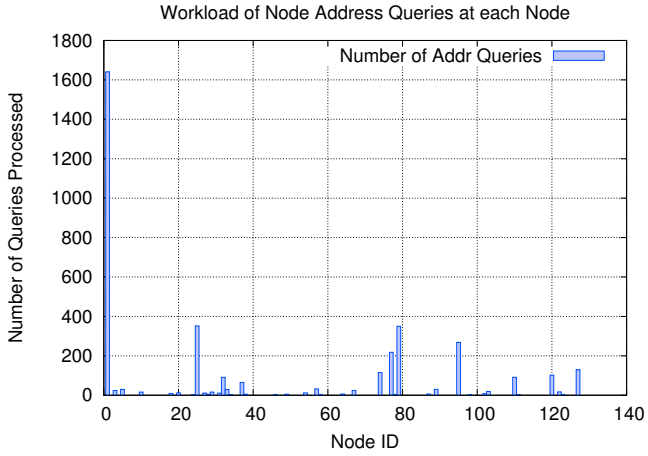


Figure 5.22: Workload of Node Address Query at Each Node

Table 5.5: Ranking of Address Resolution Workload, total: 3779 queries

Rank	Node ID	Number of Queries	Subtree Size (rank)	Level
1	1	1641, 43.4%	124 (1)	1
2	25	352, 9.3%	18 (3)	2
3	79	350, 9.2%	37 (2)	2
4	95	268, 7.1%	12 (5)	4
5	77	218, 5.8%	10 (6)	2
6	127	130, 3.4%	17 (4)	3

5.4 Summary

Through the link quality measurements and the experiments, we have identified the following issues that should be addressed for employing a geographic routing protocol in a real network deployment.

- Due to the limit on memory size and memory access mechanism, the routing algorithm should be implemented with one-dimensional data structures, e.g. array and list. The number of functions and temporary variables is determined by the flash memory size and the global variables are limited by RAM. The floating point value must be packed into an integer variable before being transmitted in a message.

- Storing the mapping between node ID and node address in the memory is only applicable when the memory size is large enough for all nodes deployed. If the network size is overwhelming, a gateway node with dedicated storage space for location service should be installed. Resorting to the data flash for extra storage space is another alternative, although it will increase the lookup delay and energy consumption.

- In an indoor deployment, the network density may have very large variations at different locations. When the neighbor list is not large enough to hold all neighbors, a blacklisting method should be applied to preserve the good links.

- On some testbeds, it is a feasible approach to use the reception ratio of broadcast beacons to estimate the reception ratio of unicast data packets on the link. However, beacon message intervals

should be selected carefully to achieve a balance between the control message overhead and the agility to reflect link quality fluctuations. An adaptive beaconing mechanism used in the Collection Tree Protocol can be adopted to adjust the beacon interval based on the link status.

- When a message needs to be propagated to the entire network, the *hop-by-hop propagation* method exhibits very high reliability compared to a simple *flooding*. The flooding method is vulnerable to packet drops due to collisions in dense areas and it may fail to reach the weakly connected nodes as a result of link losses. Nevertheless, hop-by-hop propagation may introduce longer delay depending on the propagation interval of each hop and the network diameter.

- When sufficient memory is available, it is applicable to employ a distributed design for each node to compute its coordinates independently. This will eliminate the risk of a single point of failure and provide higher resilience to network dynamics. For a large-scale indoor network deployment, where a distributed localization method is not feasible, the geographic coordinates can be computed by a centralized gateway. It will considerably reduce the memory consumption and complexity of the program on the individual nodes. This offline approach also allows fine-tuning of the coordinates before they are assigned to the nodes.

- Even if most links in a network have high delivery ratios, packet losses may still occur due to collision, fluctuation in the link quality or interference from the environment. To provide more reliable data delivery, the retransmission mechanism should be enabled at the MAC layer or be implemented at the application layer. In order to avoid duplicates caused by the MAC layer retransmissions, each unicast message should carry a unique identification tag.

- A packet buffer should be provided to handle bursty traffic, such as the chain reactions triggered by a control message. For point-to-point delivery, an ETX-based spanning tree with polar coordinates can obtain very high reliability. The greedy forwarding method can be integrated into tree-based routing to avoid the hop-spot effects and reduce the average path length. However, the hybrid method may risk using poor links in order to minimize the distance to the destination, leading to more packet losses.

Chapter 6

Conclusion

Besides the basic converge-cast service for collecting data, many sensor network applications are implemented on the more generic point-to-point communication service, such as data-centric storage, transmission control and in-network query processing. The ability provided by the point-to-point routing protocols to exchange information between any pair of connected nodes will diversify the design of sensor applications and greatly improve their usability.

As sensor network deployments spread from outdoor to indoor, the sensor nodes can be placed on multiple floors inside a building, creating a three-dimensional topology. Therefore, an adaptive point-to-point routing protocol should be able to provide reliable service in both 2D and 3D network scenarios.

Due to the delay and network overhead in the path discovery procedures, the conventional ad hoc routing protocols are not feasible for the current sensor platforms. Geographic routing protocols provide a plausible alternative for point-to-point communication in wireless sensor networks. Geographic routing discards the usage of the routing table by identifying each node through its coordinates. It obviates the requirement for path discovery by forwarding packets based on the destination address in a localized manner.

The tree-based routing protocols provide a promising option, as the tree structure can be built efficiently in both 2D and 3D networks. The coordinates of the nodes at each level provide a description of the subtree structure, which can be used to locate a unique path towards the destination. However, the existing tree-based routing protocols are often designed for 2D networks and the performance in 3D networks is not investigated extensively. The position information is assumed to be available through localization methods, while the effects of location errors are often ignored. The routing efficiency is provided by the shortcut links in the tree, which is hard to utilize due to the imbalanced tree structure, leading to long routing paths.

Greedy forwarding can be integrated into tree-based protocols to reduce the path length. For many indoor deployments, the actual node positions are not always known. In this case, hop count vector-based routing offers a practical option for greedy forwarding, as hop count vectors can be learnt automatically and be populated in a timely manner. Since the delivery ratio heavily depends on the number of beacons available, it generally requires higher control overhead to achieve better routing performance. Without an effective measure to reduce the routing cost, the trade-off between performance and overhead will prevent hop count vector-based routing from being applied to large-scale networks.

In this thesis, we propose the Spherical Coordinate Routing protocol (SCR) to evaluate the performance of tree-based routing in 3D networks. We also apply the Principal Component Analysis-based dimension reduction algorithm (PCA) to hop count vector-based routing, such that greedy forwarding can be performed at lower cost when combined with tree-based routing for better routing efficiency. As geographic routing requires a location service to map the node IDs to the coordinates, the maintenance cost and performance of the location service will significantly affect the applicability of the routing protocols. We implemented a location service based on the VPCR protocol to show that the reliable location service can be provided at a relatively low cost.

6.1 Summary of Contributions

To provide point-to-point routing in 3D networks, we designed the SCR protocol based on VPCR. It applies a 3D variant of NoGeo to localize the network and allocates the spherical coordinates based on the location and subtree size of child nodes. Instead of using the shortcut links between sibling nodes, the SCR protocol applies greedy forwarding based on the hop count vectors built at the localization step, which can significantly reduce the average path length against irregularities in an unbalanced tree topology. We compare the performance of SCR with BVR, LCR and VPCR by simulations in 3D network scenarios. The SCR protocol achieves lower hop stretch than VPCR as a result of the effective greedy forwarding procedure. The paths returned by SCR are slightly longer than those found by LCR; however, SCR can successfully identify the routes for all node pairs without resorting to flooding. The high delivery ratio and low hop stretch performance makes SCR a more favorable protocol for 3D networks.

To further reduce the cost of greedy forwarding in SCR, we apply the PCA algorithm to hop count vector-based routing. The PCA algorithm collects the hop count vectors from the beacon nodes through hop-by-hop propagation. To maintain a stable topology, the propagation paths are selected based bi-directional link qualities instead of the hop count metric. In the current distributed design, each node independently applies the algorithm to convert the hop count vector to low-dimensional Euclidean coordinates. This conversion allows packet forwarding to be done at much lower cost without sacrificing the routing performance significantly. The PCA algorithm is implemented in TinyOS and we conducted both simulations in TOSSIM and experiments on the Indriya testbed with 127 TelosB nodes for performance evaluation. The simulation and experimental results show that the PCA algorithm can achieve higher delivery ratio, lower hop stretch and shorter flooding range than the hop count vectors using the same packet overhead, which makes it a promising approach to improve greedy forwarding in a real sensor network.

In summary, for point-to-point data delivery in 3D networks, a tree-based routing protocol can achieve very high delivery ratio with longer average path length compared to greedy-based protocols. The hop count vector-based greedy forwarding method can be integrated into the tree-based protocols to reduce the path length. However, the delivery ratio may be degraded when greedy forwarding attempts shorter but less reliable links. Since the performance of the hop count vector-based algorithms largely rely on the selection of anchors, more anchors are preferred in order to improve the success ratio of the greedy procedure, which creates a trade-off between packet overhead and delivery performance. Dimension reduction methods, such as PCA, can be applied to retain the performance while reducing the

packet overhead for better communication efficiency; however, they may also impose higher computational cost and memory requirement on the sensor nodes. For a small-scale network with around 100 nodes, it is possible to implement a centralized location service on the current platform (e.g. TelosB). The node address queries can be resolved reliably (100% in our experiments), when the link-layer and application-layer retransmission schemes are enabled.

6.2 Future Work

The work introduced in this thesis leaves the following issues for future exploration.

- The tree-based routing protocols ensure the packet delivery by aggregating the location information of subtrees at the parent nodes, such that the routing path can be identified from the subtree information. This hierarchical structure can be expensive to maintain under network dynamics, as changes in the subtrees must be updated in the parent levels. It will be desirable to create a more efficient coordinate system, where the insertion or removal of child nodes will impose negligible effects to the coordinates of the parent and the packet delivery is still guaranteed for any two connected nodes without flooding.

- The current PCA-based dimension reduction scheme can reduce the packet overhead; however, it requires $O(k^2)$ memory for computation, where k is the number of beacons. For a large-scale network, this cost prevents PCA from being adopted in a distributed design. Other dimension reduction methods should be studied, in order to reduce the packet overhead at lower temporary memory demand.

- In the current implementation, the link quality is estimated from the reception of beacon packets sent at static intervals. To remain responsive to link changes and reduce the communication overhead, adaptive beacon intervals should be applied based on the density and temporal changes in the network. The existing data traffic should also be exploited to give a better estimation.

- In the experiments, the neighbor table size was set large enough to handle all entries based on the link measurement results. Without the prior knowledge, the neighborhood size may actually exceed the capacity of the neighbor table. A blacklisting method should be incorporated to select the reliable link adaptively without causing network segmentation.

- Our work does not consider the effects of duty-cycling, a common energy-saving technique for battery-powered sensor networks. When the nodes are not always in the active mode, the temporal effects may change the design of route selection, buffer management and retransmission algorithms.

Appendix A

Sensor Network Applications

We summarize below 7 categories of sensor applications to highlight some popular areas where the wireless sensor networks are currently applied.

A. *Ecosystem monitoring*: Wireless sensor networks are extensively exploited in the ecosystem monitoring projects, where continuous data collection is required in remote [88] or hostile environments [89] and frequent human intervention may disturb the natural behavior of the targets [90]. The Great Duck Island project [1, 15] deployed 32 sensors to the nest burrows of seabirds to observe the nesting behavior from fluctuations in temperature. The self-organized sensor nodes form a spanning tree to report data back to solar-powered gateways which in turn transmit the data to back end servers through a satellite link. Barrenetxea *et al.* built a network with 16 stations [91] on the 2500 m high Génépi peak at the Swiss Alps. The stations provided temperature, humidity and rainfall data on the rock glacier for two months, which was used to evaluate the possibility of avalanche and land slide. Hefeeda *et al.* modeled the early forest fire detection [92] as a k-coverage problem. By measuring the temperature, relative humidity, wind and rain statistics, they compute the probability of ignition and fire intensity through the Fire Weather Index system. Santoni *et al.* conducted real fire sensor experiments by protecting the sensor nodes with the thermal insulation layer named Firesensorsock [93]. This allows the sensor nodes to remain functioning during the spreading of fire, such that the spatial and temporal movement of the fire zone can be predicted.

Batalin *et al.* combined the autonomously articulated sensor nodes with the static nodes and built a 3D mobile sensor network [94]. It measures the solar radiation, temperature, and chemical composition in the atmosphere at the tree canopy. Upon sensing a variation in the readings, the static nodes will call the mobile nodes for a close up examination of the phenomenon, which effectively provides better spatial and temporal coverage at a lower cost. A similar system is deployed by Tolle *et al.* [95] to observe the micro-climate at different altitude in a forest. Vasilescu *et al.* experimented an underwater sensor network [96], where static nodes are placed under the sea to collect the water temperature and pressure data, and mobile nodes are used for data muling and network maintenance. The acoustic signal is used for node localization and synchronization, while optical transmission is used for point to point communication to relay data from the static nodes to the mobile nodes. Instead of using static deployment, SensorFlock [97] is designed to actively trace and detect the target event in an open air space through micro aerial vehicles. The airborne sensor network can be remotely controlled to follow the phenomenon such as a

toxic plume, a spreading wild fire or a storm to gather real time information. Some other environmental monitoring projects include weather monitoring [98, 99], early warning flood detection [100], acoustic noise monitoring at the airport [101], and estimation of the canopy coverage [102, 103].

B. *Intrusion detection and target tracking*: Arora *et al.* created a target classification and tracking system [2] which uses the magnetometer, infrared sensor and radar sensor to detect the existence of a person, a soldier and a vehicle and perform target classification based on the speed, bearing and magnetic signature of the target. The network focuses on a collaborative sensing and data processing design, where the collected data is processed at a central classifier to counter the side effects of noise and hardware errors. Simon *et al.* proposed a counter-sniper system [104] using a group sensors to track the acoustic signal generated from a gun fire. The shooter's position can be computing by comparing the arrival time of the signal with the position of the sensors. Gu *et al.* proposed a hierarchical detection and classification system [105], where the target classification procedure is performed hierarchically at the sensor node, group and base levels in order to reduce the data traffic in the network. Each node computes a confidence vector based on the sensor readings. The confidence vectors are reported back to the cluster head and the data sink for further analysis, such that higher level information can be obtained such as the number and the speed of intruders. The anti-theft car park system [106] constructs a spanning tree from the cars, while each car is equipped with a communication sensor with a unique identity. As each node periodically reports a beacon back to the data sink, a car theft event is detected from a timeout. Cyclops [107] is designed to be a low power image sensor that can be used in a video surveillance sensor network for detecting intruding objects. Tso *et al.* presented the 3G robot surveillance system [108], which allows the user to control the patrol robot through a cellphone. The static sensor nodes report the observed phenomenon to the gateway which will notify the user through SMS. The user can use the cellphone to dispatch the robot to retrieve visual information.

C. *Structural health monitoring*: Kim *et al.* [109] deployed 64 nodes in a span of 4200 feet on the Golden Gate Bridge to detect ambient vibration and strong motion of the bridge structure. All nodes form a spanning tree based on the measured link qualities and each node samples the accelerometer at an interval of 5 ms. Over a duration of 12 hours, the data sink pulled data from each node one by one. The seismic signal is examined to evaluate the time scale and severity of the motions. The Wisden [110] uses the accelerometer to monitor the vibration of the buildings due to wind or an earthquake. It performs wavelet-based compression to convert the sensor readings to byte stream and rely on hop-by-hop and point-to-point recovery to ensure reliable data delivery to the sink. By subjecting the sum of forwarding delay from the arrival time, the event occurrence time can be derived without clock synchronization. Krishnamurthy *et al.* [111] built a industrial sensor network with sensors installed on the equipments to predict the hardware failures. A gateway is used to coordinate the schedule and collect vibration signature from nodes through direct communication. The Nonintrusive Autonomous Water Monitoring System (NAWMS) [112] estimates the water flow rate and computes the water usage from vibration of the water pipes, which will be exploited for water conservation. Nasipuri *et al.* [113] deployed a sensor network at the substation to monitor the temperature of the circuit breakers and transformers, which eliminates the hazards of on site data collection. Allen *et al.* proposed a volcanic earthquake localization system [114] to detect and localize the earthquake from the arrival time of the seismic and acoustic signals at various sensors.

D. *Road mapping*: The BikeNet [115] is a mobile sensing system installed on bikes to collect road condition information and the rider's performance data. Each bike is equipped with multiple sensors to record the wheel speed, paddling speed, inclination angle, traveling distance, etc. The environment factors such as the noise level, air pollution and traffic condition are also monitored by the chemical, acoustic sensors and magnetometers. The cellphone carried by each rider acts as a gateway to forward the local data to the Service Access Points (SAP) along the routes. The bikes can also be configured as data mules to relay data for other members encountered. The collected data is analyzed offline to evaluate quality of the routes and refine the athletics training program. The Nericell [5] system is designed to record the noise condition and traffic condition for mobile vehicles during a ride. Nericell exploits the commodity sensors readily available on the cellphones, such as the accelerometer, microphone and the GPS. It provides a reorientation algorithm to calibrate the placement of the cellphone. The accelerometer will record the bumpy road sections and the GPS device can identify the exact location. The average driving speed is used to derive the traffic condition. The microphone records the environmental noise and applies a filtering algorithm to differentiate human voice from the honk of vehicles. The traffic condition collected from the various routes can serve as the input for route planning, such that the chaotic roads can be avoided.

E. *Body sensor network*: Mercury [4] is a wearable sensor network platform to evaluate the motion function of the patient and progression of the disease. A patient wears 8 accelerometer sensors on the limb segments, such that the body movements can be constantly monitored. All sensor nodes form a one-hop collection tree with the data sink installed in the ward and periodically report the motion data. Upon detecting suspicious movement pattern, the data sink can trigger the gyroscope sensor for detailed examination. The CenWits [116] is a search and rescue system that tracks the location of the hikers through the witness information. Each person carries a wireless sensor, which exchanges the ID information with neighboring sensors. The records are tagged with the time stamp of the encounter. Each sensor also acts as a data mule, such that the information can be duplicated to increase the chances of delivery. When a target is reported missing, all witness information can be retrieved to sketch the search area. The CenceMe application [117] running on the smart phones estimate the type of human activities and the occasion based on the embedded microphone, accelerometer and GPS. It collects sound samples from the microphone and perform discrete Fourier transformation to classify the source as a conversation or ambient noise. The accelerometer samples are used to determine whether the person is sitting, walking, or exercising. The Bluetooth MAC addresses on the cellphones are gathered to identify the neighbors around. Random photos can be taken if the video camera is enabled to provide a snapshot of the current occasion a person is involved with. The results can be uploaded to social network websites as a real time information sharing mechanism.

F. *Precision agriculture*: The LOFAR-agro project [118, 119] deployed 97 wireless sensor nodes on the potato field to monitor humidity and temperature under the crop canopy, as an effort to prevent the fungal disease. The network experienced severe packet losses, attributed to the ill-formed routing topology and inadequate neighbor list management scheme in the MAC layer. The Deluge scheme used for network programming also kept most nodes constantly awake and considerably shortened the network lifetime. This project provides several valuable lessons for all researchers planning a large-scale deployment. Wark *et al.* developed a "smart farm" [120] that applies wireless sensor network technology

to animal agriculture. The smart farm incorporates static sensors to monitor the pasture growth and mobile sensors attached on cattle to monitor the animal behavior. The static nodes are placed in the soil to monitor the temperature and soil moisture from which the growth of the pasture can be predicted. The mobile node consists of a GPS receiver, an electronic compass and an accelerometer. The cattle behavior such as sleeping, grazing and ruminating can be monitored from the accelerometer readings and the speed reported by GPS. Combined with the location information, the farm owners can protect the pasture areas from overgrazing and land erosion. Beckwith *et al.* [121, 3] use the wireless sensor network to predict maturity of grapes based on temperature readings and perform preventive measure if a potential frost is predicted. The Camalie Networks Solutions [10] provides the technical support for wireless vineyard monitoring. One of their early deployments placed 26 nodes to monitor the temperature, soil moisture and the water tank pressure. By adjusting the irrigation system according to the temperature and soil moisture, the farmers can improve the grape quality and reduce the water consumption.

G. *Inventory tracking*: The MAX system [122] was designed to facilitate human-centric search of the objects in the physical world. MAX employs a hierarchical structure, consisting of the base station, the substations and the tags. Each object carries a "tag" which is either an RFID tag or a wireless sensor node. The tag contains the description of the labelled object and the substations are placed at the selected locations as landmarks. When a query is issued, the substations will collect the responses from the tags and find the matched replies. The RSSI of the response message serves as an indicator to the distance of the objects. The results are presented based on the similarity level and the estimated distance. McKelvin *et al.* [123] and Mason *et al.* [124] proposed to integrate the RFID reader with a wireless sensor module through the RS232 interface, such that the response received by the readers can be relayed back to the data sink through the ad hoc networks formed by sensor modules over a longer distance. As the RFID tags may carry the identity information of the target object or target area, the combination of RFID readers with sensor modules will help reveal the relations between the sensed events and the corresponding sources. Ho *et al.* proposed a prototype system [125] that uses RFID tags and sensors to monitor patients' medication in take. The patients and the medicine are identified by the RFID tags they carry. The consumption of medicine can be computed from the weight change of the medicine bottles.

Appendix B

Challenges and Research Areas

Table B.1 provides the specifications of the commonly used sensor nodes. Some sensor network testbeds open to public are shown in Table B.2.

Table B.1: Hardware Specifications of Sensor Nodes

Name	Processor	Memory	Flash	Radio	Provider
BSN	TI MSP430 8MHz	-	64KB	TI CC2420 2.4GHz, 250kbps	Imperial College, UK
BTnode rev3	ATmega128L 8MHz	4KB	128KB	TI CC1000 868MHz, 76.8kbps	ETH, Zurich
Ember EM351	ARM Cortex M3 12MHz	12KB	128KB	2.4GHz, 250kbps	Ember
eyesIFXv2	TI MSP430 4MHz	10KB	40KB	Infineon TDA5250 868MHz, 19.2kbps	EYES
Imote2	Intel PXA271 Xscale 13-416MHz	256KB	32MB	TI CC2420 2.4GHz, 250kbps	CrossBow
Medusa	Atmega128L 4MHz	4KB	32KB	RFM TR1000 916.5MHz, 115.2kbps	UCLA
MeshBeaon2	ATmega1281V 4MHz	4KB	128KB	AT86RF230 2.4GHz, 250kbps	Meshnetic
Mica2	Atmega128L 8MHz	4KB	128KB	TI CC1000 868/916MHz, 38.4kbps	CrossBow
MICAz	ATmega128L 8MHz	4KB	128KB	TI CC2420 2.4GHz, 250kbps	CrossBow
TelosB	TI MSP430 8MHz	10KB	48KB	TI CC2420 2.4GHz, 250kbps	CrossBow
TMote Sky	TI MSP430 8MHz	10KB	48KB	TI CC2420 2.4GHz, 250kbps	Moteiv
uPart Particle 2/29	PIC 18F6720 20MHz	1KB	128KB	RFM TR1001 868MHz, 125kbps	TECO, University of Karlsruhe

The compact size and the low power hardware components in the wireless sensors impose new challenges for the protocol design and network deployment.

- **Radio Range:** The radio chips currently used in a typical wireless sensor node commonly operate at high frequencies, providing limited network bandwidth and a relatively short transmission range. Powered by the batteries, the output power of radio is very low, which implies that the network connectivity may be easily interfered leading to frequent updates in the topology. Due to the limited radio range and network bandwidth, high network densities are preferred in order to

Table B.2: Sensor Network Testbeds

Name	Sensor Platform	Network Size	Deployment	OS	Maintainer
Emulab [126]	Mica2, Stargate	25 (static) + 6 (mobile)	Indoor, single floor	TinyOS	University of Utah
Indriya [17]	TelosB	127	Indoor, 3 floors	TinyOS	National University of Singapore
JHU Testbed [127]	TMote	47	Indoor	Kamikaze	John Hopkins University
Kensai [128]	XSM, Stargate	210	Indoor, grid, 5800 sq feet	Emstar Dev.	Ohio-State University
Mirage [129]	MICAz	100	Indoor, single floor, 160×140 sq feet	TinyOS	Intel Research, Berkeley
MoteLab [130]	TMote Sky	190	Indoor, 3 floors	TinyOS	Harvard University
NetEye [131]	TelosB	130	Indoor	TinyOS	Wayne State University
TWIST [132]	TMote Sky, eyesIFX	102, 102	Indoor, 3 floors	TinyOS	Technische Universität Berlin
Tutornet [133]	TMote Sky, MICAz	91, 13	Indoor, 2 floors	TinyOS	University of Southern California
VineLab [134]	TMote Sky	48	Indoor, 1 floor	TinyOS	Virginia University

provide sufficient coverage and robustness for network communication. For data collection in such a dense network, the packet transmission should be carefully planned, in order to reduce intra-flow and inter-flow contention.

- **Error Detection:** Most radio transceivers in the sensor nodes only provide limited error detection functions. For critical application requiring reliable delivery, MAC layer retransmission is normally enabled to counter the effects of packet drops. Therefore, the application layer must handle duplicates resulted from premature MAC retransmissions. All packets must be tagged at the application layer such that duplicates can be identified and ignored.

- **Battery Power:** For sensor nodes using batteries as the sole energy supply, the duty-cycling mechanism must be enabled to extend the network lifetime for long term deployments. The duty-cycle ratio must be determined based on the task required on each node. As the radio chip switches to the sleep mode periodically, nodes must coordinate with the neighbors to ensure the packet delivery will not be interrupted by the power saving mechanism and the network latency is maintained at a satisfactory level. Time synchronization is often required to ensure the relevant nodes will remain available for the transmission.

- **Processing Capacity:** The low power processors equipped on the sensor nodes may become another performance bottle neck. On TelosB, the TI MSP430 processor has a clock frequency of 8 MHz, which is generally adequate for radio communication. However, some applications require more sophisticated in-network data processing to reduce the network traffic. For example, sound samples collected in the time domain may be transformed into the frequency domain for spectrum analysis. Such procedures often involve computation over large volume of data, imposing extra workload on the processor. The algorithm and program flow must be optimized carefully, such that the normal network operations will not be affected.

- **Memory:** One of the toughest challenges of working on the sensor motes is to cope with the limited amount of memory available for programming. For example, a TelosB mote is equipped with a RAM of 10 KBytes and a programming flash memory of just 48 KBytes. The RAM determines the total amount of memory available to the global variables at run time. The program

flash memory provides space for data structures and function declarations. All auxiliary variables and data structures must be simplified to comply with the memory constraints. As sensor applications often involve packet exchange, packet buffering is commonly applied to reduce packet drops during transient network congestion. Given the maximum packet size of 128 bytes in TinyOS-2, a buffer with 10 slots will occupy 1.2 KBytes of RAM. Another memory consuming component is the neighbor list holding the neighboring nodes' information, such as the node ID, coordinates and battery level. As the network density varies with different radio output powers at different locations, when the neighbor list size cannot accommodate all candidates, the network protocol must incorporate an eviction algorithm to select desirable neighbors for the application.

- **Software:** The limitations in the development toolkit create another hurdle for designing and implementing sensor network protocols. The TinyOS library provides limited options for data structures and functions, especially in memory management. For example, it does not support two-dimensional arrays, which is inconvenient for applications involving matrix operations. Although floating number calculation is allowed on a mote, a floating point value must be packed into an integer variable for network transmission. Dynamic memory management is not yet supported in the standard TinyOS library. Recursion should be avoided in the algorithm design in order to avoid the risk of stack overflow. The cross compilers currently employed are not as intelligent as the standard C compilers; thus, memory operations must be examined carefully and run time errors may lead to malfunction of multiple components. Due to the lack of an output console for debugging information, the development of sensor network program tends to be more time consuming.

To cope with the hardware constraints and tackle the challenges in the software stack, new protocols have been proposed from various aspects to facilitate the intended tasks. Some active areas in wireless sensor network research are summarized below.

- *Routing* protocols are in charge of delivering data packets across the network through multi-hop forwarding. In a data collection scenario, convergecast can be achieved through a spanning tree [135] where routing paths are built on reliable links. In a data dissemination scenario, information will spread from a source node to the entire network. To disseminate larger volume of data such as network re-programming [136], data integrity must be ensured and data discrepancies must be resolved efficiently. For more complex network functions where point-to-point delivery service is required, more generic routing protocols must be employed.

- *Localization* algorithms [59, 58] provide the network embedding function, which computes the node coordinates through the topological information. It allows the target events to be reported with specific addresses and the coordinates can also be used in geographic routing to support more complex communication patterns other than convergecast.

- *Time synchronization* algorithms [137] allow node scheduling to be implemented, such that nodes can switch to power saving modes in a coordinated manner while maintaining sufficient network coverage and connectivity.

- *Congestion control* algorithms [9, 138] will help to reduce network contention due to concurrent transmissions or bandwidth limitation. Some congestion control algorithms perform bandwidth allocation to the flows based on the scale or priority of the reported information, creating a prioritized transmission mechanism.

- *Topology control* protocols [139, 140] help to maintain good network connectivity by ignoring the weak links among the neighboring connections. By avoiding the poor links, packets can be forwarded through more reliable paths, such that higher delivery ratio and better energy savings can be achieved.

- *Data aggregation* [141] or *data compression* [142] algorithms are applied to reduce the network traffic through in-network data processing. The data aggregation algorithms allow each node to create application dependent index values from collected data, such that the sensed information can be summarized and conveyed in a compact format. The data compression algorithms can be applied to trade the low computational cost for the high communication cost in terms of energy consumption.

- *MAC* protocols [143] can play a crucial part on energy saving by reducing packet collisions [144], idle listening [145] and the control packet overhead. As duty-cycling is often employed [146] to reduce power consumption during periods of in-activities, MAC protocols must provide a proper scheduling among the neighboring nodes to ensure efficient packet delivery.

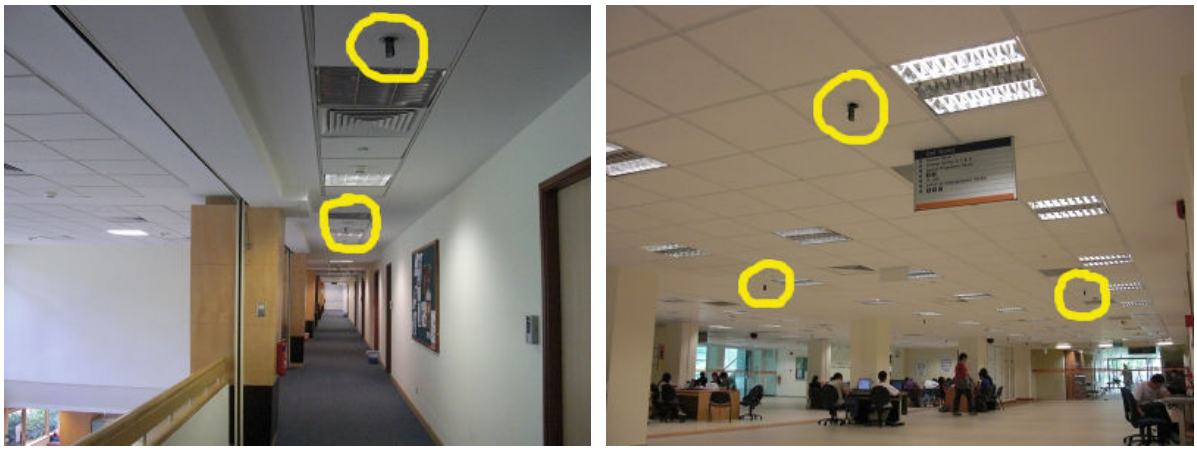
- *Security* and *encryption* algorithms [147, 148] can help to counter the effects of malicious nodes and prevent unauthorized access to the sensor network resources.

- *Network coding* algorithms [149, 150] can provide resilience to packet corruption and packet losses in network transmissions by introducing data redundancy in the packets. It can also protect the data integrity of in-network storage systems from individual node failures.

Appendix C

Connectivity and Link Quality Measurements

The Indriya testbed [17] contains 127 TelosB motes deployed on three floors of the COM-1 building of School of Computing at the National University of Singapore. The nodes are divided into 6 clusters, where power supply and the backbone connections are provided through USB links. A web interface ported from the MoteLab [16] project is used for testbed status monitoring and job management. Some snapshots of the testbed are given in Fig. C.1, showing the motes installed on the ceilings of two floors in COM-1.



(a) Level-1, Corridor (b) Level-2, Student Area

Figure C.1: Snapshots of the Indriya Testbed at COM-1 Building

The specifications of the TelosB motes are summarized in Table C.1. The *program flash memory* indicates that the program image size should be limited to 48 KBytes. The run time variables should occupy less than 10 KBytes of RAM. The CC2420 radio chip has a data rate of 250 kbps compliant with the IEEE 802.15.4 standard. The output power level of the radio can be configured from level 1 to 31, equivalent to the range from -24 dBm to 0 dBm. As Indriya is an indoor testbed, the expected communication range is between 20 to 30 meters due to the presence of obstacles. Our experiments show that the observed link can exceed 50 meters when two nodes are within the line of sight.

Table C.1: Specifications of the TelosB Mote (source: [7]).

Attribute	Value
Processor	8 MHz, TI MSP430, 16-bit RISC
Program Flash Memory	48 KBytes
RAM	10 KBytes
Measurement Serial Flash	1024 KBytes
Frequency band	$2400 \sim 2483.5$ MHz
Data Rate	CC2420, 250 kbps
RF Power	-24 dBm ~ 0 dBm
Range	Outdoor: $75 \sim 100$ meters Indoor: $20 \sim 30$ meters
Antenna	Integrated Onboard Antenna
Interface	USB

We measure the network density, the link quality with both broadcast and unicast packets and the network topology in the Indriya testbed. The network density will indicate the minimum and maximum number of neighbors of the nodes, which can be used to determine the neighbor list size in the program. The average density value can also be used to the select beacon intervals, in order to maintain the agility to reflect network dynamics with low overhead traffic. The link qualities collected from the testbed will be used to determine the appropriate buffer size on the radio interface in order to cope with the bursty traffic and retransmissions. We compare the link quality results measured from broadcast and unicast packets, showing that both methods will obtain nearly identical results. This implies that it is a valid approach to use broadcast beacons for bi-directional link quality estimation on the Indriya testbed. The network topology can be derived from the connectivity information. It will reveal the network diameter and the distribution of node clusters, which can help improve the routing efficiency of the anchor nodes selected manually.

C.1 Configuration of Link Measurements

Both *broadcast* and *unicast* probes are employed to measure the network connectivity and the link qualities. The parameters are summarized in Table C.2. Each node maintains a neighbor list with a maximum capacity of 40 and the warm up period is set to 300 seconds for the neighbor discovery process. A probe message is 19 bytes in length, including the source and destination node IDs, the probe iteration index, the total number of probes and a sequence number. A node with k neighbors will send the probes in $k+1$ iterations, where the first batch will be sent as broadcast to all neighbors and each subsequent iteration will use unicast to probe an individual neighbor. Each node is scheduled to send packets at a different interval to avoid collisions due to concurrent probing. The transmission power is set to the maximum of 0 dBm.

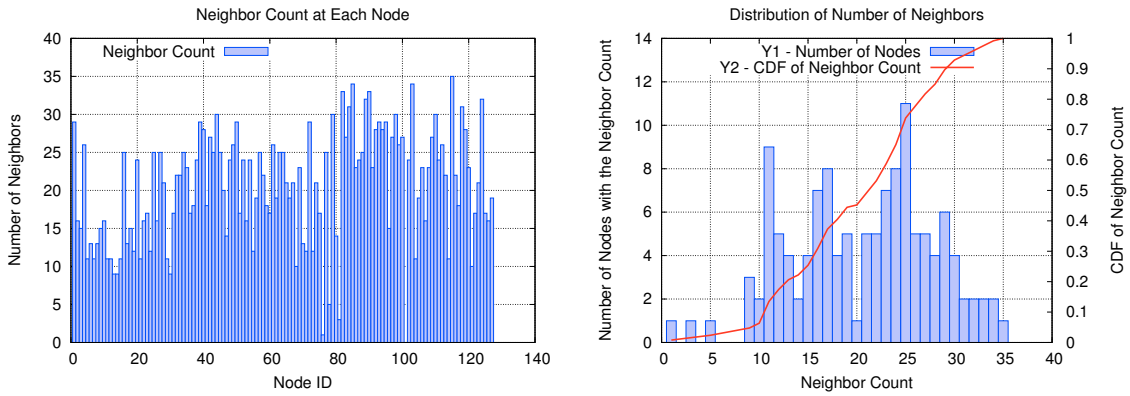
C.2 Network Density

The network density on the Indriya testbed is given in Fig. C.2, showing the number of neighbors at each node and the neighbor count distribution. Although the placement of the nodes is arranged in a grid pattern, the connectivity varies significantly. The number of neighbors observed by the nodes ranges

Table C.2: Configuration of Link Measurements

Parameter	Value
Number of Probes	100
Probe Message Size	19 bytes
Probe Interval	*burst*
Probe Method	Broadcast and Unicast
RTX and ACK	*disabled*
Neighbor List Capacity	40
Transmission Power	max level of 31, 0 dBm
Warm up Duration	300 seconds
Initial Probe Delay	5 minutes × node ID
Beacon Interval	10 seconds

from 1 to 35, depending on the location and the surrounding environment. Node 101 fails to log its neighbor list to the database due to a faulty USB cable. Fig. C.2(b) reveals that over 80% of the nodes have more than 10 neighbors and the average neighbor count is 21; therefore, the testbed is a dense network at the maximum transmission power.



(a) Number of Neighbors at Each Node

(b) Neighbor Count Distribution

Figure C.2: Neighbor Count of 127 Nodes in the Indriya Testbed.

C.3 Link Quality

We observed 2445 directional links during the experiment and the cumulative distribution of the packet reception ratios is given in Fig. C.3. The link qualities measured by the broadcast and unicast probes follow two nearly identical distributions. As the average inter-node distance on the deployment grid is 8 meters (much shorter than the expected indoor range of 20 to 30 meters [7]) and all nodes use the default maximum transmission power, most links exhibit high reliability. The average reception ratio is 99.2% and 85% of the links delivered more than 95% of the unicast and broadcast probe packets.

The total number of node pairs that have connections in at least one direction is 1363. For every connected node pair, its link quality in each direction is given in Fig. C.4. Each point represents a connected pair $A \Leftrightarrow B$ and its position (p_1, p_2) indicates that the Packet Reception Ratio (PRR) from Node A to B is p_1 and the PRR in the reverse direction is p_2. We examine the bi-directional link

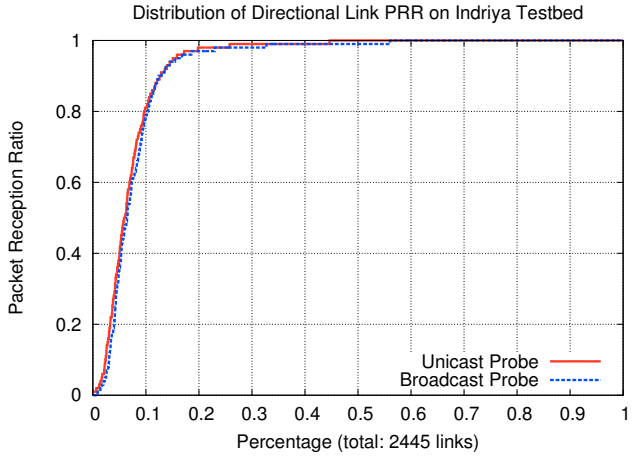
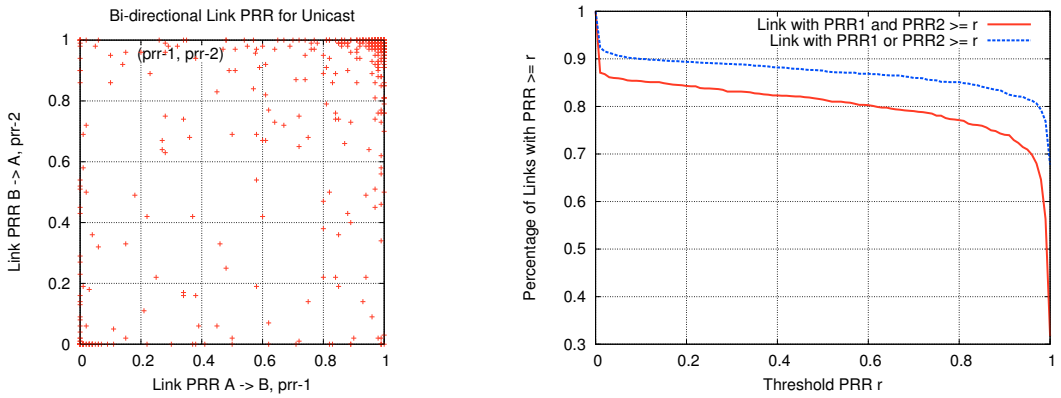


Figure C.3: CDF of Link Packet Reception Ratio, number of links = 2445

qualities and plot the CDF curve in Fig. C.4(b). While 74% of the links retain a high PRR $\geq 90\%$ in both directions, around 83% links have a PRR $\geq 90\%$ in at least one direction. As the network is dense and most links are highly reliable in both directions, a blacklisting method can be incorporated to ignore the weak links without sacrificing network connectivity. Since the probes are scheduled to avoid interferences from simultaneous probing at nearby neighbors, these results represent the network link quality in an ideal scenario under minimum network contention. Nevertheless, we believe that the results provide a benchmark on the best performance a data delivery protocol can achieve on the testbed.



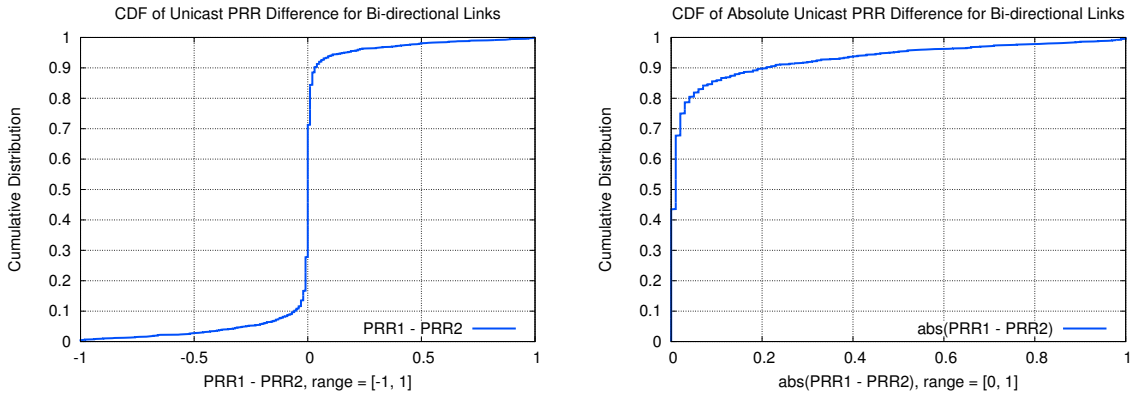
(a) Packet Reception Ratio

(b) CDF of Bi-directional Link Quality

Figure C.4: Packet Reception Ratio on Bi-directional Links

We define the level of link asymmetry as the difference of the PRRs $(p_1 - p_2)$. In Fig. C.5, over 85.9% of the links have consistent delivery ratios in both directions, with a difference less than 0.1. Fig. C.5(b) depicts the CDF of the absolute PRR difference, where 43.6% of the links have identical delivery ratios in both directions. Around 82% of the links have $|p_1 - p_2| < 0.05$, revealing that most links in the testbed are highly symmetric. There are also 2% of the links having a PRR difference above 0.8, most of which belong to transient connections that fail to be discovered by one side of the link during the experiment.

As mentioned in [151], the wireless link quality estimated with broadcast packets may fail to reflect



(a) CDF of PRR Difference (b) CDF of Absolute PRR Difference

Figure C.5: Link Asymmetry on the Bi-directional Links

the actual throughput of the unicast traffic due to different data rates used for broadcast and unicast traffic. On the Indriya testbed, the CC2420 radio [6] in the TelosB motes provides a transmission rate of 250 kbps for all packets. In order to examine the validity of using broadcast probes to estimate the unicast reception ratio, we compare the link estimation results obtained from the broadcast and unicast probes in Fig. C.6 and Fig. C.7. Every point in Fig. C.6 represents a directional link, whose coordinates (x, y) are formed by the broadcast PRR $p_b = x$ and its unicast PRR $p_u = y$. As most of the data points are aligned along the diagonal, the link quality estimated from the broadcast probes is generally consistent with the unicast results. The PRR difference between the broadcast traffic and the unicast traffic is depicted in Fig. C.7(a). Over 91.7% of the links retain a link quality difference less than 0.1 and on 87.3% of the links, the difference is below 0.05. There are 1.2% of the links with $|p_u - p_b| > 0.8$, most of which are transient connections unavailable during either the unicast or broadcast probe iteration. As the unicast performance can be closely approximated by the broadcast performance, the results in Fig. C.7 imply that broadcast probing is an efficient and reliable method for link quality estimation on the Indriya testbed.

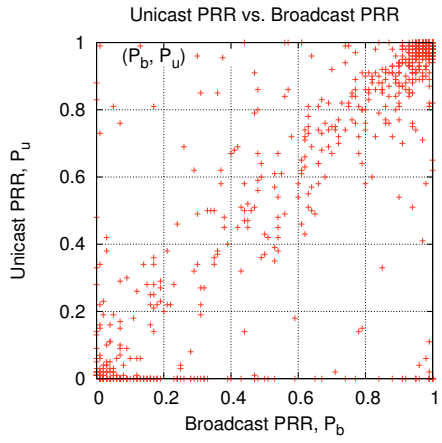
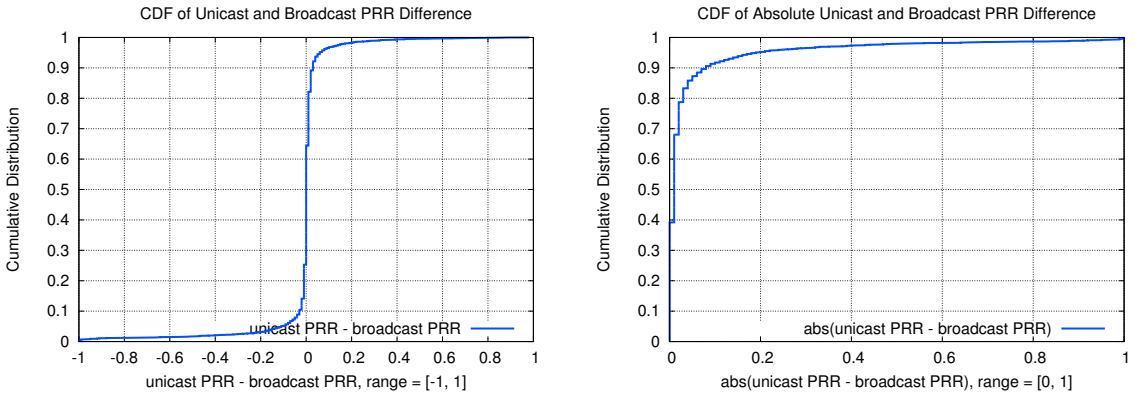


Figure C.6: Packet Reception Ratio of Unicast and Broadcast Probes



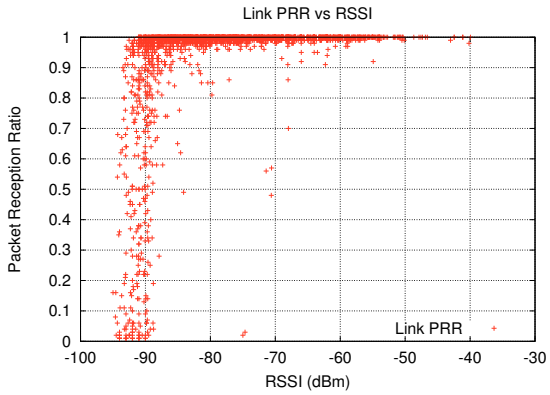
(a) Unicast PRR - Broadcast PRR (b) abs(Unicast PRR - Broadcast PRR)

Figure C.7: PRR Difference for Broadcast and Unicast Probes

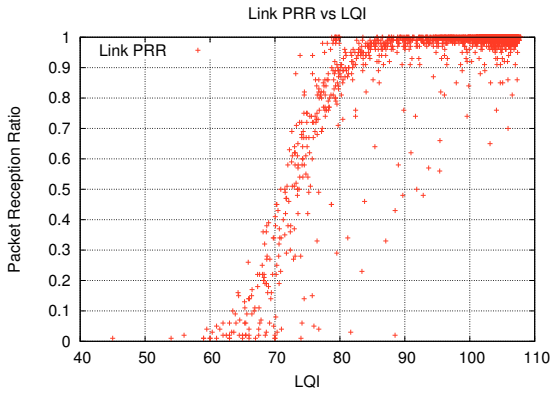
C.4 RSSI and LQI

Srinivasan *et al.* [72] evaluated the approach of deriving real time link qualities based on RSSI and LQI values reported by the CC2420 radio chip. Their results conclude that a threshold RSSI value can be utilized as a binary indicator to identify the reliable links with high reception ratios. As a comparison, we compute the average RSSI and LQI values for all the unicast probe packets collected from the testbed and plot the relations between RSSI, LQI and PRR in Fig. C.8. The RSSI value observed on the links varies between -95 dBm to -38 dBm. There is a sharp increase in the average reception ratio when the average RSSI increases above -90 dBm. The *transitional area* lies between the RSSI values of -95 dBm and -87 dBm, where the link PRR fluctuates between 0% and 100%. When the average RSSI is larger than -87 dBm, the packet reception ratio consistently remains above 95%, indicating that the average RSSI can serve as a threshold to differentiate the high quality links from the weak ones. The relation between the average LQI and PRR in Fig. C.8(b) shows a different pattern. The perceived LQI value lies between 54 to 107, while most links have an average LQI greater than 95. The PRR linearly increases as LQI grows from 60 to 90. For LQI values above 95, the corresponding packet reception ratios remain above 98% on average.

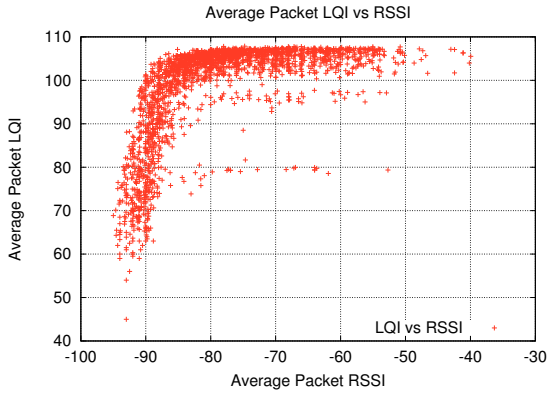
We plot the average RSSI and the corresponding LQI readings of each links in Fig. C.8(c). There is a radical increase for the LQI value as RSSI grows from -95 dBm to -87 dBm. When the average RSSI on a link is ≥ -87 dBm, the corresponding LQI readings are usually above 95. Combined with the PRR results in Fig. C.8, the RSSI value of -87 dBm and the LQI value of 95 can be utilized as the threshold to distinguish the good links with an average PRR above 95%. Although there is also a correspondence between the average LQI and PRR at the transitional region with RSSI between -95 dBm and -87 dBm, it is not feasible to estimate the link quality based on LQI in this range due to the large PRR variation (not shown) and the lack of samples.



(a) PRR vs. RSSI



(b) PRR vs. LQI



(c) RSSI vs. LQI

Figure C.8: Packet Reception Ratio vs. RSSI and LQI

Bibliography

[1] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, and John Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 88–97, Atlanta, GA, USA, 2002.

[2] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46:605–634, 2004.

[3] Jenna Burrell, Tim Brooke, and Richard Beckwith. Vineyard computing: Sensor networks in agricultural production. *Pervasive Computing*, pages 10–17, 2004.

[4] Konrad Lorincz, Bor rong Chen, Geoffrey Werner Challen, Atanu Roy Chowdhury, Shyamal patel, Paolo Bonato, and Matt Welsh. Mercury: A wearable sensor network platform for high-fidelity motion analysis. In *Proceedings of th 7th International Conference on Embedded Network Sensor System*, Berkeley, CA, USA, 2009.

[5] Prashanth Mohan, Venkata N. Padmanabhan, and Ramachandran Ramjee. Nericell: Rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th International Conference on Embedded Network Sensor Systems*, pages 323–336, Raleigh, NC, USA, 2008.

[6] Texas Instrument. *CC2420: 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*, 2008. `http://focus.ti.com/lit/ds/symlink/cc2420.pdf`.

[7] Crossbox Technology Inc. *TelosB - Telosb Mote Platform*, 2009. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf`.

[8] Sylvia Ratnasamy, Brad Karp, Lin Yin, Fang Yu, Deborah Estrin, Ramesh Givindan, and Scott Shenker. GHT: A geographic hash table for data-centric storage. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 78–87, Atlanta, Georgia, USA, 2002.

[9] Sumit Rangwala, Ramakrishna Gummadi, Ramesh Govindan, and Konstantinos Psounis. Interference-aware fair rate control in wireless sensor networks. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures and Protocols for Computer Communications*, pages 63–74, Pisa, Italy, 2006.

[10] Camalie Networks LLC Wireless Vineyard Monitoring. `http://camalienetworks.com/`.

[11] Yong Yao and Johannes Gehrek. The cougar approach to in-network query processing in sensor networks. *SIGMOD Rec.*, 31(3):9–18, 2002.

[12] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *SIGCOMM Comput. Commun. Rev.*, 24(4):234–244, 1994.

[13] Charles E. Perkins and Elizabeth M. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, USA, 1999.

[14] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[15] Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson, and David Culler. An analysis of a large scale habitat monitoring application. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor System*, pages 214–226, Baltimore, MD, USA, 2004.

[16] Geoffrey Werner-Allen, Patrick Swieskowski, and Matt Welsh. Motelab: A wireless sensor network testbed. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, Los Angeles, CA, USA, 2005.

[17] Communication and Internet Research Lab. *Indriya Testbed*. National University of Singapore, 2009. `http://indriya.comp.nus.edu.sg`.

[18] Brad Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual International Conference Mobile Computing and Networking*, pages 243–254, Boston, MA, USA, 2000.

[19] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: of theory and practice. In *Proceedings of the 22nd Annual Symposium on Principles of Distributed Computing*, pages 63–72, Boston, MA, USA, 2003.

[20] Young-Jin Kim, Ramesh Govindan, Brad Karp, and Scott Shenker. On the pitfalls of geographic face routing. In *Proceedings of the 2005 Joint Workshop on Foundations of Mobile Computing*, pages 34–43, Cologne, Germany, 2005.

[21] J. Newsome and D. Song. GEM: Graph embedding for routing and data-centric storage in sensor networks without geographic information. In *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems*, pages 76–88, Los Angeles, CA, USA, 2003.

[22] B. Leong, B. Liskov, and R. Morris. Geographic routing without planarization. In *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation*, pages 339–352, San Jose, CA, USA, 2006.

[23] Q. Cao and T. Abdelzaher. A scalable logical coordinates framework for routing in wireless sensor networks. In *Proceedings of the 25th IEEE Real-Time Systems Symposium*, pages 349–358, Lisbon, Portugal, 2004.

[24] R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. Stoica. Beacon vector routing: Scalable point-to-point routing in wireless sensornets. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation*, volume 2, pages 329–342, Boston, MA, USA, 2005.

[25] S. Srinivasan and E. Zegura. An empirical evaluation of landmark placement on internet coordinates schemes. In *Proceedings of the 13th International Conference on Computer Communications and Networks*, pages 335–340, Chicago, IL, USA, 2004.

[26] Ke Liu and Nael Abu-Ghazaleh. Addressing anomelies in geometric routing. In *Proceedings of the IEEE Upstate New York Workshop on Communications and Networks*, Rochester, NY, USA, 2006.

[27] Hideaki Takagi and Leonard Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Transactions on Communications*, 32(3):246–257, 1984.

[28] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. A distance routing effect algorithm for mobility(DREAM). In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 76–84, Dallas, Texas, USA, 1998.

[29] Evangelos Kranakis, Harvinder Singh, and Jorge Urrutia. Compass routing on geometric networks. In *Proceedings of the 11th Canadian Conference on Computational Geometry*, pages 51–54, Vancouver, Cananda, 1999.

[30] Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[31] Q. Cao and T. Abdelzaher. Scalable logical coordinates framework for routing in wireless sensor networks. *ACM Transactions on Sensor Networks*, 2(4):557–593, 2006.

[32] Yao Zhao, Bo Li, Qian Zhang, Yan Chen, and Wenwu Zhu. Efficient hop id based routing for sparse ad hoc networks. In *Proceedings of the 13th IEEE International Conference on Network Protocols*, pages 179–190, Boston, MA, USA, 2005.

[33] Roland Flury and Roger Wattenhofer. Randomized 3d geographic routing. In *Proceedings of the 27th Annual IEEE Conference on Computer Communications*, pages 834–842, Phoenix, AZ, USA, 2008.

[34] Omnet++ community site. `http://www.omnetpp.org/`.

[35] The network simulator - ns-2. `http://www.isi.edu/nsnam/ns/`.

[36] Tinyos 2.1.0. `http://www.tinyos.net/`.

[37] Gregory G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISI/RR-87-180, University of Southern California Marina del Rey Information Science Institute, Los angeles, CA, USA, 1987.

[38] Ben Leong, Sayan Mitra, and Barbara Liskov. Path vector face routing: Geographic routing with loal face information. In *Proceedings of the 13th IEEE International Conference on Network Protocols*, pages 147–158, Boston, MA, USA, 2005.

[39] Susanta Datta, Ivan Stojmenović, and Jie Wu. Internal node and shortcut based routing with guaranteed delivery in wireless networks. In *Proceedings of the 21st International Conference on Distributed Computing Systems Workshop*, pages 461–466, Mesa, AZ, USA, 2001.

[40] Hannes Frey. Scalable geographic routing algorithms for wireless ad hoc networks. *IEEE Network*, 18(4):18–22, 2004.

[41] Hannes Frey and Ivan Stojmenović. On delivery guarantees of face and combined greedy-face routing in ad hoc and sensor networks. In *Proceedings of the 12th International Conference on Mobile Computing and Networking*, pages 390–401, Los Angeles, CA, USA, 2006.

[42] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008.

[43] Jerzy W. Jaromczyk and Godfried T. Toussaint. Relative neighborhood graphs and their relatives. *Proceedings of the IEEE*, 80(9):1502–1517, 1992.

[44] K. Ruben Gabriel and Robert R. Sokal. a new statistical approach to geographic variation analysis. *Systematic Zoology*, 18(3):259–278, 1969.

[45] Xiang-Yang Li, Gruia Calinescu, and Peng-Jun Wan. Distributed construction of a planar spanner and routing for ad hoc wireless networks. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1268–1277, New York, NY, USA, 2002.

[46] Alec Woo, Terence Tong, and David Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pages 14–27, Los Angeles, CA, USA, 2003.

[47] Young-Jin Kim, Ramesh Govindan, Brad Karp, and Scott Shenker. Lazy cross-link removal for geographic routing. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, pages 112–124, Boulder, Colorado, USA, 2006.

[48] Jiangwei Zhou, Yu Chen, Ben Leong, and Pratibha Sundar Sundaramoorthy. Practical 3d geographic routing for wireless sensor networks. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, Zurich, Switzerland, 2010.

[49] G. R. Hjaltason and H. Samet. Properties of embedding methods for similarity searching in metric space. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 25(5):530–549, 2003.

[50] Yun Mao, Feng Wang, Lili Qiu, Simon S. Lam, and Jonathan M. Smith. S4: Small state and small stretch routing protocol for large wireless sensor networks. In *Proceedings of the 4th USENIX Symposium on Networked Systems Design and Implementation*, Cambridge, MA, USA, 2007.

[51] Y. Chen, K. H. Lim, R. Katz, and C. Overton. On the stability of network distance estimation. *ACM SIGMETRICS Performance Evaluation Review*, 30(2):21–30, 2002.

[52] R. Zhang, Y. Charlie Hu, X. Lin, and S. Fahmy. A hierarchical approach to internet distance prediction. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, pages 73–80, Lisboa, Portugal, 2006.

[53] G. Kao, T. Fevens, and J. Opatrny. Position-based routing on 3d geometric graphs in mobile ad hoc networks. In *Proceedings of the 17th Canadian Conference on Computational Geometry*, pages 88–91, Windsor, Canada, 2005.

[54] A. E. Abdallah, T. Fevens, and J. Opatrny. Power-aware 3d position-based routing algorithms for ad hoc networks. In *Proceedings of the IEEE International Conference on Communications*, pages 3130–3135, Glasgow, Scotland, 2007.

[55] M. Costa, M. Castro, A. Rowstron, and P. Key. Pic: Practical internet coordinates for distance estimation. In *Proceedings of the 24th International Conference on Distributed Computing Systems*, pages 178–187, Tokyo, Japan, 2004.

[56] Y. Mao and L. K. Saul. Modeling distances in large-scale networks by matrix factorization. In *Proc. of the 4th ACM Internet Measurement Conference*, pages 278–287, Taormina, Sicily, Italy, 2004.

[57] C. Lumezanu and N. Spring. Playing vivaldi in hyperbolic space. Technical Report CS-TR-4843, University of Maryland, College Park, 2006.

[58] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *Proceedings of the 9th ACM Annual International Conference on MObile Computing and Networking*, pages 96–108, San Diego, CA, USA, 2003.

[59] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *Proceedings of the 23rd Annual IEEE Conference on Computer Communications*, pages 15–26, Portland, Oregon, USA, 2004.

[60] B. Leong, B. Liskov, and R. Morris. Greedy virtual coordinates for geographic routing. In *Proceedings of the 2007 International Conference on Network Protocols*, pages 71–80, Beijing, P.R.China, 2007.

[61] Gilbert Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 1st edition, 1993.

[62] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. Kolaczyk, and N. Taft. Structural analysis of nework traffic flows. In *Proceedings of Joint ACM SIGMETRICS / Performance conference*, pages 61–72, New York, NY, USA, 2004.

[63] P. Xi, C. Shu, and M. Rioux. Principal components analysis of 3-d scanned human heads. In *Proceedings of the 34th International Conference and Exhibition on Computer Graphics and Interactive Techniques (poster)*, San Diego, California, USA, 2007.

[64] H. Lim, J. C. Hou, and C. Choi. Constructing internet coordinate system based on delay measurement. In *Proceedings of the 3rd ACM Internet Measurement Conference*, pages 129–142, Miami, FL, USA, 2003.

[65] L. Tang and M. Crovella. Virtual landmarks for the internet. In *Proceedings of the 3rd ACM Internet Measurement Conference*, pages 143–152, Miami, FL, USA, 2003.

[66] J. Kleinberg, A. Slivkins, and T. Wexler. Triangulation and embedding using small sets of beacons. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 444–453, Rome, Italy, 2004.

[67] Jerry Zhao and Ramesh Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pages 1–13, Los Angeles, CA, USA, 2003.

[68] Alberto cerpa, Jennifer L. Wong, Louane Kuang, Miodrag Potkonjak, and Deborah Estrin. Statistical model of lossy links in wireless sensor networks. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, pages 81–88, Los Angeles, CA, USA, 2005.

[69] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, pages 134–146, San Diego, CA, USA, 2003.

[70] Alberto Cerpa, Jennifer L. Wong, Miodrag Potkonjak, and Deborah Estrin. Temporal properties of low power wireless links: Modeling and implications on multi-hop routing. Technical Report CENS 0044, UCLA, 2005.

[71] Gentian Jakllari, Stephan Eidenbenz, Nicolas Hengartner, Srikanth V. Krishnamurthy, and Michalis Faloutsos. Link positions matter: A noncommutative routing metric for wireless mesh networks. In *Proceedings of the 27th IEEE Conference on Computer Communications*, pages 744–752, Phoenix, AZ, USA, 2008.

[72] Kannan Srinivasan and Philip Levis. Rssi is under appreciated. In *Proceedings of the 3rd Workshop on Embedded Networked Sensors*, Cambridge, MA, USA, 2006.

[73] Omprakash Gnawali, Mark Yarvis, John Heidemann, and Ramesh Govindan. Interaction of retransmission, blacklisting and routing metrics for reliability in sensor network routing. In *Proceedings of 1st IEEE International Conference on Sensor and Ad Hoc Communications and Networks*, pages 34–43, Santa Clara, CA, USA, 2004.

[74] Tao Liu, Ankur Kamthe, Lun Jiang, and Alberto Cerpa. Performance evaluation of link quality estimation metrics of static multihop wireless sensor networks. In *Proceedings of the Sixth Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, Rome, Italy, 2009.

[75] Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson, and Philip Levis. Four-bit wireless link estimation. In *Proceedings of the 6th Workshop on Hot Topics in Networks (HotNets)*, Atlanta, GA, USA, 2007.

[76] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris. A scalable location service for geographic ad hoc routing. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 120–130, Boston, Massachusetts, United States, 2000.

[77] Jorge Ortiz, Chris R. Baker, Daekyeong Moon, Rodrigo Fonseca, and Ion Stoica. Beacon location service: A location service for point-to-point routing in wireless sensor networks. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, pages 166–175, Cambridge, Massachusetts, USA, 2007.

[78] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[79] Linksys. *WRT150N Product Data*, 2007. 7032810A-JL, `http://www.linksys.com`.

[80] Crossbow Technology Inc. *MICAz - Wireless Measurement System*, 2008. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAZ_Datasheet.pdf`.

[81] Deepak Ganesan, Deborah Estrin, Alec Woo, and David Culler. Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical Report CSD-TR 02-0013, UCLA, 2002.

[82] Lifeng Sang, Anish Arora, and Hongwei Zhang. On exploiting asymmetric wireless links via one-way estimation. In *Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 11–21, Montréal, Québec, Canada, 2007.

[83] J. B. Tenenbaum, V. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[84] J. Edward Jackson. *A User's Guide to Principal Components*, chapter 3, pages 72–75. Wiley-Interscience, 1st edition, 1990.

[85] J. Hicklin, R. F. Boisvert, and *et al. Jama: Java Matrix Package*. The MathWorks and NIST, `http://math.nist.gov/javanumerics/jama/`, 1.0.2 edition, July 2005.

[86] C. Bond. Singular value decomposition source code, 2000. `http://www.crbond.com/download/misc/svd.c`.

[87] S. Lin, J. Zhang, G. Zhou, L. Gu, T. He, and J. Stankovic. Atpc: Adaptive transmission power control for wireless sensor networks. In *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems*, pages 223–236, Boulder, Colorado, USA, 2006.

[88] Walter Colitti, Kris Steenhaut, Nicolas Descouvemont, and Adam Dunkels. Satellite based wireless sensor networks: Global scale sensing with nano- and pico-satellites. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, pages 445–446, Raleigh, NC, USA, 2008.

[89] Muhammad Omar Khan, Affan Syed, Wei Ye, John Heidemann, and Jack Wills. Bringing sensor networks underwater with low-power acoustic communications. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, pages 379–380, Raleigh, NC, USA, 2008.

[90] Pei Zhang, Christopher M. Sadler, Stephen A. Lyon, and Margaret Martonosi. Hardware design experiences in zebranet. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor System*, pages 227–238, Baltimore, MD, USA, 2004.

[91] Guillermo Barrenetxea, Francois Ingelrest, Gunnar Schaefer, and Martin Vetterli. The hitchhiker's guide to successful wireless sensor network deployments. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, pages 43–56, Raleigh, NC, USA, 2008.

[92] Mohamed Hefeeda and Majid Bagheri. Wireless sensor networks for early detection of forest fires. In *Proceedings of the 2007 International Conference on Mobile Adhoc and Sensor Systems*, pages 1–6, Pisa, Italy, 2007.

[93] Thierry Antoine-Santoni, Jean-Francois Santucci, Emmannuelle de Gentili, Xavier Silvani, and Frederic Morandini. Performance of a protected wireless sensor network in a fire. analysis of fire spread and data transmission. *Sensors*, 9:5879–5893, 2009.

[94] Maxim A. Batalin, Mohammad Rahimi, Yan Yu, Duo Liu, Aman Kansal, Gauray S. Sukhatme, William J. Kaiser, Mark Hansen, Gregory J. Pottie, Mani Srivastava, and Deborah Estrin. Call and response: Experiments in sampling the environment. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor System*, pages 25–38, Baltimore, MD, USA, 2004.

[95] G. Tolle, N. Turner, K. Tu, and P. Buonadonna. A macroscope in the redwoods. In *Proc. of the 3rd ACM Conference on Embedded Networked Sensor Systems*, pages 51–63, San Deigo, CA, USA, 2005.

[96] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke. Data collection, storage and retrieval with an underwater sensor network. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor System*, pages 154–165, San Diego, California, USA, 2005.

[97] Jude Allred, Ahmad Bilal Hasan, Saroch Panichsakul, William Pisano, Peter Gray, Jyh Huang, Richard Han, Dale Lawrence, and Kamran Mohseni. Sensorflock: An airborne wireless sensor network of micro-air vehicles. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, pages 117–129, Sydney, Australia, 2007.

[98] Hock Beng Lim, Keck Voon Ling, Wenqiang Wang, Yuxia Yao, Mudasser Iqbal, Boyang Li, Xiaonan Yin, and Tarun Sharma. The national weather sensor grid. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, pages 369–370, Sydney, Australia, 2007.

[99] Chin-Jung Liu, Huang-Chen Lee, Jung Yang, Jen-Tse Huang, Yao-Min Fang, Bing-Jean Lee, and Chun-Ta King. Development of a long-lived, real-time automatic weather station based on wsn. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, pages 401–402, Raleigh, NC, USA, 2008.

[100] Elizabeth A. Basha, Sai Ravela, and Daniela Rus. Model-based monitoring for early warning flood detection. In *Proceedings of the 6th International Conference on Embedded Network Sensor Systems*, pages 295–308, Raleigh, NC, USA, 2008.

[101] Jinhai Cai, Dominic Ee, Andy Lau, Richard Mason, Binh Pham, Paul Roe, Jinglan Zhang, and Stuart Gage. Acoustic sensor networks for environmental monitoring. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, pages 391–392, Sydney, Australia, 2007.

[102] Lufeng Mo, Yuan He, Yunhao Liu, Jizhong Zhao, Shaojie Tang, Xiang-Yang Li, and Guojun Dai. Canopy closure estimates with greenorbs: Sustainable sensing in the forest. In *Proceedings of the 7th International Conference on Embedded Network Sensor System*, Berkeley, CA, USA, 2009.

[103] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu, W. Kang, J. Stankovic, D. Young, and J. Porter. Luster: Wireless sensor network for environmental research. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, pages 103–116, Sydney, Australia, 2007.

[104] Gyula Simon, Miklos Maroti, Akos Ledeczi, Gyorgy Balogh, Branislav Kusy, Andras Nadas, Gabor Pap, Janos Sallai, and Ken Frampton. Sensor network-based countersniper system. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor System*, pages 1–12, Baltimore, MD, USA, 2004.

[105] Lin Gu, Dong Jia, Pascal Vicaire, Ting Yan, Liqian Luo, Ajay Tirumala, Qing Cao, Tian He, John A. Stankovic an dRarek Abdelzaher, and Bruce H. Krogh. Lightweight detection and classification for wireless sensor networks in realistic environments. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, pages 205–217, San Diego, California, USA, 2005.

[106] Hui Song, Sencun Zhu, and Guohong Cao. Svats: A sensor-network-based vehicle anti-theft system. In *Proceedings of the 27th IEEE International Conference on Computer Communications*, pages 171–175, Phoenix, AZ, USA, 2008.

[107] Mohammad Rahimi, Rick Baer, Obmdinachi I. Iroezi, Juan C. Garcia, Jay Warrior, Deborah Estrin, and Mani Srivastava. Cyclops: In situ image sensing and interpretation in wireless sensor networks. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor System*, pages 192–204, San Diego, California, USA, 2005.

[108] Fung Po Tso, Lizhou Zhang, and Weijia Jia. Video surveillance patrol robot system in 3g, internet and sensor networks. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, pages 395–396, Sydney, Australia, 2007.

[109] Sukun Kim, Shamim Pakzad, David Culler, James Demmel, Gregory Fenves, Steven Glaser, and Martin Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, pages 254–283, Cambridge, MA, USA, 2007.

[110] Ning Xu, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. A wireless sensor network for structural monitoring. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor System*, pages 13–24, Baltimore, MD, USA, 2004.

[111] Lakshman Krishnamurthy, Robert Adler, Phil Buonadonna, Jasmeet Chhabra, Mick Flanigan, Nandakishore Kushalnagar, Lama Nachman, and Mark Yarvis. Deisgn and deployment of industrial sensor networks: Experiences from a semiconductor plant and the north sea. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor System*, pages 64–75, San Diego, California, USA, 2005.

[112] Younghun Kim, Thomas Schmid, Zainul M. Charbiwala, Jonathan Friedman, and Mani B. Srivastava. Nawms: Nonintrusive autonomous water monitoring system. In *Proceedings of the 6th International Conference on Embedded Network Sensor Systems*, pages 309–322, Raleigh, NC, USA, 2008.

[113] Asis Nasipuri, Luke Van der Zel, Ralph McKosky, Robert Cox, Hadi Alasti, Bienvenido Rodriguez, and Joseph Graziano. Wireless sensor network for substation monitoring: Design and deployment. In *Proceedings of the 6th International Conference on Embedded Network Sensor Systems*, pages 365–366, Raleigh, NC, USA, 2008.

[114] Geoffrey Werner-Allen, Patrick Swieskowski, and Matt Welsh. Real-time volcanic earthquake localization. In *Proceedings of the 4th International Conference on Embedded Networked Sensor System*, pages 357–358, Boulder, Colorado, USA, 2006.

[115] S. B. Eisenman, E.Miluzzo, N. D. Lane, R. A. Peterson, G. S. Ahn, and A. T. Campbell. The bikenet mobile sensing system for cyclist experience mapping. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, pages 87–101, Sydney, Australia, 2007.

[116] Jyh-How Huang, Saqib Amjad, and Shivakant Mishra. Cenwits: A sensor-based loosely coupled search and rescue system using witnesses. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, pages 180–191, San Diego, California, USA, 2005.

[117] Emiliano Miluzzo, Nicholas D. Lane, Kristof Fodor, Ronald peterson, Hong Lu, Mirco Musolesi, Shane B. Eisenman, Xiao Zheng, and Andrew T. Campbell. Sensing meets mobile social networks: The design, implemetation and evalaution of the cenceme application. In *Proceedings of the 6th International Conference on Embedded Network Sensor Systems*, pages 337–350, Raleigh, NC, USA, 2008.

[118] Aline Baggio. Wireless sensor networks in precision agriculture. In *Proceedings of ACM Workshop on Real-World Wireless Sensor Networks*, Stockholm, Sweden, 2005.

[119] Koen Langendoen, Aline Baggio, and Otto Visser. Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture. In *Proceedings of the 14th International Workshop on Parallel and Distributed Real-Time System*, Island of Rhodes, Greece, 2006.

[120] Tim Wark, Peter Corke, Pavan Sikka, Lasse Klingbeil, Ying Guo, Chris Crossman, phil Valencia, Dave Swain, and Gred Bishop-Hurley. Transforming agriculture through pervasive wireless sensor networks. *IEEE Pervasive Computing*, 6(2):50–57, 2007.

[121] Richard Beckwith, Dan Teibel, and Pat Bowen. Unwired wine: Sensor networks in vineyards. In *Proceedings of the IEEE Sensors*, pages 561–564, Vienna, Austria, 2004.

[122] Kok-Kiong Yap, Vikram Srinivasan, and Mehul Motani. Max: Human-centric search of the physical world. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor System*, pages 166–179, San Diego, California, USA, 2005.

[123] Jr. Mark L. McKelvin, Mitchel L. Williams, and Nina M. Berry. Integrated radio frequency identification and wireless sensor network architecture for automated inventory management and tracking applications. In *Proceedings of the 2005 Conference on Diversity in Computing*, pages 44–47, Albuquerque, New Mexico, USA, 2005.

[124] A. Mason, A. Shaw, A. I. Al-Shamma'a, and T. Welsby. RFID and wireless sensor network integration for intelligent asset tracking systems. Technical report, General Engineering Research Institute, Liverpool John Moores University, 2006.

[125] Loc Ho, Melody Moh, Zachary Walker, Takeo Hamada, and Ching-Fong Su. A prototype on RFID and sensor networks for elder healthcare: Progress report. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis*, pages 70–75, Philadelphia, PA, USA, 2005.

[126] Emulab Total Network Testbed. `http://www.emulab.net/tutorial/mobilewireless.php3`.

[127] John Hopkins Testbed. `http://hinrg.cs.jhu.edu/Main/NSLU2`.

[128] Kansei: Sensor Testbed for At-Scale Experiments. `http://ceti.cse.ohio-state.edu/kansei/`.

[129] Brent N. Chun, Philip Buoadonna, Alvin AuYoung, Chaki Ng, David C. Parkes, Jeffrey Shneidman, Alex C. Snoeren, and Amin Vahdat. Mirage: a microeconomic resource allocation system for sensornet testbeds. In *Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors*, pages 19–28, 2005.

[130] MoteLab: Harvard Sensor Network Testbed. `http://motelab.eecs.harvard.edu/`.

[131] NetEye. `http://neteye.cs.wayne.edu/neteye/home.php`.

[132] TKN Wireless Indoor Sensor network Testbed. `http://www.twist.tu-berlin.de/wiki`.

[133] Tutornet: A Tiered Wireless Sensor Network Testbed. `http://enl.usc.edu/projects/tutornet/`.

[134] VineLab Wireless Testbed. `http://www.cs.virginia.edu/~whitehouse/research/testbed/`.

[135] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, Berkeley, CA, USA, 2009.

[136] Jonathan W. Hui and David Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd International Conference on Embedded Network Sensor System*, pages 81–94, Baltimore, MD, USA, 2004.

[137] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the 5th Symposium on Operating System Design and Implementation*, pages 147–163, Boston, Massachusetts, USA, 2002.

[138] Kai-Wei Fan, Zizhan Zheng, and Prasun Sinha. Steady and fair rate allocation for rechargeable sensors in perpetual sensor networks. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, pages 239–252, 239-252, 2008.

[139] Dongjin Son, Bhaskar Krishnamachari, and John Heidemann. Experimental study of the effects of transmission power control and blacklisting in wireless sensor networks. In *Proceedings of the 2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, pages 289–298, Santa Clara, CA, USA, 2004.

[140] Gregory Hackmann, Octav Chipara, and Chenyang Lu. Robust topology control for indoor wireless sensor networks. In *Proceedings of the 6th ACM Conference on Embedded Networks Sensor Systems*, pages 57–70, Raleigh, North Carolina, USA, 2008.

[141] Nisheeth Shrivastava, Chiranjeeb Buragohain, and Divyakant Agrawal. Medians and beyond: New aggregation techniques for sensor networks. In *Proceedings of 2nd International Conference on Embedded Networked Sensor Systems*, pages 239–249, Baltimore, MD, USA, 2004.

[142] Christopher M. Sadler and Margaret Martonosi. Data compression algorithms for energy-constrained devices in delay tolerant networks. In *Proceedings of the 4th International Conference on Embedded Network Sensor System*, pages 265–278, Doulder, Colorado, USA, 2006.

[143] G. P. Halkes, T. Van Dam, and K. G. Langendoen. Comparing energy-saving mac protocols for wireless sensor networks. *Mobile Networks and Applications*, 10(5):783–791, 2005.

[144] Kyle Jamieson, Hari Balakrishnan, and Y. C. Tay. Sift: A mac potocol for event-driven wireless sensor networks. *Wireless Sensor Networks*, pages 260–275, 2006.

[145] Jan-Hinrich Hauer. Tkn15.4: An ieee 802.15.4 mac implementation for tinyos 2. Technical Report TKN-08-003, Telecommunication Networks Group, Technique University Berlin, 2009.

[146] Wei Ye. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transaction on Networking*, 12(3):493–506, 2004.

[147] David J. Malan. Implementing public-key infrastructure for sensor networks. *ACM Transactions on Sensor Networks*, 4(4), 2008.

[148] Prasanth Ganesan, Rannath Venugopalan, Pushkin Peddabachagari, Alexander Dean, Frank Mueller, and Mihail Sichitiu. Analyzing and modeling encryption overhead for sensor network nodes. In *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications*, pages 151–159, San Diego, CA, USA, 2003.

[149] Yufeng Lin, Ben Liang, and Baochun Li. Passive loss inference in wireless sensor networks based on network coding. In *Proceedings of the 28th IEEE Conference on Computer Communications*, pages 1809–1817, Rio de Janerio, Brazil, 2009.

[150] Jun-Hong Cui Zheng Guo, Peng Xie and Bing Wang. On applying network coding to underwater sensor networks. In *Proceedings of the 1st ACM International Workshop on Underwater Networks*, pages 109–112, Los, Angeles, CA, USA, 2006.

[151] Kyu-Han Kim and Kang G. Shin. On accurate measurement of link quality in multi-hop wireless mesh networks. In *Proceedings of the 12th International Conference on Mobile Computin and Networking*, pages 38–49, Los angeles, CA, USA, 2006.