uOttawa

L'Université canadienne
Canada's university

Simple Geometric Constructs for Routing and Boundary Detection in Sensor Networks

Marwan Fayed

Thesis Submitted to the Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements for the degree of Doctor of Phylosopy in Computer Science

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Abstract

Micro-sensor and radio technologies now permit the manufacture of cheap sensor or embedded devices deployable en masse. Applications appear in a diverse set of environments for far reaching applications including but not limited to structural monitoring, target tracking, and early warning systems. When deployed to create a sensor network, have no foreknowledge of their environment. In a network of this type traditional networking techniques are unsuitable. In general, sensing and embedded devices are shaped by four constraints: limited power supply, small memory, unattended operation, and the error-prone nature of wireless communications.

Our work is motivated by the hypothesis that within view of each node are geometric features that impact network characteristics and behaviour. The central objective in this thesis is to investigate the geometry of the network graphs. Doing so allows us to identify some of the unique features of the network that constrain larger problems.

We first propose boundary detection solutions using two well-known structures. First with the convex hull we build a localised heuristic, local convex view (lcv), that is designed on the premise that a node on the convex hull of a small region of the network is likely on the convex hull of the whole network. We show positive results via analysis and simulation and discover that the geometric properties are directly responsible for its resilience to error. We propose propose further the alpha-hull, whose structure can reveal details in the 'shape' of a set of points. We find that

by selecting the α-parameter carefully, it is possible to infer the network-wide α-hull from local communications and computations.

We also investigate the limits of routing according to left- or right-hand rule (LHR). Using LHR, a node upon receipt of a message will forward to the neighbour that sits next in counter-clockwise order in the network graph. When used to recover from greedy routing failures, LHR guarantees success if implemented over planar graphs. We identify network constraints that lead us to propose the Prohibitive-link Detection and Routing Protocol (PDRP) that can guarantee delivery over non-planar graphs. As the name implies, the protocol detects and circumvents 'bad' links. Our implementation of PDRP reveals the same level of service as face-routing protocols despite preserving most intersecting links in the network.

Acknowledgements

I would like to thank my mother for her endless encouragment, my sister for always setting me straight, and my father for his experience and advice. A very special thanks to Jeff Wheeldon and Paul Elliot, whose company during lunches was sometimes the only reason I came to do work. It would be an understatement to say that our exchanges were inspiring and invigorating. Finally, my unending gratitude to my advisor, Hussein T. Mouftah. I might never have completed this work without his unending patience and infinite faith.

Thank you.

Dedication

This work is dedicated to the memory of my father, Muhammad E. Fayed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Micro-sensor and radio technologies are permitting the manufacture of cheap sensor or embedded devices deployable en masse. Applications appear in a diverse set of environments for far reaching applications including but not limited to structural monitoring, target tracking, and early warning systems [3]. Each of these sensors is identical and, when deployed to create a sensor network, have no foreknowledge of their environment. In a network of this type traditional networking techniques are unsuitable [22]. Moreover, sensor networks demand a level of context awareness for which there was previously a limited body of previous work. In general, sensing and embedded devices are shaped by four constraints [77]: limited power supply, small memory, unattended operation, and the error-prone nature of wireless communications.

When researching solutions to context-awareness and routing problems in large sensor networks it is natural to work with the graph [1] that corresponds to network nodes and links. As such it is natural to look to disciplines in which graphs are well studied to find solutions to the problems faced by sensor networks. The body of knowledge generated by the graph and computational geometry communities has

[1] For our purposes we use 'graph' to refer to the network embedding, where the network graph is imposed upon a 2-dimensional plane.

been indispensable to the networking community. Still, the gap between theory and practice has yet to be bridged.

1.1 Motivations and Objectives

Our work is motivated by the hypothesis that within view of each node are geometric features that impact network characteristics and behaviour. The central objective in this thesis is to investigate the geometry of the network graphs. Doing so allows us to identify some of the unique features of the network that constrain larger problems. We emphasise to our reader the subtlety of this approach: It is common to shape or change the network to meet the needs of established solutions to related problems. In contrast, we seek to better understand network geometry so as to better shape the solutions to meet the demands of the network.

We offer as an analogy the urban sprawl that is endemic in North America. North American cities have been built around 'innovations' such as the automobile [51]. This is an example of tailoring the environment to a specific system. The contrasting approach would be to construct a city according to the needs of 'community', thereby meeting the constraints of its inhabitants. With respect to sensor networks the trend is to constrain nodes and links to meet the needs of a system; by contrast we seek to identify the constraints imposed by node and link features so that better systems may be designed.

1.2 Thesis Contributions

From the perspective described we approach boundary detection, and position-based routing. Each is considered a vital service for sensor-related applications. Problem summaries and contributions appear in the following sections. Detailed discussions are reserved for relevant chapters.

1.2.1 Boundary Detection

Context-awareness is increasingly important in wireless and sensor networks. When available, knowledge of position, nearby physical obstacles, or topological features, can be exploited to provide better communication protocols and deployment techniques in resource constrained environments.

Intuitively, many pure sensing applications benefit from knowledge of network boundaries ([8, 12, 24, 28, 71, 101, 110, 126]). Nodes along the outer edge of the network, for example, are assumed to be the best candidates for beacons in virtual coordinate constructions. Here the assumption is that the finest resolution in coordinates appear using a set of beacons that are furthest apart. Perceived network edges also may bound holes in the network or other regions of interest. Such regions may indicate physical boundaries or node failures due to environmental effects, so that additional nodes may be deployed.

We propose boundary detection solutions using two well-known structures. Our investigation begins with the convex hull, a structure that bounds a set of points in space. We build a localised heuristic, local convex view (lcv), that is designed on the premise that a node on the convex hull of a small region of the network is likely on the convex hull of the whole network. We show positive results via analysis and simulation and discover that the geometric properties are directly responsible for its resilience to error.

Our investigation continues using the generalisation of the convex hull, the alpha-hull, whose structure can reveal details in the 'shape' of a set of points. We find that by selecting the α-parameter carefully, it is possible to infer the network-wide α-hull from local communications and computations.

In these works we tailor both structures in a way that reveals the nodes and edges that bound the network.

1.2.2 Position-Based Routing

Position-based routing protocols have long been regarded as ideal for sensor networks. They are generally simple in operation, need only local communication, and store constant state.

The construction of network subgraphs appropriate for position-based (or geographic) routing protocols has, to date, remained a complex problem. These subgraphs are needed to recover from the local minima problem (see [2, 15]) that prevents delivery and plagues position-based protocols. Network subgraphs constructed using only 1-hop information risk inaccuracies that cause routing failures. If permitted to cooperate, nodes may construct a network subgraph that remedies any inaccuracies. Yet the energy needed to power the many needed rounds of communication risks being prohibitive in such a resource-constrained environment. The ideal wireless network subgraph would a) require only 1-hop information and b) acquire such information passively.

Our approach is to understand the causes for a position-based routing protocol to fail to recover from local minima and deal with those causes, directly. We have chosen to investigate the limits of routing according to left- or right-hand rule (LHR). Using LHR, a node upon receipt of a message will forward to the neighbour that sits next in counter-clockwise order in the network graph. (Alternatively, clockwise order if using right-hand rule.) When used to recover from greedy routing failures, LHR guarantees success if implemented over planar graphs; for this reason it is often called 'face-routing'. We note, however, that if planarity is violated then LHR is only guaranteed to eventually return to the point of origin. Our work seeks to understand why.

The result leads us to propose the Prohibitive-link Detection and Routing Protocol (PDRP). As the name implies, the protocol detects and circumvents 'bad' links. Our implementation of PDRP reveals the same level of service as face-routing protocols despite preserving most intersecting links in the network. The design of PDRP

was made possible by an enumeration and analysis of the challenges faced by LHR in general networks. It is our belief this approach may be used to tailor additional routing protocols to the constraints of the network.

1.3 Thesis Contents

To better place our work in context we discuss previous and related work in Chapter 2. In Chapter 3 we describe an algorithm to solve the boundary detection problem using convex hulls. Though it is heuristic in nature we find that it performs well in error-proned environments. We augment the boundary-detection algorithm using alpha-hulls in Chapter 4. An analysis of the network geometry reveals that, using the alpha-hull, an accurate and deterministic boundary will emerge. In Chapter 5 we turn our focus to position-based routing. In this chapter, an enumeration and analysis of the limits of left-hand rule leads to a protocol that that operates locally and preserves most network links. Finally, some concluding remarks appear in Chapter 6.

Chapter 2

Related Work

2.1 Introduction

A growing number of sensor network applications require point-to-point services. Intuitively, pure sensing applications, where environment is monitored or events tracked, require some geographical or geometric context for successful operation. In such a network data interpretation and management is often tied to node positions. In addition to traditional sensing applications are a growing number of proposed applications that require no knowledge of geography yet do require advanced point-to-point services ([36, 40, 102, 111]). Traditional Internet techniques are unsuitable in either setting [22]. In this chapter we focus on the merits and challenges of algorithms and protocols that provide boundary detection as well as point-to-point services through position-based routing, where forwarding decisions are made by maximising or minimising some function of node locations within a coordinate system. The focus of discussion is on those protocols suited to static sensor networks.

The remainder of this chapter is organised as follows: We begin in Section 2.2 with an overview of solutions to the boundary detection problem. There we discuss geometric, probabilistic, and topological approaches in Sections 2.2.1, 2.2.2 and 2.2.3,

respectively. A discussion of position-based routing protocols suitable to sensor networks begins in Section 2.3. Protocols where knowledge of sensor positions is assumed to exist prior to protocol execution appear in Section 2.3.1. While the presentation is cumulative, each section may be read independently of the others. In Section 2.3.2 we present protocols where no a priori knowledge of position exists; amongst such protocols the routing element is coupled to the localisation mechanism. We summarise our discussion in Section 2.4.

2.2 Boundary Detection

Our work appears amid a growing body of research on boundary detection. We focus on works that are distributed or localised. Existing work may be classified according to the taxonomy presented in [119] as either geometric, statistical, or topological in nature.

2.2.1 Geometric Solutions

Geometric solutions to the boundary identification problem use the positions associated with each node. Our work falls into this category. To our knowledge the work by Fang *et al.* [23] is the first such work. In it a geometric relationship is described, and labelled as the *tent-rule*. The tent-rule is used by each sensor node in the network to determine whether it lies on the boundary of a routing hole or of the network. We exemplify the tent-rule using the pictorial representation that appears in Figure 2.1. At each node x neighbours are sorted angularly. Now consider for every pair of neighbours u and v where $\overrightarrow{xu}$ is left of $\overrightarrow{xv}$, the perpendicular bisectors of each link (labelled $b1$ and $b2$ in Figure 2.1). If the intersection of the bisectors falls outside of the range of x then x has no neighbours closer to the region bounded by the communication range and the bisectors (ie. the shaded region in Figure 2.1). Note this method indentifies

boundary regions rather than nodes. This is particularly suitable in sensor networks where sensing regions may be of particular interest.

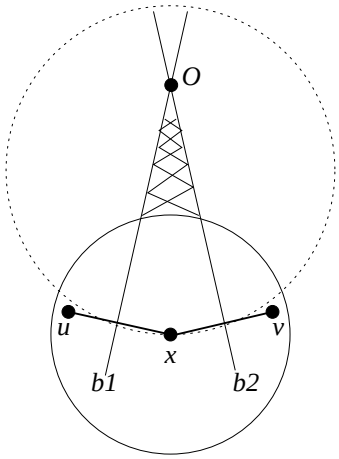


Figure 2.1: Tent Rule [23]: Node x has no neighbours closer to shaded region as determined by the intersection of bisectors of adjacent links.

As stated previous, every node must evaluate its neighbourhood using the tent-rule. Only at nodes where the tent-rule fails, may packets get stuck. Upon detecting a failure, stuck nodes are responsible for initiating the following hole-mapping process. Say node x sees that it abuts a boundary. It marks and sends a packet for for discovery to the neighbour whose connecting edge is immediately counter-clockwise, or left, from $\overrightarrow{xd}$. Each node which receives the discovery packet appends the newly traversed edge, then forwards the packet along the next counter-clockwise edge.

During the mapping process a discovery packet may traverse an edge that intersects with another edge that was previously traversed. Where edge intersections occur care must be taken. A traversal according to left-hand rule will always return to the source node, yet intersections may cause the traversal to be led away from the region that requires a map as can be seen in Figures 2.2(a) and 2.3(a). If an intersection appears during a traversal of nodes $t_0 \cdots t_{k-1}$ then we can say that $t_j t_{j+1}$ intersects $t_i t_{i+1}$, with $j > i$. There are only two such cases (the proof of which is

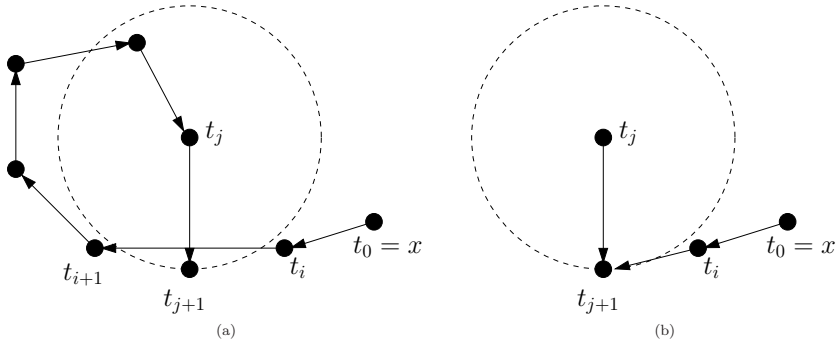


Figure 2.2: "Inside" intersections, before and after pruning.

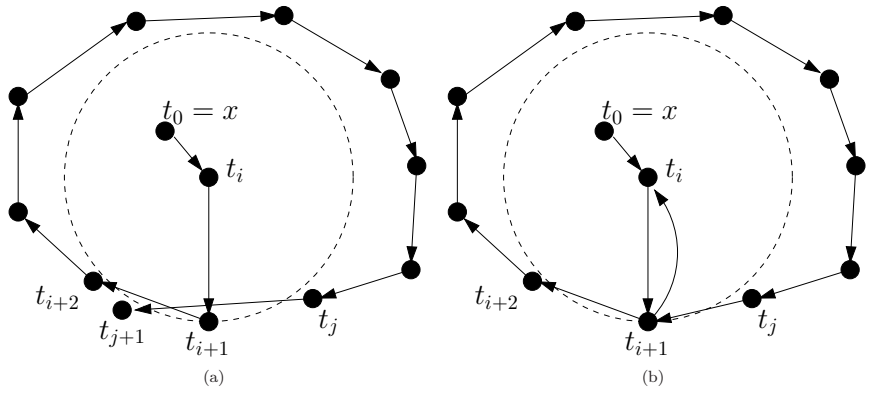


Figure 2.3: "Outside" intersections, before and after pruning.

available in [23]).

- *Inside Intersections* where node t_j is not visible to nodes t_i and t_{i+1}. Upon detection of this intersection node t_j replaces segments $t_{i+1}t_{i+2}\cdots t_jt_{j+1}$ with $t_it_{j+1}tj$, before forwarding to the node that next appears in counter-clockwise order from t_{j+1}. The effect of this pruning may be seen in Figure 2.2.

- *Outside Intersections* where node t_i is not visible to nodes t_j and t_{j+1}. Upon detection of this intersection node t_j ignores t_{j+1} and forwards instead to t_{i+1}, thereby ignoring the intersection all together. This example may be seen in Figure 2.3.

The traversal terminates upon its return to x, having recorded the path P that maps the complete stuck region. Once known, a recovery path is shared with all nodes along the path so that resources required in processing and storing discovery packets may be avoided in the future.

2.2.2 Probabilistic Solutions

Probability distributions underlying network deployments have been used to formulate statistical solutions. One solution, proposed by Fekete *et al.* [26], relies on the idea that nodes close to network boundaries have fewer incident edges in the network graph than internal nodes. The underlying assumptions rest with the fact that part of the area covered by the communication range of a boundary node rests outside of the network. Initially the algorithm initiates a leader election process and constructs a network-wide communication tree. Statistical methods are then used by the leader to derive suitable thresholds μ for density and α for area, to separate edge nodes from internal nodes using the node degrees. These thresholds are used by remaining nodes when counting neighbours and links to determine boundary status. Example results may be seen in Figure 2.4.

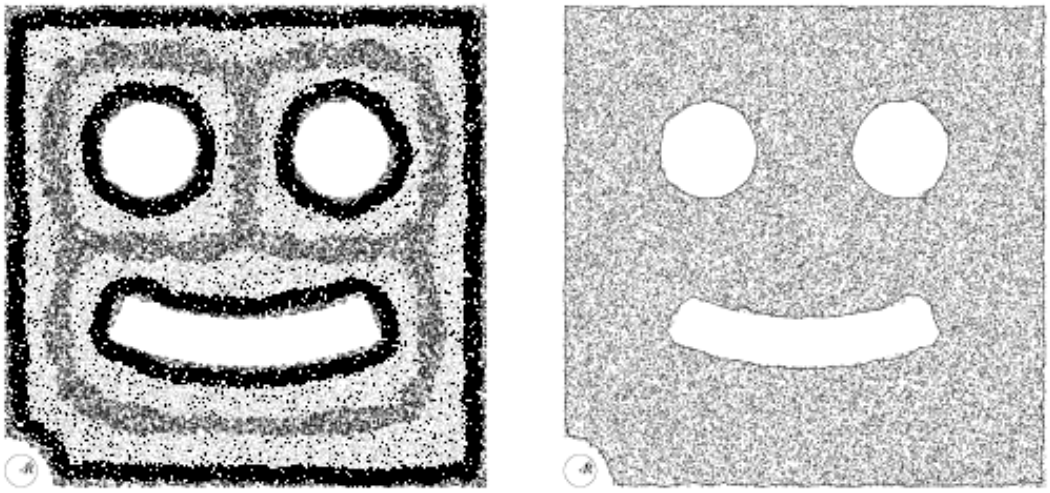


Figure 2.4: (Left) Boundary and near-boundary nodes; (Right) Loops determined to enclose boundaries.

In [104] a similar statistical separation is proposed. Boundary nodes are separated from internal nodes by using a 'centrality' measure which counts the number of shortest paths that pass through a node. Higher centrality values occur among internal nodes. As above, the network elects a leader and constructs a communication tree. The leader aggregates measurements from all nodes, it infers a centrality measure, and returns the result to the network. A node declares itself to be on the boundary of the network if its own measurements fall within the threshold given.

Statistical solutions generally hinge on uniformly distributed networks and exceedingly high densities. Our work in Chapters 3 and 4 shares in the view that nodes at the boundary exhibit unique characteristics. Unlike statistical methods, our approach is localised, is shown to be resilient to the underlying distribution, and performs well in lower densities environments.

2.2.3 Topological Approaches

Topological solutions appear in [31, 64, 101]. Kröller *et al.* propose a combinatorial approach in [64]. It is the only deterministic work of which we are aware to produce correct results without relying on the unit disc graph model (where all communication ranges are normalised). This solution comes at a high cost: It deals with complex combinatorial structures in a distributed manner. Rao *et al.* [101] suggest a single-beacon broadcast solution with a total messaging cost is similar to localised approaches such as ours. Their work begins with a 'hello' message from a single beacon located at least 1-hop from the network edge. A node decides it is on the edge if it finds itself to be furthest from the beacon amongst nodes in its 2-hop neighbourhood. While only heuristic in nature this method suffices for their purpose, which is to construct a coordinate and routing system.

Funke [31] later proposed extensions to a similar idea. The idea, shown in Figure 2.5, relies on the observation that contours 'rings' of the network that are described by hop-distances to a beacon, are broken when encountered by a network boundary. This project was later refined by Funke *et al.* in [32]. Topological inference is used by Wang *et al.* [119] to detect internal boundaries. Their detection method works by identifying the distinct portions of similar paths that span the network.

Zhang *et al.* provide a planarization of the network graph that requires neither uniform communication range nor node locations (see later, Chapter 2.3.1). They propose a progressive construction of trees and their arrangement as bipartite graphs. By planarizing each bipartite graph recursively, the faces of network boundaries emerge as a natural consequence. Finally, somewhat related is the contour tracking project [128]. Here the authors identify the boundaries of a binary event. A binary event may be, for example, a chemical concentration beyond some threshold or a set of grouped targets. When tracking such events there are 'binary' nodes, labelled black and white, with entire neighbourhoods either inside or outside of the events, respectively'. Using

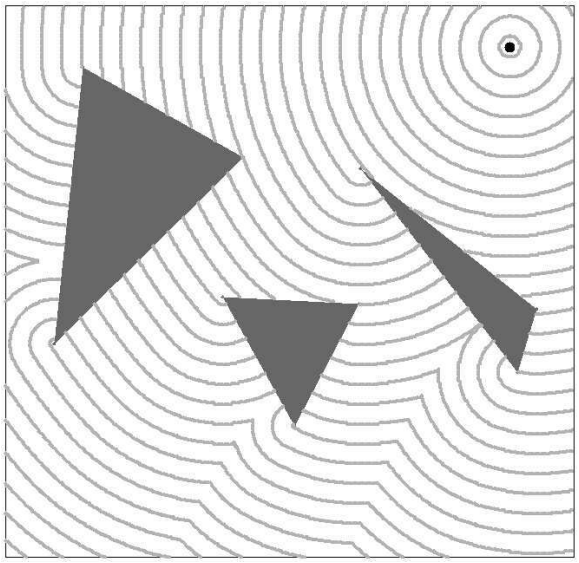


Figure 2.5: One beacon, three objects and the broken contours that are induced.

limited scope broadcast it is possible to identify the contour consisting of 'grey-area' nodes, nodes with some black neighbours and some white.

2.3 Position-Based (Geographic) Routing

In sensor networks IP-like routing techniques face scalability issues since node identifiers, if available, share no topological or geographical similarities and hence, no addressing information. In addition, IP-like routing requires global cooperation and dissemination of information which places undue demand on the energy constraints that are inherent in such devices.

Current point-to-point routing solutions under investigation in the research community fall into one of two classes, on-demand routing [50, 98, 99], and position-based (often referred to as geometric or geographic) routing [10, 30, 55, 66, 101]. Among the former class routes are found as needed, often without any prior communications. These methods consume network resources and are known not to scale. In

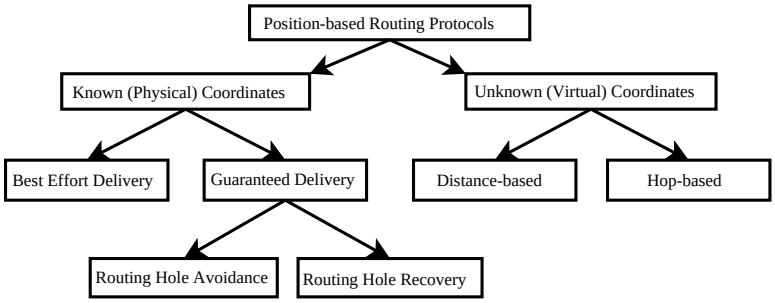


Figure 2.6: A classification of position-based routing protocols

the latter category, geographic routing decisions are based on relative locations of sensors, and are generally greedy in nature (eg. nodes forward to neighbours that are geographically closer to the destination).

In position-based routing the next hop is decided by evaluating the coordinates assigned to neighbouring nodes relative to the coordinates of the destination and the current node. For example, a node may choose to forward a message to its neighbour that further minimises the remaining Euclidean distance to the destination, or that maximises the savings in energy. Position-based routing is particularly attractive due to its scalability. In the best case a node need communicate only its position and only store information regarding its immediate one hop neighbourhood. Thus the storage, length of communication, and data over which decisions are made, is $O(1)$.

However, position-based routing is not without its drawbacks and challenges. When positions are known, for example, messages may get trapped in local minima where no neighbouring nodes further optimise the decision criteria. When positions are not known, there exists the additional challenges of establishing and maintaining some coordinate system in order to determine node positions in the network. The localised nature of position-based routing may also render algorithms blind to obstacles that might cause routing decisions to fail.

Until recently the focus of position-based routing has been on the design of protocols that assume the existence of a globally known coordinate system, for exam-

ple, through pre-programming or the Global Positioning System (GPS). More recent advances demonstrate that it is possible to establish a network-centric coordinate system, the design of which is often coupled to the routing mechanism. Our attentions concentrate on families of algorithms that are thought to be feasible, and in the case of [55, 59], have implementations in development.

Position-based routing algorithms for static sensor networks may be classified according to the chart in Figure 2.6. This chart matches the categorisation of position-based routing protocols used to organise this chapter. Our discussion continues in the next section by introducing the topic that follows the left branch of our categorisation, those routing protocols that assume sensors are aware of their physical coordinates.

2.3.1 Routing with Known Positions

It is natural, when investigating position-based routing algorithms for sensor networks, to assume the a priori knowledge of location information via pre-programming or GPS-like services. Thus, no efforts are made by these protocols to establish location within a network. Current methods make some necessary assumptions. First, links are bidirectional: if node x can receive messages from node y, then y can receive messages from x. In addition, communication models often adhere to the simple unit disc graph (UDG) model where all communication ranges are normalised to some range r. In the UDG neighbours are pairs of nodes separated by a Euclidean distance $\leq r$. The UDG model is relaxed later in our discussion. Feasible schemes must, at a minimum, prove to be loop-free and scalable. We demonstrate some of these subtleties by starting with a discussion of the naive approaches to position-based routing.

Naive Forwarding Mechanisms

Research into scalable forwarding methods for sensor networks has explored a variety of forwarding schemes. Scalability is maintained by keeping knowledge only of the

nodes in communication range, and choosing the next hop based on this knowledge. All position-based schemes share a common theme: the next hop is determined by maximising or minimising some criteria associated with local nodes' positions. We call this the *progress* criteria.

Notions of "Progress". We begin with a formal definition of *progress*. Say a node s holds a message to be forwarded to a destination location d (see Figure 2.7), and has knowledge of all node locations within its communication radius. Then we have the following definition used to better understand the behaviour of position-based forwarding schemes.

Definition 2.3.1 The *progress of a node x en route from sender s to destination d is defined as the orthogonal projection of the location of x onto the line $\overline{sd}$.*

We use Figure 2.7 to better demonstrate various notions of progress as discussed below. In this figure sensor node s holds a message destined for sensor node d. The large circle centered at s represents the communication range of s; thus all nodes inside the circle are neighbours of s. The dotted horizontal line $\overline{sd}$ is the line onto which orthogonal projections are made in order to evaluate and compare progress criteria.

The first forwarding mechanism based on the idea of progress was proposed by Takagi and Kleinrock [115]. In their *Most Forward within Radius* scheme, or MFR, the node with the greatest progress is chosen to be the next hop. Referring to Figure 2.7 we can see that node m projects furthest onto the line that joins s and d. It is important to note that m provides the greatest amount of progress though it is not the node closest to the destination. The motivation behind the use of this myopic routing strategy was to allow for tractable analysis. Recall that progress is defined as a projection onto a line. Clearly, because we are working with projections on a line, the dot product of $dm \cdot ds$ will be minimal over all other neighbours of s when

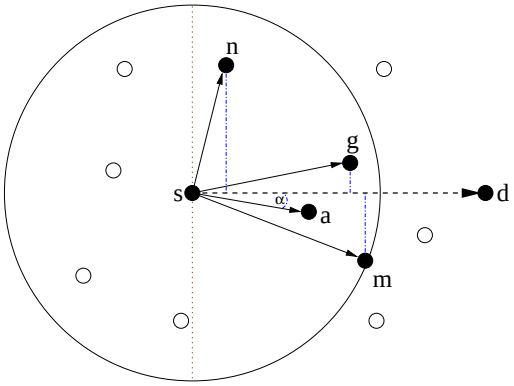


Figure 2.7: progress

using MFR. This simple notion allowed the authors of [115] to determine an optimal neighbourhood size for their specific problem.

The converse scheme *Nearest Forward Progress*, or NFP was later proposed in [47]. In this work the node with the minimum progress, depicted as node n in Figure 2.7, is selected as the next hop. This approach may seem counter-intuitive, but consider that if the broadcast range is variable then this method has the least probability of collision as well as improved energy savings.

More recently, the direction of nodes was proposed as a criteria for progress in [63]. The node chosen to be the next hop is the node that is closest to the direction of the destination. Referring to Figure 2.7 we can see that node c, with angle $\angle csd$, is smallest among all of the neighbours of s. Thus, the goal is to minimise the change in direction from the source to the destination.

The DREAM [7] and LAR [62] projects, simultaneously proposed, use an idea similar to compass routing. However, these two approaches are best suited in networks where nodes are mobile. (Despite their application in mobile environments, we provide an overview for completeness.) The node that holds a message m with destination d, calculates an angular range where the message must be forwarded. The angular range is calculated using i) the circle centered at d with radius equal to the

maximum movement of d since the last update and ii) the tangents from the current location to the aforementioned circle. All nodes within this angular range are sent a copy of the message. Clearly, the success of both methods relies on some knowledge of the global network, as well as duplication of messages. Such methods fall outside of the domain of position-based routing.

Each of MFR, NFP, and Compass Routing is myopic. They are localised algorithms that require knowledge only of the immediate neighbourhood. Unfortunately their global behaviour is such that none of these approaches can claim to be loop-free. Still, consider that their forwarding decisions attempt to optimise some local criteria, the effect of which is to approximate the shortest Euclidean path between the source and destination. The shortest path may be approximated using another approach that referred to as *greedy forwarding*. Greedy forwarding is a localised forwarding scheme whose express goal is to traverse the Euclidean shortest path. It is this approach on which most research and development, where there is a known and underlying Euclidean coordinate system, has focused.

Greedy Forwarding. Greedy forwarding was first proposed in [27] as a routing protocol for wired networks. Referred to as Cartesian routing, the next hop was chosen to be the neighbour that is closest to the destination. As this work predated GPS and other localisation services, knowledge of the global topology was required. The same idea has been re-applied in wireless network settings as the foundation of an innumerable routing schemes and algorithms. It is known to behave especially well in dense wireless networks such as those envisioned in many sensor networks.

Greedy forwarding works as follows. Say node s has the neighbourhood $N = n_1, ..., n_k$ of size k where each n_i is a potential next hop in a traversal which passes through s. Any message that arrives at s has embedded within it the destination d. The greedy approach says that the successor to s will be the neighbour n_i that minimises the Euclidean distance to d. In Figure 2.7 we can see that node s will

select g, the node that most further reduces the distance to the destination among its neighbours.

It can be shown that greedy forwarding is loop-free (using the simple fact that every hop reduces the distance to the destination). The delivery rate of greedy forwarding is known to be quite high in dense networks, but diminishes quickly as the density falls([25]). The problem is that a greedy path may terminate in a routing hole, or void, as shown in the next section.

Routing Holes

Greedy forwarding may be loop-free, but delivery is not guaranteed. A consequence of greedy forwarding is that a route may terminate at local minima, where nodes have no neighbours that further reduce the distance to the destination [2].

We refer our reader to Figure 2.8 which depicts the types of local minima that may occur: the smaller dotted circle represents the communication range of node S, and the larger circle centered at D is used to show that all neighbours of S are further from D than S, itself. Consider a message destined for node D reaches a minima at node S. There are two cases to consider. The first, as shown in Figure 2.8a, occurs where neighbours of S make no progress towards D according to Definition 2.3.1. This is the obvious case. Less obvious is the case where neighbours of S may actually make progress towards the destination, yet increase the distance from the current location. This example is demonstrated in Figure 2.8b where S clearly lies within the circle centered at D, while the neighbours of S, x and A, lie outside of the same circle.

These local minima are commonly referred to as voids, holes, or stuck regions. Their occurrence largely determines the performance of greedy forwarding, whose performance varies with network density and distribution. (In 3-dimensional environments their impact may be less sever [1].) A robust sensor network routing

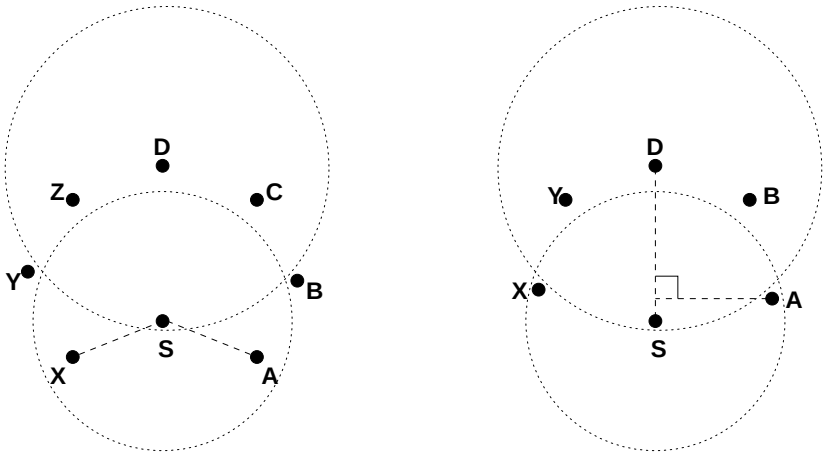


Figure 2.8: Neighbours of stuck nodes may or may not make progress.

protocol must perform well despite the occurrence of local minima. We proceed in our discussion with proposed solutions to the routing hole problem.

Algorithms for Recovery from Routing Holes

Greedy forwarding is known to perform well in sufficiently dense networks ([55]) yet there are no delivery guarantees. Many sensor network applications are loss-sensitive and have little to no tolerance for undelivered information. Thus greedy forwarding schemes aiming to guarantee delivery demand that routing voids be circumvented or avoided. Work in [25] presents evidence that the frequency and impact of routing holes is manageable. Current solutions to this problem are generally categorised as either broadcasting, planar subgraph methods, or hole mappings, discussed below.

Broadcasting, BFS, and DFS Approaches The naive solution to the routing hole problem is simply to broadcast some number of hops from the stuck node until a node is found that is closer to the destination. Two of the first solutions which aimed to be more efficient were the Geographic Routing Algorithm (GRA) in [100] and the

Optimal Transmission Ranges (OTR) approach in [113].

The authors of GRA implement greedy forwarding as the primary means of transport. If, however, a message reaches a stuck region GRA launches a route discovery packet from the stuck node. The route discovery itself takes the form of a breadth-first search amounting to a limited broadcast, or a depth-first search that eventually produces a single acyclic path. Each node visited during the recovery phase appends its own location to the recovery packet, and all discovered paths are stored in routing tables. In networks of size n routing tables are found to have a mean size of $O(L \log n)$ where L is the average path length between two nodes. In addition, the authors provide some mechanisms for dealing with location inaccuracy, node failure, and node mobility.

The goal of OTR differs in that it creates Quality of Service (QoS) paths through the network. The necessary characteristics are collected and disseminated during path traversals, which occur from s to d via the depth-first search method. Routing tables are not used, but instead each node stores the next and previous hop for each packet until the status of that packet can be confirmed. Such methods face scalability issues as traffic volume grows, and are additionally challenged where acknowledgements are not returned.

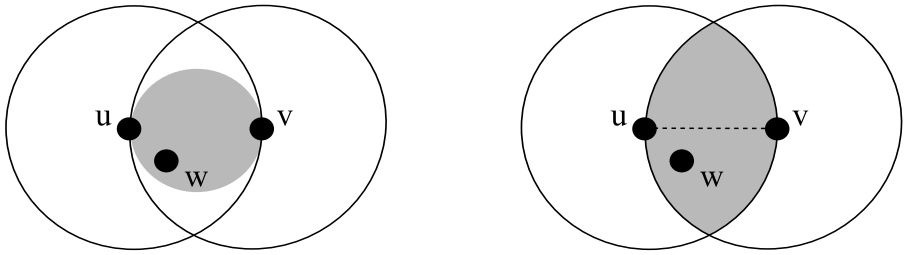
Hole Recognition and Mappings A second approach to the recovery phase is to recognise and map the hole boundaries in advance. This is the subject of work in [23, 26, 31]. Described in detail in Chapter 2.2.1 is the boundhole algorithm in [23]. As an alternative the boundhole algorithm may terminate upon reaching some node u such that $|\overline{ud}| < |\overline{xd}|$. If boundhole is to terminate early then every recovery packet must have encoded within it the subpath $p \in P$ traversed from x. We call p the escape path. Clearly, the number of hops in $|p| < |P|$, otherwise x could not be a stuck node. Experiments in [25] reveal that the escape path p is significantly smaller than the path P that bounds the void region; perhaps enough to warrant early termination when a

stuck region is encountered instead of a complete mapping.

Hole recognition, in general, has thus far received little attention in the research community. Additional algorithms are proposed in [26] and in [31]. The former uses a statistical approach to recognise holes in very dense, uniformly distributed networks. The latter study is the first where the hole recognition algorithm does not rely on location information. Neither provides methods for circumventing the stuck regions, yet it is important to recognise that hole-recognition is an important problem in and of itself. More recent work in [84] tries to predict rather than recognise holes, while [34, 44, 103] investigate methods that identify holes using passive listening methods.

Planar Subgraph Methods The recovery mechanisms so far described all rely on resources that may be taxing on sensing devices given energy and scheduling constraints. DFS, BFS, and mapping methods, for example, require storage either in-device, in-packet, or both. Also, their communication in the worst case, amounts to a limited broadcast. Such communication requires little to no additional memory, but demands additional energy consuming transmissions. Drawbacks such as these often render broadcast-like protocols unsuitable for point-to-point services in sensor networks. To solve this challenge many projects have investigated the restriction of routing to subgraphs of the original network graph. This class of algorithm is exemplified by one very desirable feature: Such algorithms are stateless, ie. a node requires no knowledge of the network outside of its own neighbourhood, yet is able to guarantee delivery. This class all share a simple characteristic: they rely on the construction of the network's planar subgraph. For our purposes a planar subgraph is one which contains all the vertices of G, but where edge intersections occur only at a vertex. (Rather, no edges overlap.)

The most prominent and best known recovery algorithms route around the routing hole face (or perimeter) in the planar subgraph. This method is equivalently



(a) Gabriel Graph Construction (b) Relative Neighbourhood Graph Construction

Figure 2.9: Localised constructions of planar subgraph.

known as *face routing* ([9, 10]) and *perimeter routing* ([55]). Face routing was first proposed by Bose et al. in [9] with some theoretical bounds. Karp et al. independently proposed an identical mechanism in [55] but with work on a MAC-compatible implementation. Variants have since emerged addressing, for example, theoretical bounds in [67, 66, 69]. In [52], face-routing is augmented into a "select-and-protest" reactive protocol in order to reduce the information required to planarize the graph.

The inspiration behind these methods comes from the application of a simple rule known to guarantee escape from a maze. In a maze, by keeping one hand against the wall at all times one is guaranteed to find an exit, or return to the point of origin if no exit exists. This is referred to as the left- or right-hand rule. This works because of a salient feature of maze construction: if we represent the walls of the maze as edges in a graph, and the intersection of those walls as vertices, then the resulting graph is planar, where no edges intersect.

Wireless network graphs may consist of intersecting edges so it is necessary for planar subgraph method to prune edges from the network graph so that it is planar and so that it remains connected. Gabriel Graphs (GG) and Relative Neighbourhood Graphs (RNG) are planar graphs whose constructions are localised, a characteristic particularly suitable to sensor environments. Intersecting edges are eliminated by

connecting pairs of nodes through *witness* nodes, if such a node exists in a common region.

We refer the reader to Figure 2.9 for examples of each construction where large circles centered at nodes u and v represent the communication ranges of nodes u and v, respectively. The GG construction is depicted in Figure 2.9a. For any pair of neighbours u, v, if a witness node w exists within the circle whose diameter is $|\overline{uv}|$ (shown as the shaded region in Figure 2.9a), then $\overline{uv}$ is removed from the graph. Similarly for the RNG in Figure 2.9b, if w appears within the intersection of the two circles centered at u and v with radii equal to $|\overline{uv}|$ (again shown as the shaded region), then $\overline{uv}$ is removed from the graph. In each, pruning is valid since communication may continue through w. In [9, 10] and later in [55], it was shown that if the unit disc graph is connected then the intersections with Gabriel graph $UDG \cap GG$, and Relative Neighbourhood graph $UDG \cap RNG$, remains connected.

Face- and perimeter-routing techniques choose to route greedily whenever possible. The recovery phase is initiated only when a message gets stuck at some node. Upon receipt of message m destined for node t, node s inspects the message to reveal it either in *greedy* or *recovery* mode. The corresponding algorithms, executed at each node, are listed in Algorithms 2.1 and 2.2.

Algorithm 2.1 Greedy Mode

1: let s be the current node
2: let w be the neighbour closest to t
3: **if** $(w, t) < (s, t)$ **then**
4: forward m to w
5: **else** {no such w exists; m is stuck at s}
6: mark packet as *recovery* with location of s
7: forward to neighbour that is left of $\overrightarrow{s, t}$
8: **end if**

Algorithm 2.2 Face/Recovery Mode

1: let u be the node from which m was received
2: let v be the current node
3: let w be the neighbour closest to t
4: **if** $(w,t) < (s,t)$ **then**
5: mark packet as *greedy*
6: forward to w
7: **else**
8: forward to neighbour that is left of $\overrightarrow{v,u}$
9: **end if**

Algorithms 2.1 and 2.2 assume a stuck node has first constructed the portion of the planar sub-graph that occurs within its view, as above. If in greedy mode a packet is forwarded according to Algorithm 2.1, where a sensor forwards to the neighbour closest to the destination. If no such neighbour exists then the sensor node forwards according to Algorithm 2.2. Once stuck, the message m is marked *recovery* and is forwarded to the neighbour that appears first in a counter-clockwise direction. While in recovery mode each sensor that receives the message first checks for a neighbour closer to the destination than the point at which the message was marked *recovery*. Returning to Figure 2.8a, we can see s is a stuck node. In the case of left-hand rule, $\overline{sx}$ is left of $\overline{sd}$. The recovery path sxy terminates upon finding z since z is closer to d than s, where recovery began.

One special case occurs where an edge $\overline{uv}$ intersects $\overline{sd}$ during recovery. The solution is left as an exercise.

It has since been shown that 'Hello' messages may hinder network performance [43]. This is addressed in face-routing directly by [14] and more generally in [34, 44, 103]. Further work in [124] reduces the path length during the recovery phase.

These algorithms are especially suited to sensor networking environments: they

guarantee delivery, are localised in their operation, and are stateless in the sense that they require no information outside of the location of their neighbours. However, we see in the next section that planar subgraph methods are not without their drawbacks.

Drawbacks and Challenges of Planar Methods

Planar subgraph methods, while promising, face three major challenges before their deployment may be considered feasible. First, planarization assumes locations are accurate, an assumption that may be untrue. Second, the localised nature of the planarization process means that one sensor node may be blind to environmental obstacles that are visible to neighbouring sensors. Finally, the unit disc model on which these algorithms are constructed is a poor representation of the real world. We use this section to touch on each of these challenges.

Routing protocols that operate over pre-established coordinate systems are generally designed under the assumption that location information is accurate. This assumption is challenged by real-world limitations. GPS, for example, offers high resolution localisation, but is subject to line-of-sight constraints rendering it ill-suited to underground, underwater, and under-coverage applications, but to name a few. Furthermore, the added expense incurred by supporting GPS in all nodes is restrictive. Many proposals exist to resolve this issue by supporting some small location-aware infrastructure, [45, 48, 79, 94] but to name a few, from which all other nodes may learn their locations; yet even these fall prey to poor resolution and estimation errors.

There are cases where localisation and estimation errors have little adverse effect. For example, in [41] greedy routing was evaluated in simulated networks with localisation errors. The results show that the performance of greedy routing is largely unaffected by inaccuracies up to 40% of the radio range. Conversely, there are contexts in which localisation errors can be destructive to correct protocol operation [56, 108, 109]. One such example is demonstrated in [57] which models and eval-

uates location error on face-routing techniques; here we find that errors of as little as 20% of the communication range caused high packet drop rates, non-optimal path selection, and looping. These experiments were reinforced in [107] which investigates the types of location errors that lead to performance degradation in face-routing. In doing so the authors were able to suggest a simple check, consisting of mutual agreement between nodes, to resolve many of the problem cases. It may also be possible to reduce the impact of error using optimization techniques [70].

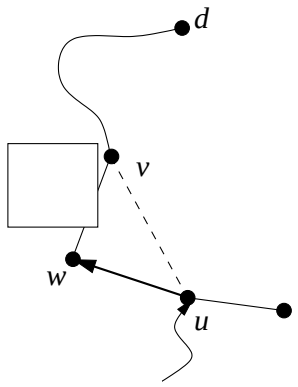


Figure 2.10: Planar methods may fail without inter-neighbour link information.

In addition to position errors, a second challenge faced by planarization processes lies in its best feature, that all operations are local and occur without need for negotiation with neighbouring nodes. The level of localisation inherent in the planarization process leaves open the possibility for incorrect output. Specifically, because a node planarizes its neighbourhood using only the locations of its 1-hop neighbours, a node may assume links exist where they do not. Consider the example in Figure 2.10 where a packet destined for node d gets stuck at node u. The planarization of the neighbourhood of u removes the link $\overline{uv}$ believing v is reachable via w. Since u is unaware of any obstacles or interference between w and v, u's planarization of its neighbourhood is incorrect. This phenomenon was first demonstrated in [54] and [86].

Finally, we must address the likelihood that the unit disc graph correctly repre-

sents wireless network graphs. The quasi-unit disc graph has been proposed in [89, 65, 68]. Experimental evidence in [121, 125] suggests that radio ranges are inconsistent and irregular. Experiments in [60] conducted on two sensor networks further demonstrate and enumerate the difficulties that occur because the unit disc assumption is violated. The experimental evidence is used to design a protocol that corrects the failings of the Gabriel and Relative Neighbourhood Graph constructions. Protocols that avoid the unit disc assumption are discussed in the next section.

Guaranteed Delivery in the Real World

Face-routing algorithms are attractive because they are localised and efficient. Yet as previously discussed, they are known to be ill-suited to physical environments. There are two reasons for this. First, the Gabriel and Relative Neighbourhood Graph constructions rely on the assumption that all communication range in the network is identical and uniform (the attributes associated with the UDG model). Moreover, these distributed constructions are unable to resolve links broken by obstacles or interference. Recent breakthroughs have begun to surmount the impracticalities of face-routing while maintaining delivery guarantees [59, 74].

The first known protocol to guarantee delivery over a global coordinate system without planarization or the use of left-hand rule is the Greedy Distributed Spanning Tree Protocol (GDSTR) algorithm in [74]. GDSTR builds on the fact that any message can be successfully delivered via depth-first search if the network is connected via a spanning tree. (This fact alone does not solve the problem: delivery would be inefficient, needing up to $2n - 3$ hops.) The authors in [74] describe a new type of spanning tree, the *hull tree*, to route more efficiently. A hull tree is a spanning tree with one added piece of information: each node records the convex hull that contains all of its descendants in the tree. (The convex hull of a set of points is the smallest polygon that contains all the points.) In GDSTR forwarding occurs greedily,

as with most position-based protocols. If a message reaches a void, a recovery mode is initiated where convex hulls are used to determine the regions of the network that contain unreachable destination. This information is used by GDSTR to route along the spanning tree to forward to the appropriate convex hull. If a node is found en route that is closer to the destination than the node where the message was stuck, then GDSTR returns to greedy forwarding. GDSTR is known to scale well as the neighbourhood size grows. Furthermore, the use of multiple hull trees adds fault-tolerance to the network and if multiple trees are rooted at opposite ends of the network, routing efficiency improves.

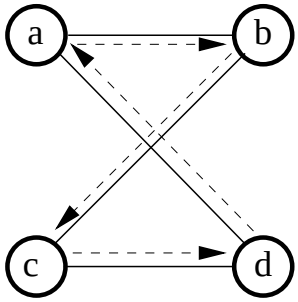


Figure 2.11: CLDP probes links using left-hand rule.

The Cross-Link Detection Protocol (CLDP) proposed in [59], and later improved in [39], circumvents voids by face-routing. It uses left-hand rule over a planar subgraph of the network; its design however, is motivated by the observation that routing difficulties in planar subgraph methods arise, in part, due to the constructions themselves. (Recall from previous that successful local planar subgraph constructions rely on the unit disc graph.) For this reason, CLDP proposes an alternate construction of planar subgraphs that assumes only that links are bidirectional. CLDP operates in a distributed fashion, exchanging some localised operation for accurate information. The idea behind CLDP is that each node is able to probe the vicinity for intersecting links. A probe packet is initialised with the endpoints of the first link to be probed. Figure 2.11 shows the simplest example of a probe traversing a graph using left-hand

rule. Say a probe starts at node d for link (d, a). Sensor node a then forwards to b as determined by left-hand rule. When the probe reaches node b, the intersection (b, c) is recorded, before the probe packet continues its traversal. (Recall that a traversal according to left-hand rule will eventually return to its starting point.) Prompted by the return of the probe packet, d proceeds to prune links. Figure 2.11 depicts only the naive case; the graph remains connected following a removal of either of the two intersecting edges. We leave as an exercise the identification of remaining cases and the care that must taken when pruning links to keep the graph connected. Furthermore, to avoid the slow process of scheduling serial probing by neighbouring nodes, a system for concurrent probing is proposed. Concurrent probing is achieved by implementing a mechanism to 'lock' links so that no more than one link is removed at a time from any vicinity. CLDP is one of very few protocols to have been implemented on testbeds [59]. The associated communication complexities and storage costs revealed in this process (see [60]) are motivation to develop alternative approaches to guarantee delivery.

Protocols such as CLDP and GDSTR, in order to be feasible for physical networks, sacrifice efficiency for accuracy. CLDP requires negotiation within each neighbourhood in order to prune appropriate links, and GDSTR must broadcast information to construct and maintain its hull trees. It remains an open question whether such trade-offs are a necessity.

2.3.2 Routing with Unknown Positions

By nature of its name position-based routing protocols may be thought to assume that position location is available for use by the routing protocol. Often this information is not and cannot easily be made available from the outset: For example, the inclusion of GPS in the manufacturing process is cost prohibitive, and despite this, sensor nodes may be deployed in adverse environments where GPS information cannot penetrate.

This begs the question, can position-based routing be applied to contexts where location information is not available to but a few nodes? This question is furthered by the notion that many proposed applications do not require any knowledge of geography, but do require advanced point-to-point routing services.

Applications such as data querying [36, 40, 81], reactive-tasking where events are triggered by local events, and data-centric storage [102, 111], all demand robust routing yet ignore physical location. Network applications such as these will identify a node by its identifier or by the data it stores, each of provides no means by which to route using position. In such contexts a network may implement position-based routing by use of a distributed hash table (DHT), which partitions keys/identifiers from the owner of the key ([6]). In a DHT sensor network, position-based routing is an appealing means to bridge this partition.

In this section we explore the potential for position-based routing where location information is generally unavailable. The solution that has received the widest attention is to develop protocols which construct their own maps and create their own coordinate systems. Such protocols are often said to construct and rely upon *virtual coordinate systems*. The routing in virtual coordinate systems is often coupled with the construction of the coordinate system, a coupling which necessitates an understanding of the coordinate construction itself.

Among the first feasible algorithms to construct a virtual coordinate system for the purpose of routing in wireless systems appears in [110]. This method requires no infrastructure. However, its centralised nature and expensive cost ($O(n^3)$) render it poorly suited for sensor networks.

The establishment of a coordinate system in a distributed fashion, in order to route messages between nodes, is non-trivial. We introduce two methods under investigation. The first is the distance-based approach where the aim is to create some semblance of a Cartesian space (without which direction between nodes is unclear). The alternative is a hop-based protocol which addresses using the distance between

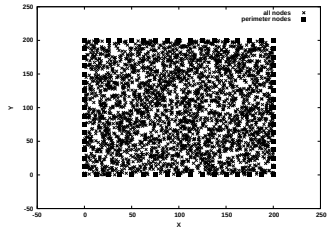
nodes, in hops. Each of these share drawbacks, a discussion we reserve until later.

Distance Based Coordinates and Routing

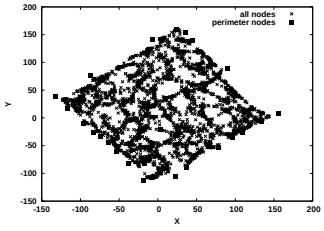
Among the earliest protocols promoting both a virtual coordinate system construct with a corresponding routing mechanism are the GEM [93] and Medial Axis [11] infrastructures. In GEM a labelled graph is constructed and embedded into the network topology in a distributed fashion, where the label encodes a node's position within the graph. The idea is demonstrated by coupled coordinate setup and routing protocols called VPCS (Virtual Polar Coordinate Space) and VPCR (Virtual Polar Coordinate Routing), respectively. Using VCPS, the GEM system is able to embed a ringed tree into the network in order to create a polar coordinate system. The VCPR algorithm designed to route over the polar coordinates is the first to guarantee delivery in a point-to-point fashion with no a priori geographic information. While the initialisation scales well, the recovery process initiated on node failure or link degradation is expensive. Recovery may force a large number of nodes to participate in a re-computation process. Similar work appears in [76].

To better demonstrate the ideas behind distance-based constructions, we focus our discussion on the NoGeo method proposed in [101]. In it the authors provide a mechanism to construct a virtual Cartesian space, and implement the simplest form of greedy routing. Their work is a derivation from previous work intended to test graph connectivity [83], and assumes that nodes may accurately measure the distances to their immediate neighbours.

Virtual Coordinate Construction For clarity, explanation of the NoGeo algorithm is presented in a cumulative fashion where at each additional step we remove some knowledge from the system. Hence we begin by describing the coordinate construction under the assumption that nodes on the network boundary, or perimeter, are aware of their position relative to the network as well as their exact location. The

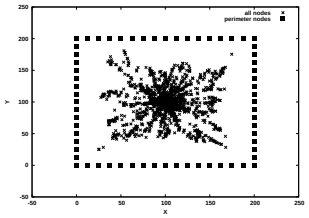


(a) Physical network in 200x200 space.

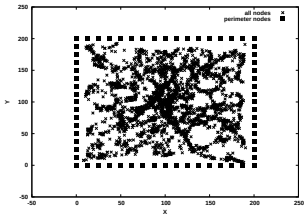


(b) Virtual network where perimeter nodes are initially unknown.

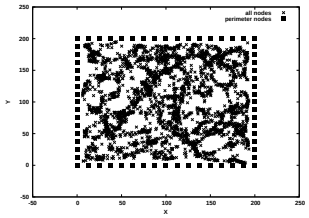
Figure 2.12: The (a) real and (b) virtual coordinates of a 3200 node network.



(a) After 10 iterations



(b) After 100 iterations



(c) After 1000 iterations

Figure 2.13: Intermittent virtual coordinates of 3200 nodes in a 200x200 space where nodes are known in advance to lie on the network's perimeter.

algorithm in [101] is based on an iterative relaxation procedure from [83] which is used to determine the locations of all remaining nodes in the network. The procedure is such that each link is represented by a force that pulls its adjoining neighbours together. The force in each of the x, y directions is proportional to the difference in the x, y coordinates. At any iteration a node's neighbours are held with fixed position; its *equilibrium* position, where the sum of the forces acting on it is zero, is the average of all its neighbours' $x-$coordinates in the $x-$direction, and similarly for the $y-$direction. This relationship motivates the iterative procedure repeated periodically at each node i using the relaxation equations

$$x_i = \frac{\sum_{k \in neighbour\_set} x_k}{size\_of(neighbour\_set(i))} \tag{2.3.1}$$

$$y_i = \frac{\sum_{k \in neighbour\_set} y_k}{size\_of(neighbour\_set(i))}. \tag{2.3.2}$$

in each of the x and y directions, respectively.

As an example, consider the wireless sensor network in Figure 2.12a. This network consists of 3200 nodes within a 200x200 unit space where each node has a communication range of 8 units. As the iterative procedure described above is executed over this network, sensor coordinates will gradually shift to match their true coordinates within the network. This 'shaping' is depicted following 10, 100, and 1000 iterations in Figures 2.13a- 2.13c, respectively. In this example he initial coordinates of each node have been set to the center of the network at (100,100). (We later return to compare the virtual and actual topologies.)

This coordinate construction may be prefaced with additional steps if there is no advance knowledge of perimeter nodes, provided there are two beacons somewhere in the network. (Beacons are distinguished from the remaining network because they either hold and disseminate information, or play some specific coordination role required for successful setup and communications.) Either beacon may be used to

Algorithm 2.3 Coordinate determination among perimeter nodes in NoGeo.

1. Each perimeter node broadcasts a HELLO message so that each may calculate its distance to every other perimeter node. The result is stored in a *perimeter vector*.

2. Each perimeter node broadcasts its perimeter vector to the network so that each perimeter node knows the distance between every pair of perimeter nodes in the network.

3. Each perimeter node triangulates to find the coordinates of every perimeter node in the network. Coordinates are chosen so as to minimise

$$\sum_{i,j \in perimeter} (measured\_dist(i,j) - euclid\_dist(i,j))^2, \tag{2.3.3}$$

where $measured\_dist(i,j)$ represents the distance between nodes i, j as measured in 1, and $dist(i,j)$ is the Euclidean distance between the virtual coordinates of i and j.

identify nodes on the perimeter, after which point the perimeter nodes exchange messages to determine their locations.

The first step in accomplishing this task is to identify nodes on the perimeter. Recall the assumption that two beacon nodes exist. Either beacon is designated as the primary beacon, that broadcasts a HELLO message. Each node uses receipt of the HELLO message to determine its location from the beacon, according to the *perimeter node criterion*. The *Perimeter Node Criterion* says that if a node is farthest away from from the bootstrap beacon among all nodes in its two-hop neighbourhood, then this node decides it lays on the perimeter of the network. This is by no means an exact determination but simulations show it identifies a sufficient number of perimeter nodes. Once nodes have identified their perimeter status they must coordinate to exchange information and calculate their coordinates according to Algorithm 2.3.

Each perimeter node, following Step 3 in Algorithm 2.3, has established its own virtual coordinate. One challenge remains. Each perimeter node has established its

own virtual coordinate yet the network lacks a global orientation. The reason is that any set of coordinates satisfying Equation 2.3.3 may be rotated, translated, or transposed. The result is that perimeter nodes cannot be guaranteed to share the same sense of direction.

The solution is the reason two beacons must exist. The two beacons share their perimeter vectors and participate in the triangulation process. During this process each of the perimeter nodes calculates the center of gravity of the network. The combination of the two beacons with the center of gravity provide a set of axes on which all nodes can agree: the center of gravity becomes the origin, while each of the two beacons define the positive x, y axes.

An example topology at equilibrium, following the execution of perimeter identification, coordinate determination and dissemination, appears in Figure 2.12b. This is the virtual network space that corresponds to the actual network in Figure 2.12a. Using this scheme the virtual coordinates maintain a similar structure to the actual coordinates, but with some caveats. For example, the network appears to have been rotated about the virtual center of gravity. However, the rotation is uniform and consistent, meaning that the rotation is network-wide. Note that holes in the virtual networks constructed in Figures 2.12b and 2.13c are much larger in size than in the actual network of Figure 2.12a. As discussed below, this salient feature does not reflect itself in the routing mechanism.

Routing in the virtual Cartesian coordinate system takes the form of pure greedy routing as described in Chapter 2.3.1: Messages are forwarded to the neighbour that most reduces the distance to destination until no such neighbour exists or the destination is found. Despite the inaccuracies of location determination and the increase in hole sizes (as reflected in physical space), greedy routing over virtual coordinate systems performs better than it does over physical coordinate systems. The rate of successful delivery rate is higher when routing over virtual coordinate systems.

The underlying reason for an increase in the rate of delivery is that locations are assigned coordinates relative to the locations of other nodes in the network, not relative to the space that the network occupies. Hence, a virtual coordinate system accurately reflects not network geography, but rather network connectivity. This relationship, when reflected in actual space, is responsible for the apparent growth in hole sizes. (Despite this fact physical coverage area is unaffected; a necessary property of sensor networks). It is also responsible for the improvement in the performance of greedy routing as it reflects network paths and connections more than node location in the physical space.

One drawback to approaches in this class of position-based routing is the expense incurred by the computation and transmission necessary to establish a reasonable coordinate system. Consider in NoGeo, for example, that it is somewhat impractical that initialisation requires $O(\sqrt{n})$ nodes to flood the network, and for at least this number of nodes to store $O(n)$ state (since the distance vectors occupy $O(\sqrt{n} \times \sqrt{n})$ space). The state requirement has since been reduced in [75] by implementing algorithms that elect a constant number of beacons, as well as establishing coordinates using a repulsion, rather than an attraction, mechanism. Still, many rounds of network-wide communications are necessary. We proceed with an approach that reduces on this expense in exchange for a loss in topological accuracy.

Hop-Based Coordinates and Routing

One alternative approach to constructing virtual coordinates forgoes Euclidean approximations, altogether. Instead designated coordinates are comprised of a vector that contains a set of hop-distances to beacons located around the sensor network [5, 12, 18, 28, 118, 126]. As is the case with NoGeo, the goal of these methods is to deliver packets in an environment with no a priori knowledge of node locations, in a point-to-point manner. We demonstrate this approach to position-based routing

using Beacon Vector Routing (BVR) in [28] and reserve discussion for the differences with other projects until later.

BVR is a protocol that assigns routing coordinates and defines a distance function used in forwarding decisions. A node's coordinate is a tuple recording the hop distance to each of a subset of available beacons, information which is disseminated using reverse path tree constructions. (A reverse path tree construction occurs when a beacon broadcasts its existence to the network and all remaining nodes record the shortest hop distance to that beacon.) A distance function is used to route greedily. In the event that greedy routing fails, a correction mechanism exists to guarantee delivery.

If we let r denote the total number of beacons, and q_i denote the distance in hops from a node to beacon i. The position of node q is the tuple $(q_1, q_2, \ldots, q_r)$. By this definition it is possible for multiple nodes to share the same coordinate so a node identifier is necessary to disambiguate between nodes with identical coordinates. The distance function must favour greedy forwarding to maintain a high level of efficiency. When evaluating the distance function the BVR metric aims to minimize the difference in coordinates component-wise. This metric is based on the idea that it is better to move towards a beacon close to the destination than it is to move away. Hence, the distance function δ is designed to move a message towards a beacon if the destination is closer to the beacon than the current node, ie. it also moves a message away from a beacon if the destination is further away. (Note that using this intuition, movement towards a beacon always reduces the distance to the destination but moving away is not: The destination may sit on the other side of the beacon from the current node.)

Let the distance function $\delta(p, d)$ measure the goodness of node p as a next hop to d. The above intuition is encapsulated into the distance function using the sums,

$$\delta_k^+(p, d) = \sum_{i \in C_k(d)} max(p_i - d_i, 0) \tag{2.3.4}$$

Algorithm 2.4 Overview of Beacon Vector Routing (BVR)

1. (Greedy.) Where possible, forward to the neighbour which minimises Equation 2.3.4, breaking ties using Equation 2.3.5.

2. (Recovery.) If no such neighbour exists, record the current distance in the packet as δ^{min} and forward to the beacon closest to the destination.

3. (Recovery.) If message has reached a beacon without reverting back to greedy mode, broadcast with a time-to-live equal to hop distance from the destination node.

$$\delta_k^-(p,d) = \sum_{i \in C_k(d)} max(d_i - p_i, 0). \tag{2.3.5}$$

where $C_k(d)$ is the set of k beacons closest to d. The metric is a sum of differences derived from Equations 2.3.4 and 2.3.5. δ_k^+ is the sum of the differences of beacons closer to the destination than the node p, while δ_k^- is the sum of the differences of beacons further away.

BVR routes greedily as follows. The next hop is chosen to be the node that minimises δ_k^+; any tie that may occur is broken by minimising δ_k^-. Note that the k beacons may number fewer than the total number of beacons in the system, and that the smallest difference δ^{min} encountered during a traversal must be stored in the message header for reference.

A global view of the main BVR algorithm is summarised in Algorithm 2.4. As is the case with other greedy schemes, there are occasions where forwarding may terminate prematurely, failing to find a neighbour that improves on δ^{min}. BVR is able to guarantee delivery using a two-tier recovery mode. First, if a node has no neighbour closer to the destination than itself, it will forward the message to the beacon closest to the destination. The idea is that if a sensor node is unable to find the destination then it should send the message in the direction of a node that can. Interim nodes that receives the message will forward to the destination as if recovery

never occurred, if possible. Second, if a beacon is unable to further minimize δ^{min} then it initiates a scope flooding to find the destination. (The scope of the flood is known since the destination coordinates record the hop-distance from each beacon.) While this recovery mechanism is an expensive means of guaranteeing delivery, it is found to occur infrequently in simulations.

Similar ideas have been proposed in [12, 18]. FT-BVR in [18], for example, extends BVR using fault-tolerant techniques to improve on a variety of performance metrics. Logical Coordinate Routing (LCR), in [12], is similar to BVR in its coordinate construction and routing mechanisms. Where it differs is in recovery. Where BVR ultimately resorts to a scoped broadcast, LCR backtracks along the path-thus-far until an alternate path is found or the message returns to the originating node where delivery is deemed to have failed. Clearly, LCR avoids the expense of a broadcast in exchange for additional state (either in the message or stored at interim nodes).

More recent work has emphasized a hybrid approach to virtual coordinate constructions. The goal of the S4 [88] project is to reduce both state and path stretch in virtual coordinate systems. S4 consists of a hierarchical system with physical coordinates at the lower level and virtual coordinates at the higher level. At the higher level, packets are directed towards clusterheads near the destination. If within the same cluster, nodes will route to the destination using the shortest path as determined by physical coordinates. The axis-coordinate routing protocol [117] operates similarly. Packets are routed using hierarchical 5-tuple coordinates developed along latitudinal and longitudinal axes. Both S4 and axis coordinates require an multi-round preprocessing step that is network-wide.

Drawbacks and Challenges with Unknown Coordinates

Generally speaking, mechanisms that construct and route over virtual coordinate systems manage to overcome many of the challenges that face routing in physical

coordinate systems. To wit, virtual coordinate routing schemes make no assumptions pertaining to the unit disc graph, an assumption which is shown to be violated in practice [17, 121, 125], nor do they require GPS-like services that may be unavailable. This benefit does come at some expense, however.

The first limitation is related to the infrastructure necessary for correct operation. Constructs that create coordinate systems which reflect connectivity require the use of beacons. (Beacons are defined as nodes that have knowledge of their location via either global positioning systems, that have some pre-programming, or that are placed strategically). Each of these incurs a cost increase on the manufacturing and deployment process that is intended to otherwise be cheap. In addition, many environments well-served by sensor-network applications can be volatile. In such environments a small number of beacons may be lost, destroyed, or otherwise disabled. The use of beacons may be restrictively expensive and complicated when deployed broadly, yet present fault-tolerance and failure issues when deployed narrowly. If beacons are required, does it suffice to select them randomly? If not, what determines the goodness of a beacon? Despite these challenges, beaconing may be an effective solution to a difficult problem.

In addition, virtual coordinate and routing methods also incur additional complexity in communication, and sometimes computation. Coordinates in the network cannot be learned unless information is broadcast so that all nodes share similar knowledge. Energy consumption is a greater concern since transmission is known to be a high-energy operation, and broadcasts increase the chance of collision (though there are proposals to *intelligently* broadcast such as in [42]). Often the determination of coordinates requires additional computation. Work to appear in [85] may reduce this complexity.

Finally, we address the accuracy of virtual coordinates. Coordinates that record hop counts are likely to be duplicated throughout the network, and so additional care must be taken to deliver messages to the intended recipient. Coordinate systems that

measure physical distances reflect network connectivity well, but are subject to limits in resolution. Work on Nagpal's algorithm in [91], a set of algorithms to construct and improve coordinates based on distances from three distinct beacons, reveals the smallest resolution to be $\frac{\pi}{4n}r$, where n is the average neighbourhood size and r is the radius. If achievable, this limit may or may not be acceptable in practice.

2.3.3 Recent Improvements

Many recent efforts investigate the merits of hybrid routing approaches, implementing position-based routing over a hierarchical environment. The face-tracing project [123], for example, embeds the network graph into a higher dimension topological space. Using probe packets that identify faces, each node orders its incident links in a such a way that permits successful face-routing in this higher-dimensional space. Optimisations include clustering and link omissions, where appropriate. The face-tracing protocol makes no unit-disc assumption despite operating on physical coordinates. However, the cluster establishment and probing phases result in expensive preprocessing and maintenance requirements. An alternative to hole recovery is hole avoidance, as suggested by [49]; here the authors have shown that the network may discover 'unsafe' regions where routes can be re-directed to avoid the hole region entirely.

The authors in [33] improve upon the hierarchical 'landmark' approach in [24]. It remains an open problem how to select appropriate landmarks. Still, despite the unit-disc assumption their work allows for uncertainty in node locations. An additional embedding appears in [61], where each node in the network graph is assigned a coordinate in hyberbolic space. The authors show that greedy routing will always find a route to the destination in this space. The cost of network-wide construction and look-ups, however, may render this approach prohibitively expensive. Additional hybrids appear in [13, 29, 38, 82] and [90].

A number of studies have focused on augmenting a single performance metric

	MFR, NFP, Compass	Greedy	GRA, DFS-QoS	Mapping	Planar Subgraph Methods	CLDP	GDSTR	NoGeo	BVR
UDG Assumption	No	Yes	No	Yes	Yes	No	No	No	No
Loop-free	No	Yes	n/a	n/a	Yes†	n/a	Yes†	Yes	Yes
Critical Nodes	No	No	No	No	No	No	Yes	Yes	No
Beacon Requirements	No	No	No	No	No	No	No	Yes	Yes
Multiple Routes	No	No	No	Yes*	No	n/a	Yes*	No	No
Startup Coordination	Local	Local	Local	Extended	Local	Extended	Global	Global	Global
Void Recovery Cost	n/a	n/a	Medium	Medium	Low	High	Medium	n/a	Medium
Guaranteed Delivery	No	No	Yes	Yes	Yes	n/a	Yes	No	Yes
Obstacle-resilient	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes
Coords Reflect Connectivity	No	No	No	No	No	No	No	Yes	Yes

† - A loop only occurs when a packet returns to its origin, indicating the destination is unreachable.
∗ - The choice of route occurs only when recovering from a void region.

Table 2.1: Summary of various protocol attributes.

in existing schemes. In [87], for example, nodes exploit the fact that nodes listen to transmissions even if the transmissions are intended for other parties. In doing so a node is able to prune the path taken during the recovery phase in face-routing.

In addition, face-routing algorithms suffer from high congestion along the recovery paths, reducing the throughput and lifetime of the network. Load-balancing provides one solution though current projects are known to converge too slowly. A recent study suggests that an element of randomisation can provably reduce congestion, and so increase throughput [114].

There are also projects that augment position-based routing using outside resources. For example, routing in [127] exploits knowledge of city maps, while [35] adds explicit mac-layer support for geo-routing. Further efforts have proposed tailoring geo-routing to deliver video signals [16] or optimise metrics other than path length [73]

2.4 Summary

In this chapter we have explored algorithms and mechanisms for boundary detection and position-based routing in sensor networks, whose communication graphs are large

and dense relative to traditional wireless ad-hoc networks. Position-based routing occurs where forwarding decisions are based on a metric that reflects the position of nodes in reference to the physical space, or to reference nodes. A selection of key attributes for a selection of protocols appears in Table 2.1.

We proceed in the next chapter with the local convex view, a method designed to identify nodes along the network edges.

Chapter 3

Localised Convex Hulls for Boundary Detection

In this chapter we introduce the local convex view (*lcv*) as a heuristic approach to identify nodes close to the network edge. The local convex view is defined as the convex hull of the nodes within range. (Thus, as many local convex views exist as nodes in the network.) In the simplest terms a node decides it is close to the network boundary by asking the following question: Am I in my own local convex view? It is inspired by the knowledge that the convex hull of a set of points consists of the outermost points in the set. Furthermore, much like statistical methods as described in Chapter 2, it is motivated by the hypothesis that within view of many nodes there exists structural information relevant to the network.

For those contexts where position information is unavailable *lcv* assigns local coordinates using its 1-hop distance measurements and the 1-hop measurements of its neighbours. Each node constructs a Cartesian space by placing itself at the origin and its furthest neighbour along the horizontal axis. All remaining nodes in the neighbourhood are assigned coordinates relative to established coordinates. The success of this triangulation approach relies on the assumption that a node's neighbourhood is

2-connected, ie. removing the node in question leaves no disconnected components. Where this assumption fails, we show the maximum number of possible disconnected components is 5. Furthermore, when a neighbourhood is not so connected, we suggest a simple probabilistic model to determine boundary status according to the criteria set in *lcv*.

In our evaluation we simulate networks of varying size and topology. Topologies vary by generating networks where node locations are selected from uniform, normal, and skewed (Pareto) distributions. In our investigation we have identified two metrics for comparison. The *edge proximity* measures the likelihood of boundary-node declarations that occur relative to the network edge. The *regional proportionality* measures the proportion of boundary- versus regular-nodes relative to the network edge.

We first compare *lcv* against the 2-hop method from [101] and an adaptation of the tent-rule in [23], whose localised operations and limited messenging complexity make them suitable for comparison. Results indicate that the tent-rule is unsuitable for network edge detection. The underlying cause is in its design which is to identify nodes that abut *any* unreachable region, including those that are 'inside' the network. The 2-hop method, which gathers some global information, generally returns the seemingly most accurate results. While perhaps unsurprising, this observation is misleading. Further investigation shows that the 2-hop method may reveal clusters of boundary nodes as the network grows dense. This leaves some regions of the network edge over-represented while others are left under-represented. We find this is an artifact of the way in which the 2-hop method gathers information. By contrast, *lcv* reveals a reasonable set of boundary nodes. Unlike the 2-hop and tent-rule methods, *lcv* is fairly resilient to changes in topology.

Our evaluation is further propelled by the growing number of studies that suggest error in position estimation is currently unavoidable [96, 105, 120]. We insert position error to the system by blurring neighbour positions stored at each node. In doing so,

a pair of neighbouring nodes is unlikely to agree on the relative position of any third node in common view.

The results are surprising and counter-intuitive: The difference in accuracy between the *lcv* in a perfect environment and one with position estimation error are statistically insignificant. Motivated by this observation we enumerate a complete base set of node configurations that may be seen by *lcv*. Our analysis reveals that *lcv* is immune to two of these configurations. Further simulations show the frequency of false-positives and false-negatives imposed by a third, ambiguous, configuration is low. We conclude that the geometric properties underlying *lcv* are responsible for its resilience to error.

In summary, in this chapter we develop the first of two boundary node detection methods. It uses only nodes in range and resolves any needed information that is missing. We study our method alongside two methods with similar properties, and in environments where position estimation is erroneous. Our simulations and analysis reveal that *lcv* is resilient to the impediments faced by competing methods, as well as errors in position estimation.

3.1 Local Convex View

In this section we describe the local convex view (*lcv*) algorithm, an autonomous method for selecting a subset of nodes to describe the network edge. Our algorithm makes only the assumptions that generated or assigned node IDs within each neighbourhood are unique, and that distance measurements are available. We consider a deployment of a large wireless network where, initially, nodes lack any knowledge of their positions.

3.1.1 Local Boundary Node Identification

A set of nodes far apart may be obtained by finding the convex hull [1] of a set of nodes. There are potential drawbacks to choosing the convex hull of the network: Too few boundary nodes may be chosen, the shape of irregular networks may fail to appear and, finally, we know of no methods by which convex hull computations may be localised.

By contrast we can localise the convex hull computation to capture the 'shape' of a sensor network. The idea is illustrated in Figure 3.1 where we compute the local convex view for each node. A node's local convex view consists of the set of nodes that comprise the convex hull of the neighbourhood in view. (Thus there are as many local convex views in a network as there are network nodes.) Observe in Figure 3.1 on the right that by taking the set of "outside" nodes from the set of local convex views, the shape of the network structure begins to emerge. We use this observation to motivate our method for boundary node identification.

In the simplest terms a node decides if it lies on the network boundary by asking the following question: Am I in my own local convex view? With the return of a positive answer, a node declares itself to be a boundary node. Referring again to Figure 3.1 we focus on the nodes labelled u, v, w. Using the local convex view criteria nodes v and w declare themselves to be boundary nodes while node u sits idle. We evaluate the accuracy of this criteria in Chapter 3.2 and proceed with a discussion of the algorithm and its merits.

The complete procedure appears in Algorithm 3.1. For each node u the boundary node declaration process consists of three steps. Steps 1 and 2 remedy the initial lack of position awareness. Recall that network nodes may lack any knowledge of their positions, information that is necessary to compute a convex hull. We obtain this information by constructing a local coordinate system as this is sufficient to compute

[1] In two-dimensions, a set S of points is defined as *convex* if for every $x \in S$ and $y \in S$, the segment $xy \subseteq S$ [97].

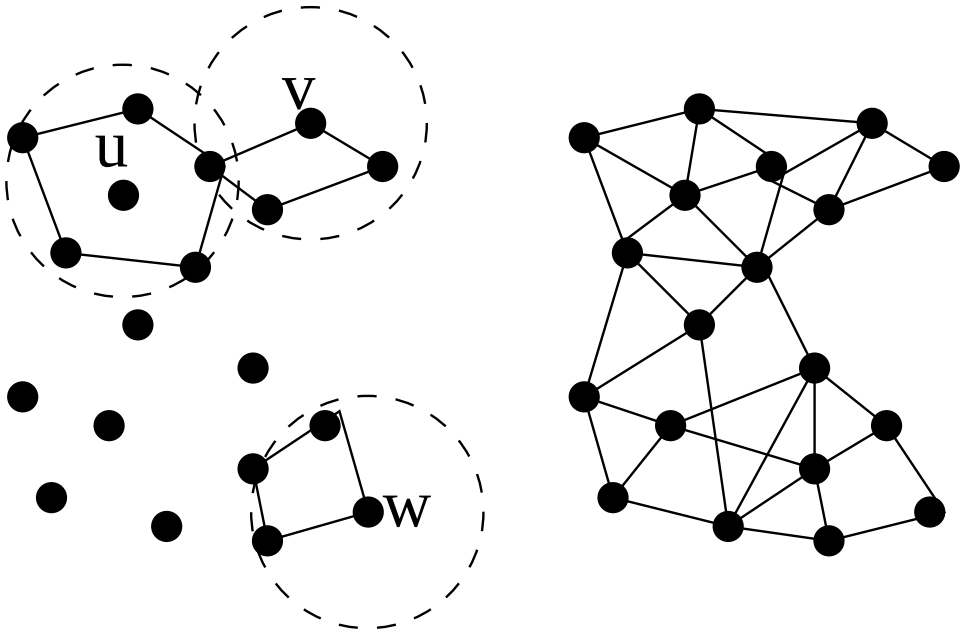


Figure 3.1: (Left) Nodes u, v, w compute their local convex hull; only v and w declare boundary status. (Right) The shape of the network emerges from the collection of local convex hulls.

Algorithm 3.1 Boundary Node Identification Algorithm at any node u.

1: Share the distance measurements to single hop neighbours.

2: Set u as the origin of, and construct local coordinate system.

3: Compute the local convex view (lcv).

4: **return** $u \in lcv$

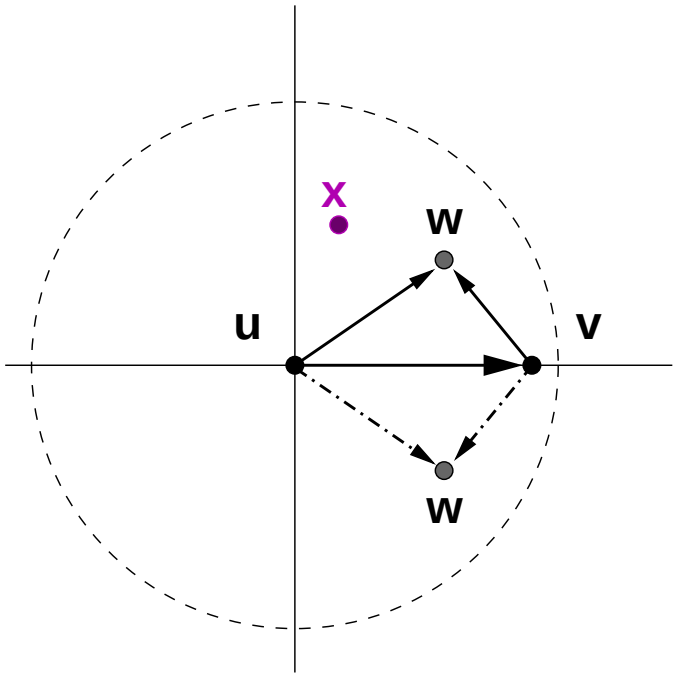


Figure 3.2: Neighbourhood coordinate establishment.

the local convex hull.

Each node in the network begins by sharing its 1-hop distance measurements with its neighbours. Once obtained, each node constructs a local coordinate system by placing itself at the origin and, in a depth-first manner, places remaining neighbours relative to those whose local coordinates have been established. Two potential options are lateration and angulation [53]. We demonstrate this idea from the perspective of node u in Figure 3.2. Node u places itself at the origin of a Cartesian space and sits neighbour v on the horizontal axis. (For our purposes we use the furthest neighbour.) The next node, w, may be placed on either side of the horizontal axis since convex hull computations are unaffected by rotations and translations. Remaining nodes are placed similarly and with respect to established coordinates.

In the final step a node computes the convex hull over the neighbourhood in view. If it sits within its local convex view (ie. in the set of points describing the

convex hull of the local neighbourhood) then it declares itself to be a boundary node. We discuss the correctness of this approach in the next section.

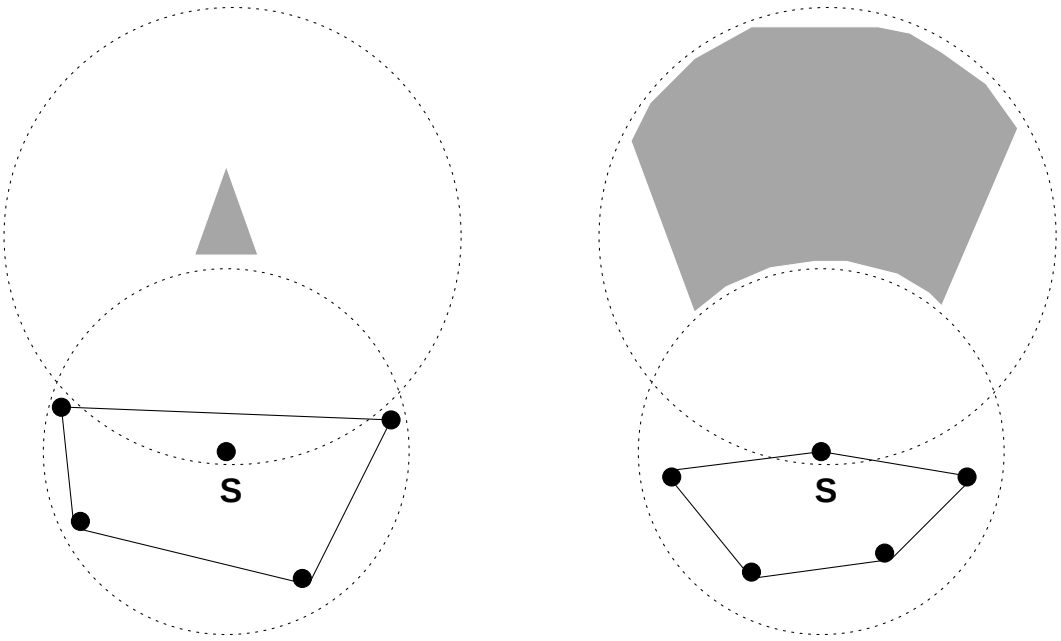
3.1.2 Correctness of Local Convex View

We note that it is impossible for a node to declare boundary status unless it sits on a network boundary. This notion is demonstrated in Figure 3.3 which depicts the two general cases faced by the *lcv* algorithm. We can see that node s may border an unreachable region according to two definitions: Figure 3.3a demonstrates the case in which we are uninterested since s has neighbours that lay qualitatively closer to the unreachable region (neighbours of s sit closer to the greyed region than the tangent at s); by contrast Figure 3.3b demonstrates that, within view, s is closest to the unreachable region. This implies that false positives may be returned (eg. routing 'holes' inside the network), and that false negatives are impossible.

The cost of the algorithm is one transmission to share distance vectors, and $O(d \log d)$ computation where d is the number of nodes in the neighbourhood. The local convex view avoids broadcasts and inter-node cooperation. It is both protocol and architecture independent.

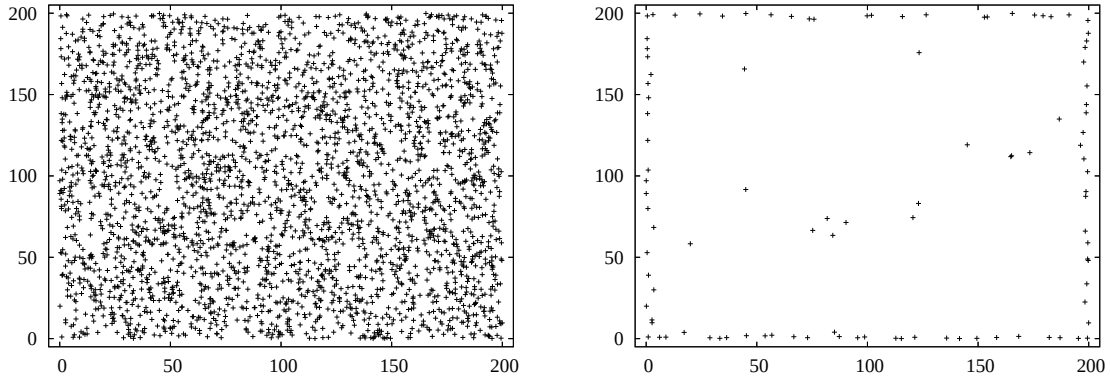
To better illustrate the outcome of our method we use the three networks depicted in Figure 3.4. Each network consists of 3000 nodes in a 200x200 space each with a range of 8 units. Figure 3.4a represents a network where node locations are distributed according to a uniformly random distribution; in Figure 3.4b node locations are distributed according to a normal distribution; in Figure 3.4c node locations are distributed according to a skewed (Pareto) distribution. Within each figure we plot the complete set of network nodes on the left and the subset of nodes that declare boundary status on the right. Comparing the right and left plots we see the majority of nodes that declare beacon status lie in the outer regions of the network.

Our continued evaluation appears in Chapter 3.2. Next we construct a simple

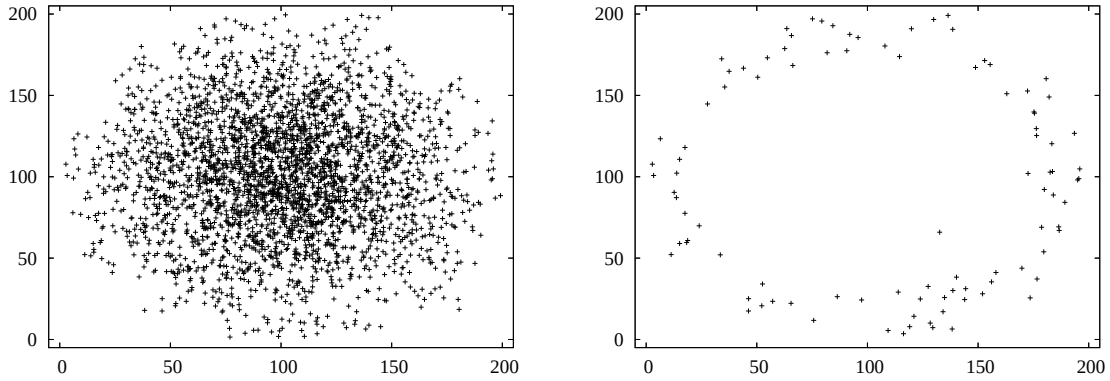


(a) The 'concave' case. (b) The convex case.

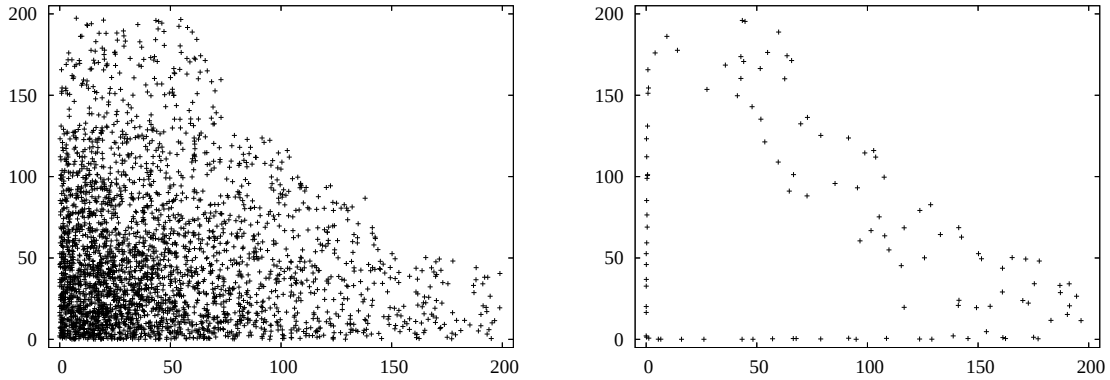
Figure 3.3: A node may abut unreachable regions in two ways. The *lcv* detects regions that are outwardly unreachable.



(a) A uniform network.



(b) A normal network.



(c) A skewed (Pareto) network.

Figure 3.4: Example networks of 3000 nodes with varying topologies on the left. The corresponding *lcv* nodes for each network on the right.

probabilistic model to deal with missing information during coordinate assignment.

3.1.3 Dealing with Incomplete Information

We previously described the way each node assigns coordinates to neighbours. There remains in this approach a caveat. Say node u constructs its neighbourhood coordinate system according to Algorithm 3.1. Coordinate assignments may only succeed if u's neighbourhood is 2-connected [2] . We can also say the removal of u must leave a single connected component. Without 2-connectedness u will be unable to assign a coordinate to at least one neighbour. In this section we suggest a probabilistic solution.

Consider two contiguous neighbours around u sorted in angular order. By normalising the communication range to 1 and solving the associated cosine rule, $1^2 = 1^2 + 1^2 - \cos A$, we determine the maximum possible angle between two communicating neighbours to be $\pi/3$. Using this result we can show that there exists at most 5 disconnected components (ie. non-communicating neighbours) around u. Furthermore, a node cannot lie on its local convex view if it has greater than 3 such components.

In the example shown in Figure 3.5 node u has assigned coordinates to three neighbours in a connected component. Wishing to assign a coordinate to node v, node u sees that v communicates with no other neighbour of u and so must lie somewhere along the dotted arc. If v lies in the ranges described by angle β then u must sit on its local convex view. Hence, u sits on its *lcv* with probability

$$\mathrm{p}[u \in lcv] = \frac{2\beta}{2\beta + \alpha}. \tag{3.1.1}$$

The range of angles of α is equivalent to the range of angles covered by the neighbours

[2] A 2-connected network is one in which there are 2 disjoint paths between every pair of nodes.

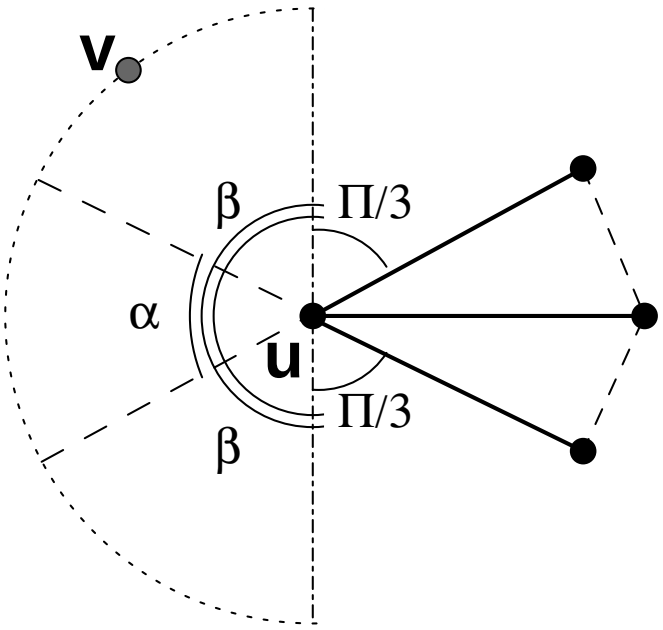


Figure 3.5: During coordinate assignment u finds v is disconnected from the remaining neighbourhood. Node u lies on the lcv only if neighbour v sits in the range denoted by β.

with assigned coordinates. So β may be written as

$$\beta = \pi - (\alpha + \frac{\pi}{3}). \tag{3.1.2}$$

Note that the probability $\mathrm{p}[u \in lcv] = 0$ when $\alpha \geq 2\pi/3$. Then, by substituting Equation 3.1.2 into 3.1.1, node u declares it is on its *lcv* as follows,

$$\mathrm{p}[u \in lcv] = \begin{cases} \frac{\frac{4\pi}{3}-2\alpha}{\frac{4\pi}{3}-\alpha}, \text{if } \alpha < \frac{2\pi}{3} \\ 0, \text{ otherwise.} \end{cases} \tag{3.1.3}$$

We have demonstrated a probabilistic solution for the case where there are two disconnected components. Next we evaluate *lcv* against comparatively similar methods.

3.2 Performance Evaluation

In this section we evaluate *lcv* with the tent-rule and the 2-hop methods, described in Chapter 2 and quickly reviewed below. Our comparison is followed with an evaluation of *lcv* when position estimation error is inserted into the system.

3.2.1 Comparative Methods

The 2-hop method [101]. Stemming from a need to identify a set of furthest beacons, the authors inject an anchor into the network. Each node in the network records and transmits its hop distance to the anchor. Any node that finds itself the furthest of all its 2-hop neighbours from the anchor becomes a beacon. On account of the focus of this work being not the beacon selection but an ensuing coordinate construction, no evaluation of the quality of selection was provided.

Tent-rule [23]. The tent-rule, so named for its appearance when represented diagrammatically, identifies all unreachable regions as defined by greedy routing. It works by sorting neighbours angularly about a node. If the bisectors of edges to

contiguous neighbours intersect outside of range of the node, then the node abuts an unreachable region. Given that the network boundary itself delineates an unreachable region, the adaptation of the tent-rule for our investigation is appropriate.

3.2.2 Experimental Design

To evaluate and compare the algorithms, we simulate networks of varying density and distribution. Network nodes are distributed in a 200x200 unit space, each node with a fixed range of 8 units. We vary node density by changing the network size. Note that by changing size instead of communication range we can vary neighbourhood density without affecting the maximum network diameter. Network sizes are 1500, 2500, and 3500 nodes. (In the uniform networks this results in average neighbourhood sizes of 7, 12, and 17 nodes.) To obtain more accurate results we tabulate and experiment over the largest connected component of each network. Experiments repeat over 25 runs of each network generated using non-overlapping streams.

Nodes locations are chosen from a normal or skewed (Pareto) distribution in addition to the uniform distribution traditionally used to generate wireless network topologies. Uniformly distributed networks may be sufficient to provide insight yet are poor representations of many real deployments. Normal coordinates are generated with an average of 100 (the center) and a standard deviation of 40. Skewed coordinates are chosen from the Pareto distribution with scale parameter 1.0 and shape parameter 100.5. Examples topologies appear in Figure 3.4.

The choice of appropriate metrics is not obvious. We have isolated two metrics we believe to be suitable for this study. Our measures of success center on locations of nodes that declare boundary status relative to the network boundary. For this reason networks are separated into partitions appropriate for each network type, described in Chapter 3.2.4. Measures of success are represented as follows:

- We measure the *edge proximity* as the probability that the location X of a

declared boundary node lies in region x. This measures the likelihood that nodes sitting on their local convex view are good beacon candidates (ie. sitting close to the network boundary).

- We measure the *regional proportionality* as the percentage of nodes within a region that declare boundary status. This ratio should be highest towards the true network boundary. It is designed to reinforce those methods that are more likely to find nodes close to the edge of the network.

Our simulations implement the algorithm described in Chapter 3.1 with one addition. Nodes that observe a local convex view consisting of three or fewer nodes return negative boundary status. Our experiments show that this reduces the number of reporting nodes in dense networks to a manageable level without compromising results in sparse networks.

3.2.3 Performance in a Perfect Environment

Our evaluation begins with a direct comparison between the performance of *lcv*, as well as the tent-rule and the 2-hop methods described above. These methods' similarities to *lcv* in properties makes them appropriate for comparison.

Boundary Node Set Size

One of our goals is to identify a reasonable set of boundary nodes: too many nodes create ambiguity, while too few risk sacrificing resolution. We report the number of actual reporting nodes as the boundary node set size. The boundary node set size is largely a subjective measure we use to gain insight into measures that are later used.

We expect and confirm uniformly generated networks to be the least-well performing of the three networks we study. Tables 3.1, 3.2, and 3.3 list the average size of the largest connected component (lcc), and the average number of nodes that declare

Table 3.1: No. of boundary nodes returned in Uniform networks with 99% confidence intervals.

Network Size (Neighbourhood)	Method			Largest Conn. Comp.
	lcv	tent rule	2-hop	
1500 (7)	264.4 ± 8.9	466.7 ± 8.1	130.3 ± 9.0	1490 ± 7.5
2500 (12)	140.9 ± 5.1	299.3 ± 6.0	103.6 ± 5.6	2499.8 ± 0.4
3500 (17)	100.6 ± 3.4	181.6 ± 5.6	101.4 ± 9.8	3499.9 ± 0.2

Table 3.2: No. of boundary nodes returned in Normal networks with 99% confidence intervals.

Network Size	Method			Largest Conn. Comp.
	lcv	tent rule	2-hop	
1500	83.9 ± 5.8	161.0 ± 6.3	63.2 ± 4.4	1406.7 ± 7.7
2500	87.3 ± 4.9	160.1 ± 4.9	68.0 ± 3.8	2433.8 ± 4.9
3500	88.0 ± 4.6	155.4 ± 5.5	69.6 ± 3.9	3450.8 ± 5.3

Table 3.3: No. of boundary nodes returned in Skewed networks with 99% confidence intervals.

Network Size	Method			Largest Conn. Comp.
	lcv	tent rule	2-hop	
1500	103.8 ± 7.1	168.8 ± 9.3	78.3 ± 10.2	1359.0 ± 12.9
2500	116.0 ± 5.6	216.7 ± 11.1	114.3 ± 15.4	2382.2 ± 10.9
3500	125.4 ± 6.0	244.1 ± 9.3	135.6 ± 17.4	3403.8 ± 12.3

boundary status for each network, for all methods. All values appear with their 99% confidence intervals.

The 2-hop rule consistently returns the smallest number of nodes that declare boundary status. We believe the underlying causes are that i) the 2-hop method compiles global information and that ii) it maintains a record of its 2-hop neighbourhood, allowing decisions that are better informed. (We will discover this observation to be untrue in later sections.) Despite the increased knowledge of the 2-hop method, *lcv* remains competitive in all but the sparsest of uniform networks. Across all networks the tent-rule returns 2-3 times greater a number of nodes than the best performing method. The underlying cause is that the tent-rule is designed to report all routing holes, including holes that are 'internal' to the network.

Comparing the values from Tables 3.1, 3.2, and 3.3, the number of boundary-status nodes appear to be most greatly affected by density in uniform networks. For example, in uniform networks the ratio of boundary-to-regular nodes reported by *lcv* varies from 1 in 6 in the sparsest networks to 1 in 30 among dense networks. While there is some variation in non-uniform network numbers, the effects associated with increases in network size are much less pronounced.

Edge Proximity

The boundary node set size hides the locations of the nodes that report boundary status. In this section we measure the proximity of reporting nodes to the edge of the network. The edge proximity, depicted in Figures 3.6,3.7, and 3.8 is tabulated as the cumulative distribution over partitions of the network. Each type of network is partitioned in a fashion that is appropriate for its overall shape, according to the following criteria.

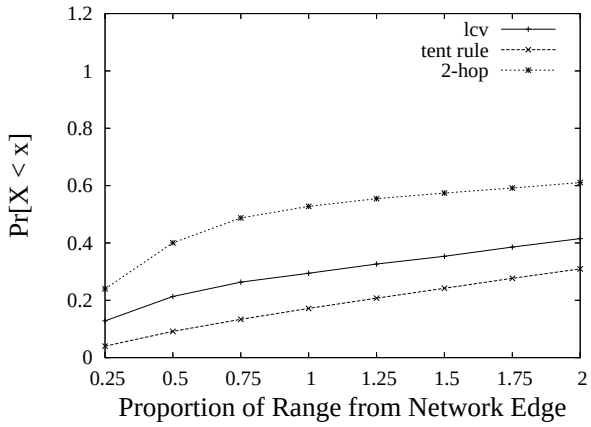
Uniform networks are partitioned into quadrilateral 'rings'. Each ring is of width equivalent to $0.25R$, where R is the communication range.

Normal networks are partitioned into rings that are 0.25 standard deviations in width. The statement "80% of reporting nodes sit outside 2σ" may be interpreted as 80% of reporting nodes sit amongst the outermost 5% of network nodes.

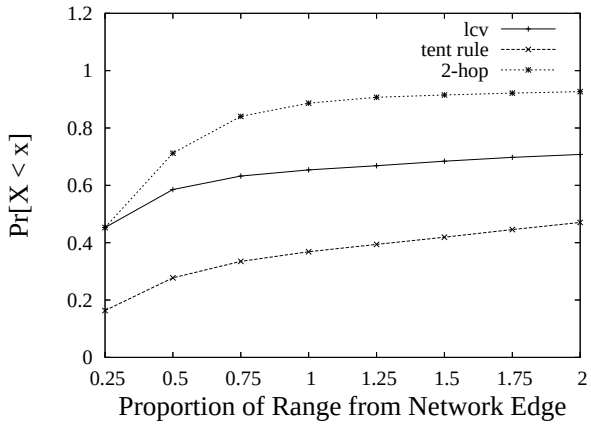
Skewed networks are partitioned into diagonals that span from the y to the $x-$ axis, 10 units apart. So, for example, nodes that report in the 180 region have x and y coordinates with a sum greater than 180.

The accuracy of all three methods generally follows similar trends. Among normal networks depicted in Figures 3.7a-3.7c each method reports a very small likelihood of edge proximity amongst the furthest 1% of nodes, and quickly converges to much higher values. As network size increases the curves shift to the left, indicating the point at which the cumulative distributions converge to 1 occurs further from the network center. Among the skewed networks in Figures 3.8a- 3.8c the 2-hop method reports the greatest number of boundary nodes closest to the edge, though all methods converge on 1 very quickly. As with the normal networks, convergence to 1 shifts further from the origin (a left shift in the curves) as the network increases in size. One important note: in normal and skewed networks the network edge physically occurs further from the origin as the network grows large. Therefore the conclusion that increases in network size are responsible for the increased accuracy represented by curves shifting to the left should be avoided.

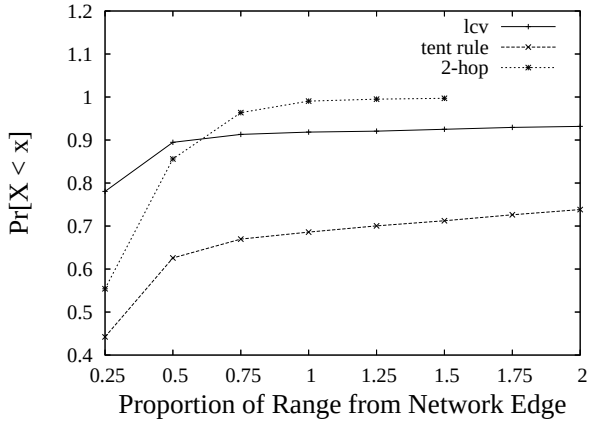
One observation requires special attention. Notice as uniformly random networks increase in size from 1500 nodes in Figure 3.6a to 3500 nodes in Figure 3.6c, the increase of boundary nodes reported by lcv in the outer-most $0.25R$ increases to twice that of the 2-hop method. Further investigation reveals that the 2-hop method suffers from a clustering effect that is due to the way it records hop counts. In a wireless environment where a node location is recorded as the hop-count from a beacon, many neighbouring nodes will share the same hop distance. These neighbours may span



(a) uniform, 1500 nodes

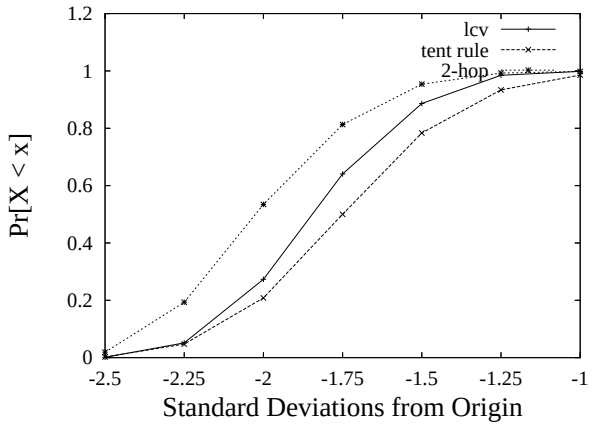


(b) uniform, 2500 nodes

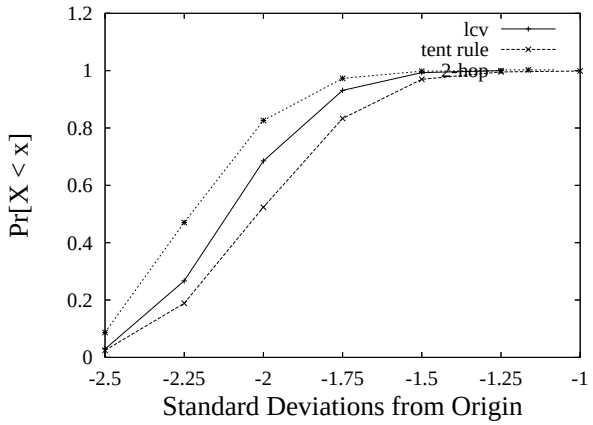


(c) uniform, 3500 nodes

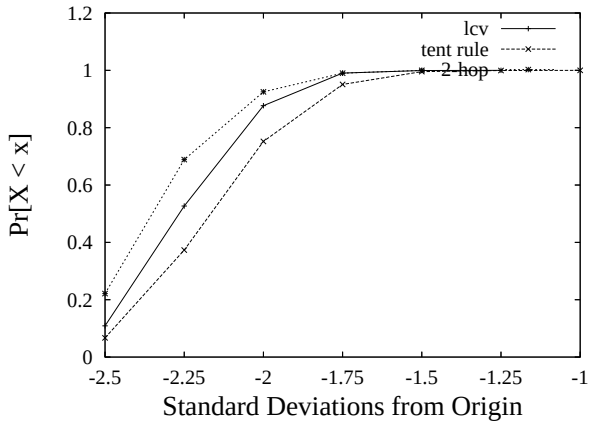
Figure 3.6: (Unform Networks.) Edge proximity distributions reveal the proximity to the network edge of nodes that declare boundary status.



(a) normal, 1500 nodes

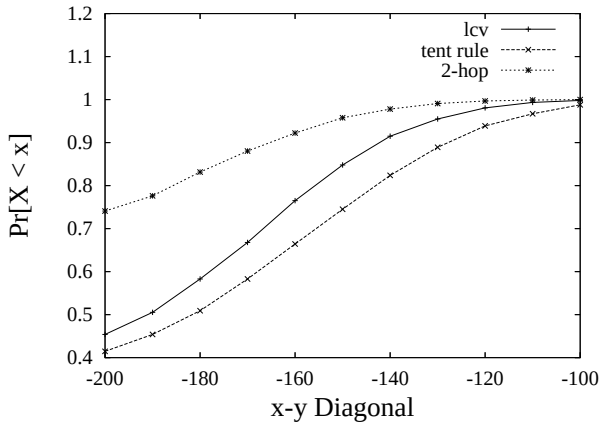


(b) normal, 2500 nodes

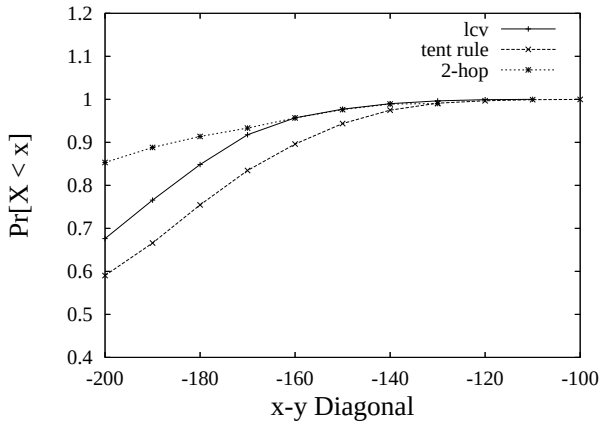


(c) normal, 3500 nodes

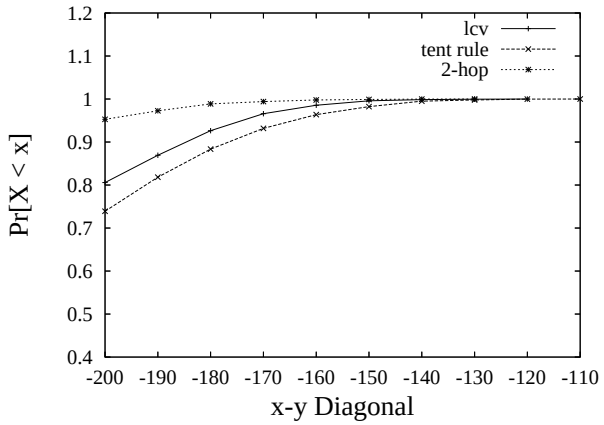
Figure 3.7: (Normal Networks.) Edge proximity distributions reveal the proximity to the network edge of nodes that declare boundary status.



(a) skewed, 1500 nodes



(b) skewed, 2500 nodes



(c) skewed, 3500 nodes

Figure 3.8: (Skewed Networks.) Edge proximity distributions reveal the proximity to the network edge of nodes that declare boundary status.

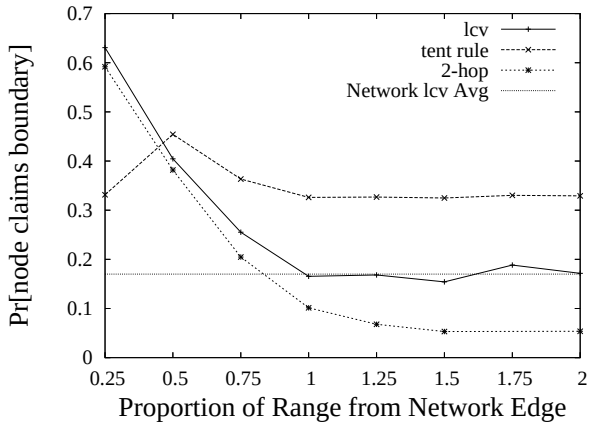
a region as wide as R units. In higher density networks the 2-hop method will heavily concentrate some boundary nodes along some portions of the network edge, while leaving other portions under-represented. This phenomenon is illustrated in Figure 3.12 which shows the boundary nodes as declared by the 2-hop method in networks of 3000 nodes. The conclusion to be drawn is that the 2-hop method is adversely affected by higher density environments.

Regional Proportionality

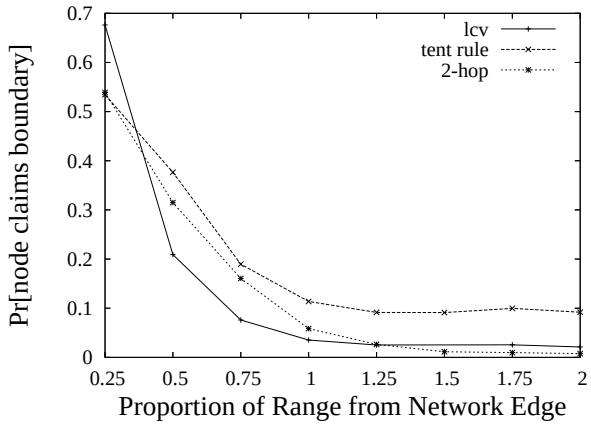
Edge proximity reveals the position of nodes declaring boundary status relative to the edge of the network. In this section we seek further insight by evaluating regional proportionality, the proportion of nodes that declare boundary status within each partition, as described in Chapter 3.2.4, versus those nodes that claim not to be on the boundary. We call this the boundary- vs. regular-node ratio.

We plot regional proportionality for each algorithm, for each network, in Figures 3.9, 3.10, and 3.11. From this view it appears that the tent-rule method produces the most accurate results, that the 2-hop method produces the least accurate results, and that *lcv* lies in between. This interpretation is misleading. It is inappropriate to compare the curves against each other. Rather, it is more appropriate to compare the curves against the network-wide proportion of boundary declaring nodes. The network-wide proportions may be calculated using the values reported in Tables 3.1- 3.3.

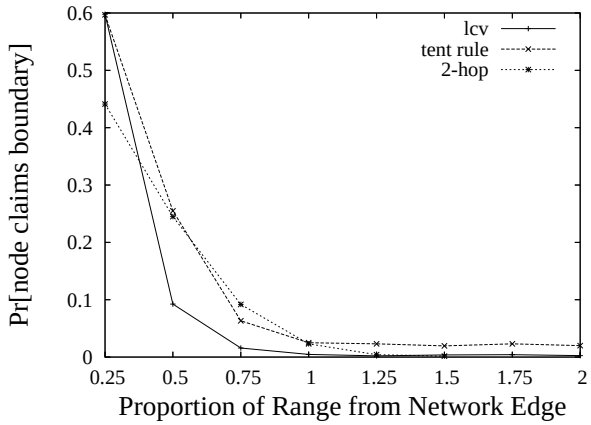
As an example, we direct our reader to the horizontal plots in Figures 3.9a, 3.10a, 3.11a. (We have otherwise omitted these curves for clarity, and due to space limitations.) The horizontal lines represent the network-wide proportion of nodes that declare boundary status using *lcv*. This average permits a clearer interpretation of the results. For example, in Figure 3.9a the network-wide proportion of boundary declaring nodes by *lcv* is 0.17. From the same figure we see that the proportion of boundary



(a) uniform, 1500 nodes

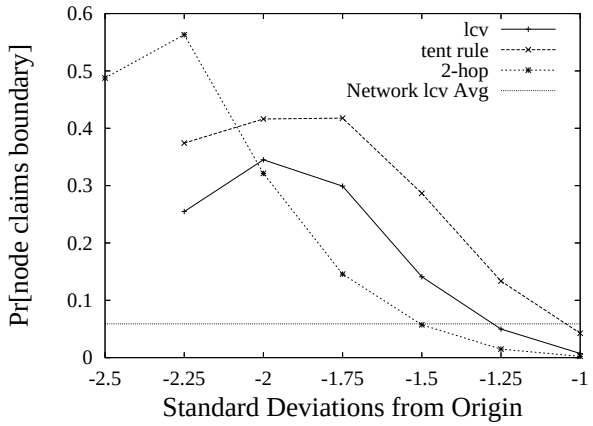


(b) uniform, 2500 nodes

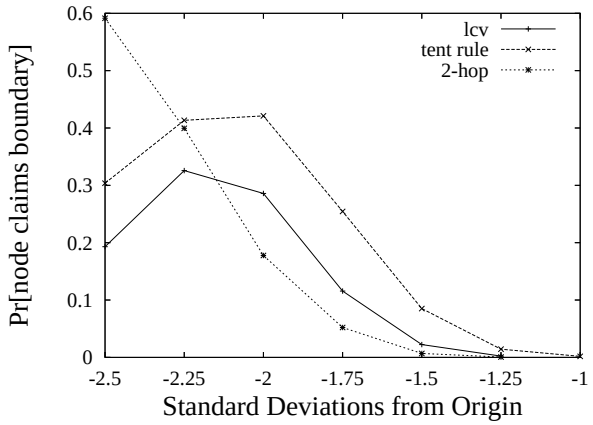


(c) uniform, 3500 nodes

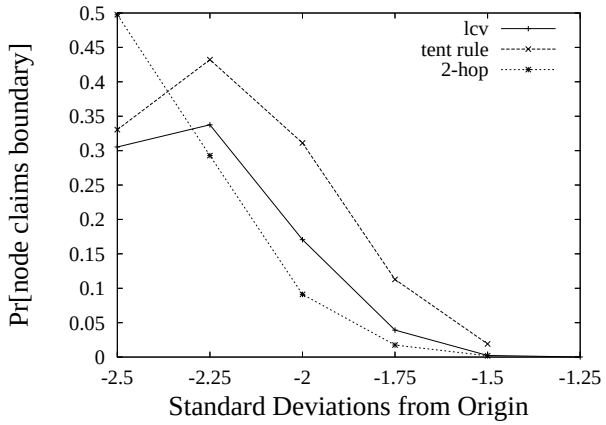
Figure 3.9: (Uniform Networks.) Regional proportionality reveals the proportion of nodes in each region to declare closeness to the network edge.



(a) normal, 1500 nodes

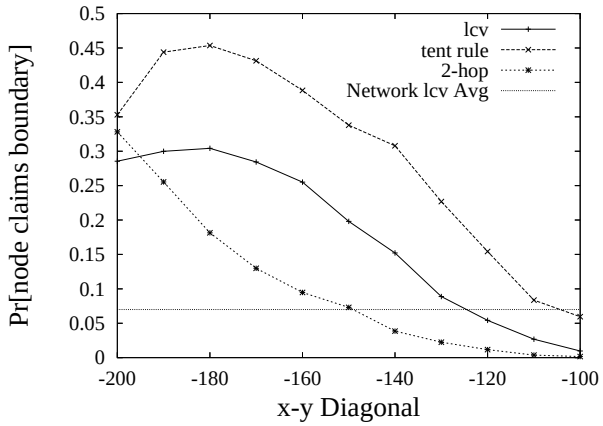


(b) normal, 2500 nodes

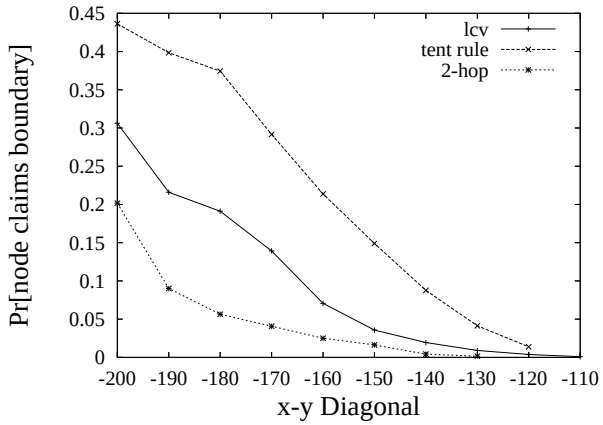


(c) normal, 3500 nodes

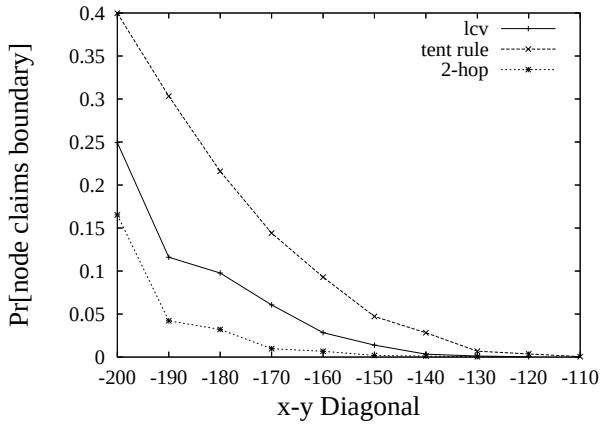
Figure 3.10: (Normal Networks.) Regional proportionality reveals the proportion of nodes in each region to declare closeness to the network edge.



(a) skewed, 1500 nodes



(b) skewed, 2500 nodes



(c) skewed, 3500 nodes

Figure 3.11: (Skewed Networks.) Regional proportionality reveals the proportion of nodes in each region to declare closeness to the network edge.

declaring nodes in the outer-most $0.25R$ region using *lcv* is 0.64. We can conclude that, using *lcv*, nodes in the outer-most ring are almost 4 times as likely to identify with the edge of the network.

Taking this perspective, the reason tent-rule appears to produce more accurate results is that it yields a higher network-wide average. By taking the network-wide averages into account we can observe that sparse uniform, and normal networks in general, are best served by the 2-hop method. In the remaining networks *lcv* produces the highest proportion of boundary nodes closer to the network edge.

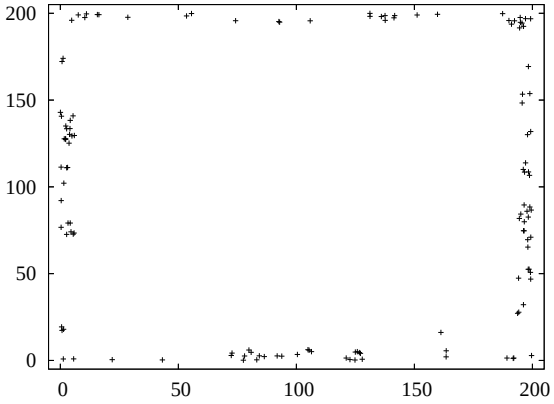
Summary of Comparison

In summary, the *lcv* approach performs consistently across all networks tested. The tent-rule, by design, finds all nodes that abut an unreachable region irrespective of the region's location in the network. By contrast the *lcv* reveals a smaller set of boundary nodes that describe the network boundary more concisely.

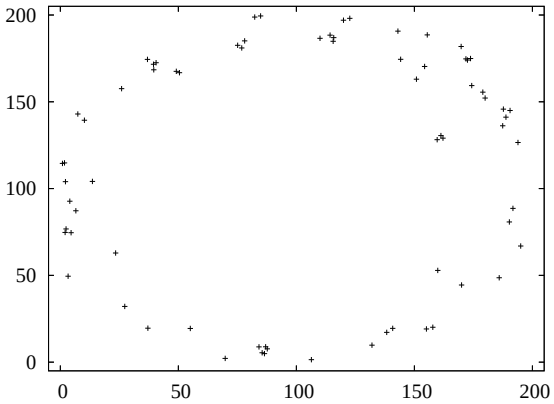
Moreover, the observation that the 2-hop approach produces more accurate results than *lcv* is misleading. Figure 3.12 illustrates that the 2-hop method is adversely affected by network density because the distance to the bootstrap beacon is recorded in hops. Since many neighbouring nodes record the same hop count, the 2-hop approach reveals boundary nodes that are closely clustered together. This leaves many portions of the network edge under-represented. Furthermore, the ability of the 2-hop method to reveal boundary nodes suffers if the bootstrap beacon is poorly placed [101]. The *lcv* suffers none of these drawbacks.

3.2.4 *lcv* and Position Estimation Error

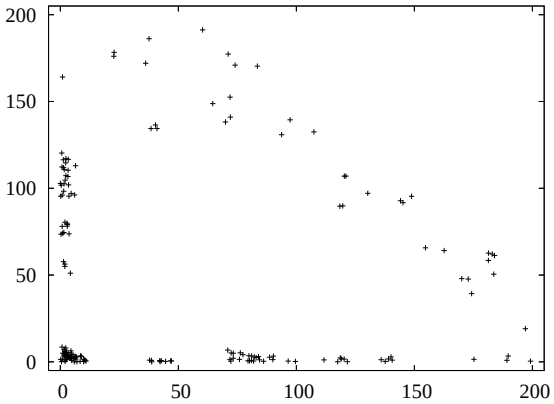
Error is added to the system by blurring the position of nodes from their actual locations. This blurring occurs from the perspective of each node so that two nodes may see a common neighbour in two different positions. Before computing the local



(a) Uniform



(b) Normal



(c) Skewed

Figure 3.12: The 2-hop method is adversely affected by increased density. Boundary declaring nodes cluster together leaving many regions of the network edge under-represented. Example networks are 3000 nodes in size.

convex view, each coordinate is shifted. We shift coordinates by adding a vector consisting of an angle chosen from the uniform distribution, and a length chosen from a parametrized normal distribution. We use the edge proximity and regional proportionally metrics described in Section 3.2.2 to evaluate the efficacy of the *lcv* method in the presence of error.

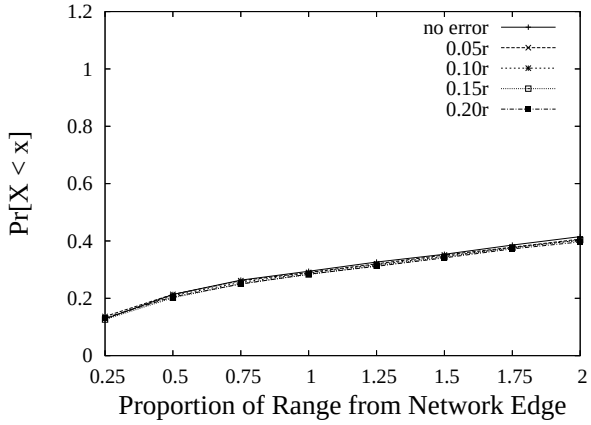
Edge Proximity under Error

Recall from Chapter 3.2.2 that *edge proximity* is defined as the probability that the location X of an *lcv*-node lies in region x. This gives the likelihood that nodes sitting on their local convex view also sit close to the network boundary.

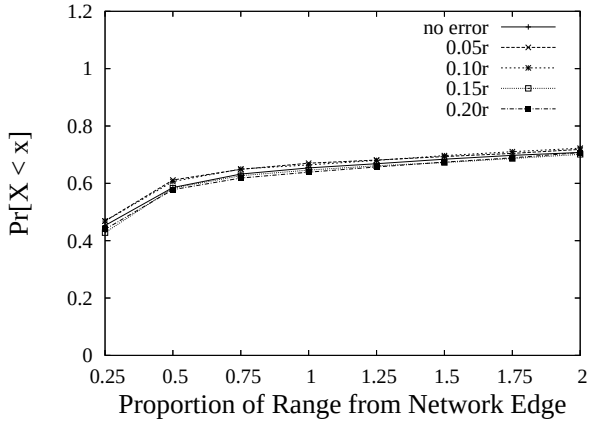
Edge proximity plots appear in Figures 3.13, 3.14, and 3.15. Subfigures are organised such that network size and density changes down each column, while the underlying network distribution varies across figures. For each network we plot the edge proximity for varying error values, parametrized by increasing the variance from 0 to 20% of the communication range. Each plot represents the cumulative distribution over partitions of the network. Networks are partitioned in the manner described in Section 3.2.3.

With respect to the effect of error on the performance of *lcv* we find the observations to be somewhat counter-intuitive. Within each subfigure, each curve represents a different degree of error. Curves within each subfigure show identical trends with differences in accuracy that are largely statistically insignificant. (Confidence intervals have been omitted for clarity.) This would indicate that, in all tested networks, the accuracy of *lcv* is largely unaffected by error. We reserve a discussion of the causes for Section 3.3.

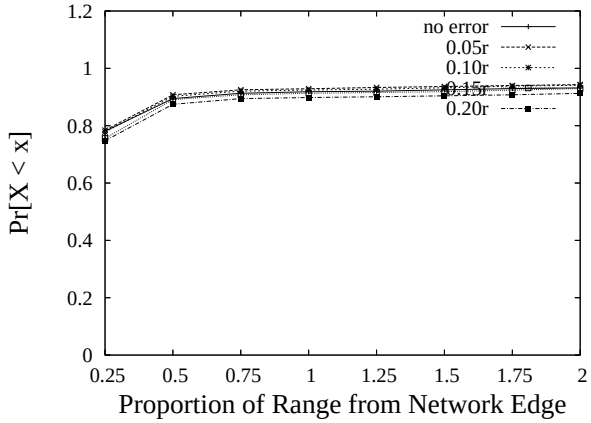
We proceed in the next section with an evaluation using a second metric to confirm our observations.



(a) uniform, 1500 nodes

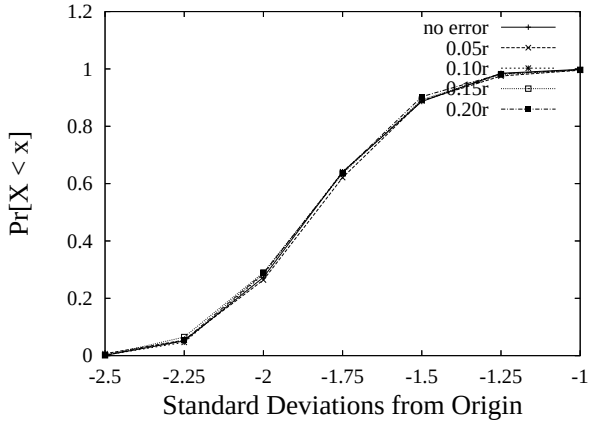


(b) uniform, 2500 nodes

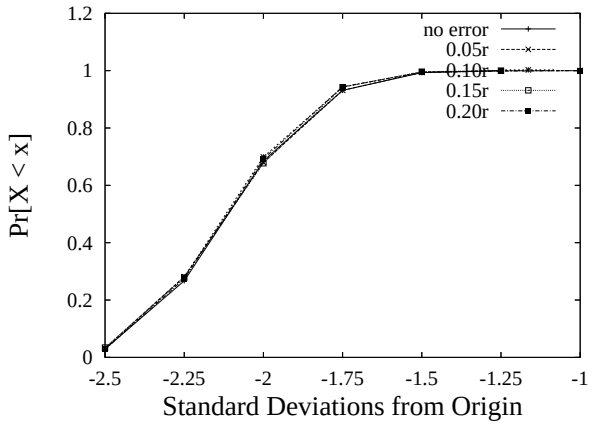


(c) uniform, 3500 nodes

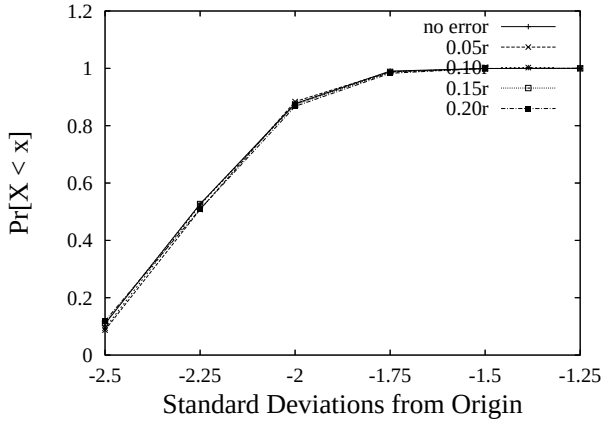
Figure 3.13: (Uniform Networks.) Edge proximity distributions reveal the proximity of *lcv*-nodes to the network edge. Error ranges from $0-20\%$ of communication range, r.



(a) normal, 1500 nodes

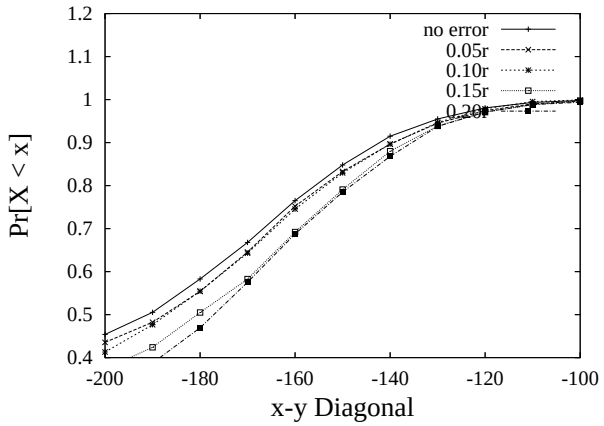


(b) normal, 2500 nodes

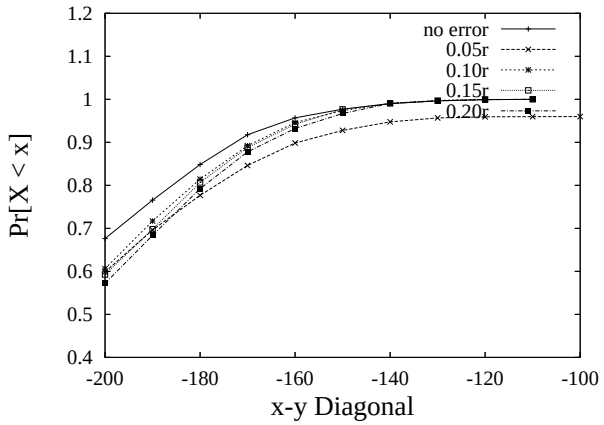


(c) normal, 3500 nodes

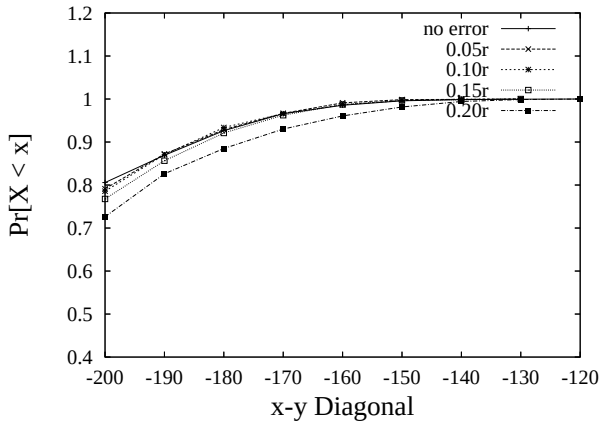
Figure 3.14: (Normal Networks.) Edge proximity distributions reveal the proximity of *lcv*-nodes to the network edge. Error ranges from $0-20\%$ of communication range, r.



(a) skewed, 1500 nodes



(b) skewed, 2500 nodes



(c) skewed, 3500 nodes

Figure 3.15: Skewed Networks.) Edge proximity distributions reveal the proximity of *lcv*-nodes to the network edge. Error ranges from $0 - 20\%$ of communication range, r.

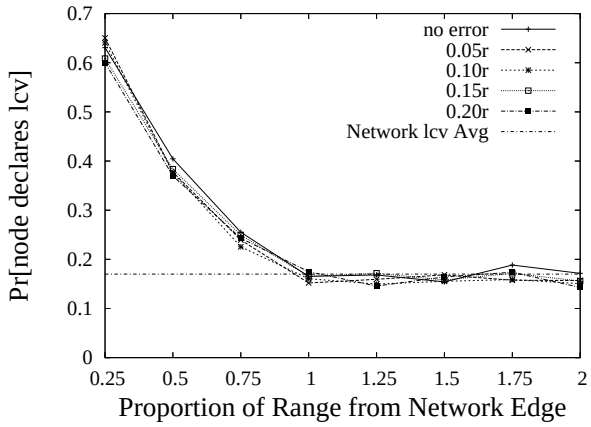
Regional Proportionality under Error

We find that, using the edge proximity metric, *lcv* seems largely unaffected by errors in position estimation. In this section we seek further insight by evaluating regional proportionality. *Regional proportionality* is the proportion of nodes that declare *lcv* status within each partition as described in Section 3.2.3, versus those nodes that claim not to be *lcv*. We call this the *lcv*- vs. regular-node ratio.

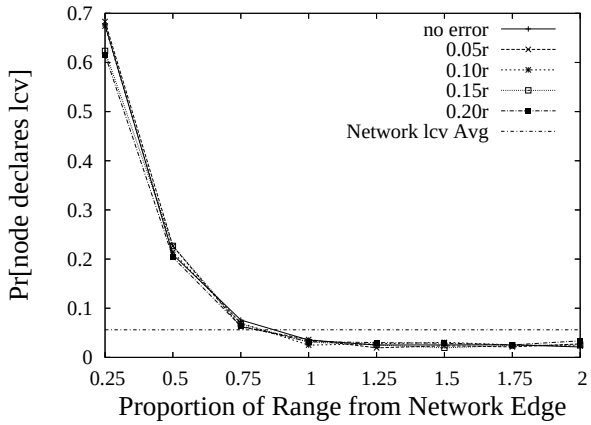
We plot regional proportionality in Figures 3.16, 3.17, and 3.18. Here, too, subfigures are organised such that network size and density changes down each column, while the underlying network distribution varies across figures. For each network we plot the edge proximity for varying error values, parametrized by increasing the variance from 0 to 20% of the communication range. Note the leftward shift of curves as size increases among normal and skewed networks. Recall from previous that this is an artifact of the network's edge shifting further from the origin as the networks grow.

Within each subfigure we can compare the curves against the network-wide proportion of *lcv*-nodes, represented by the horizontal line. The network average permits a clearer interpretation of the results. For example, in Figure 3.16a the network-wide proportion of *lcv*-nodes is 0.17. From the same figure we see that the proportion of *lcv*-nodes in the outer-most $0.25R$ region is 0.64 when there is zero error. We can conclude that, with no error, nodes in the outer-most ring are almost 4 times as likely to identify with the edge of the network.

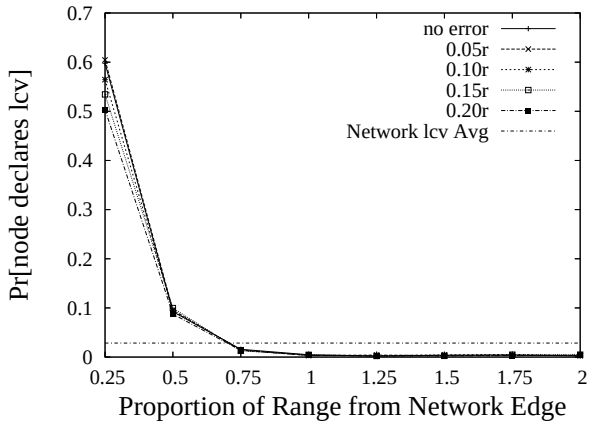
The regional proportionality metric seems to reinforce the observation that error, as tested, has little-to-no effect on the performance of *lcv*. Similar to the edge proximity metric in Section 3.2.3, plots within each subfigure show identical trends with differences in accuracy that are largely statistically insignificant. However, there are subtle noteworthy observations. We refer our reader first to plots derived from uniformly generated networks in Figures 3.16a- 3.16c. We can see that error has a



(a) uniform, 1500 nodes

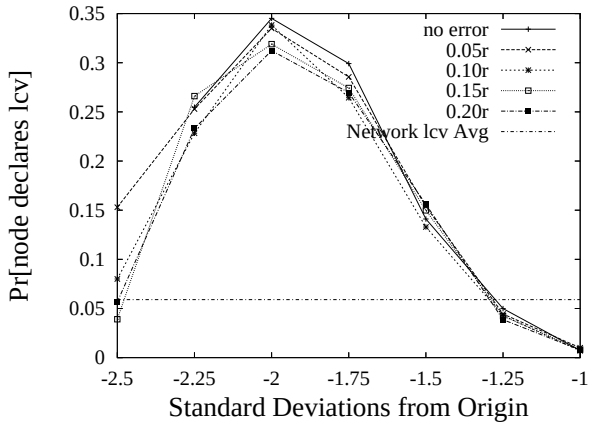


(b) uniform, 2500 nodes

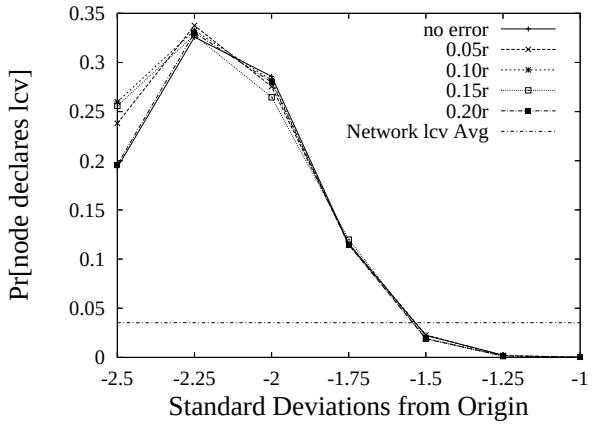


(c) uniform, 3500 nodes

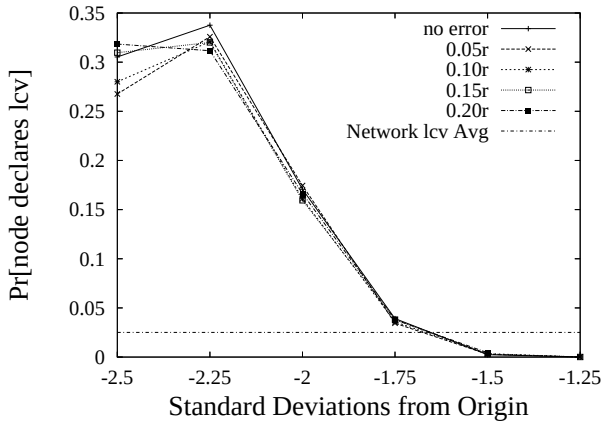
Figure 3.16: (Uniform Networks.) Regional proportionality reveals the proportion of nodes in each region to declare edge-node status via *lcv*. Error ranges from $0 - 20\%$ of communication range, r.



(a) normal, 1500 nodes

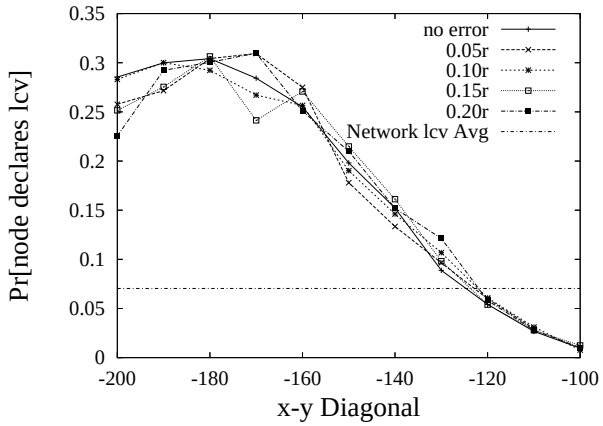


(b) normal, 2500 nodes

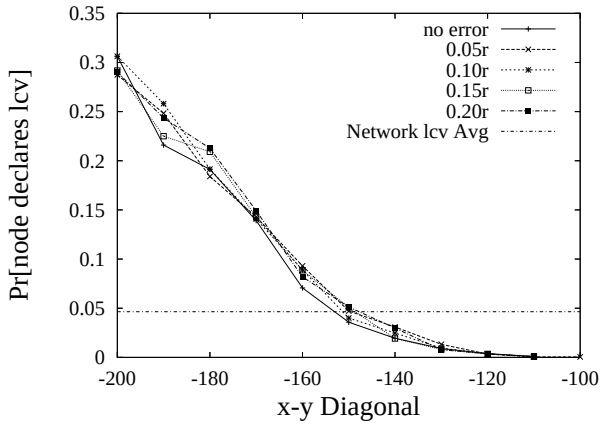


(c) normal, 3500 nodes

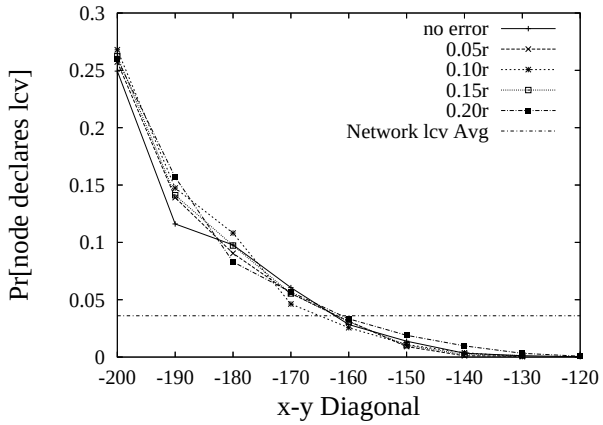
Figure 3.17: Regional proportionality reveals the proportion of nodes in each region to declare edge-node status via *lcv*. Error ranges from $0 - 20\%$ of communication range, r.



(a) skewed, 1500 nodes



(b) skewed, 2500 nodes



(c) skewed, 3500 nodes

Figure 3.18: Regional proportionality reveals the proportion of nodes in each region to declare edge-node status via *lcv*. Error ranges from $0 - 20\%$ of communication range, r.

more pronounced effect on *lcv* accuracy as the network density increases, but only in the outer-most region of the network.

We emphasize that the differences in accuracy, statistically speaking, are insignificant - with one exception. A dense uniformly generated and bounded network will eventually capture the shape enforced by the bounds. In our experiments this shape is a quadrilateral. It is directly responsible for the loss in accuracy in the outer-most ring of the network, as density increases. We develop this idea next in Section 3.3.

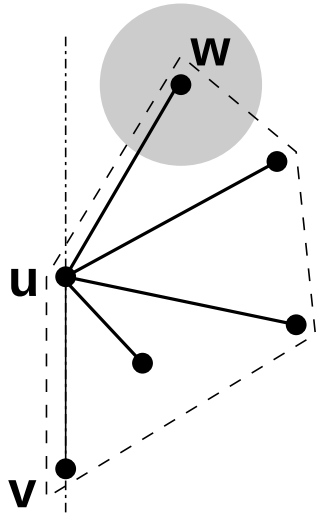
3.3 On the Resilience of *lcv* to Error

The previous section revealed little-to-no degradation in the performance of *lcv* in the presence of errors. This idea is counter-intuitive, and so we use this section to enumerate and discuss the scenarios faced by the local convex view method.

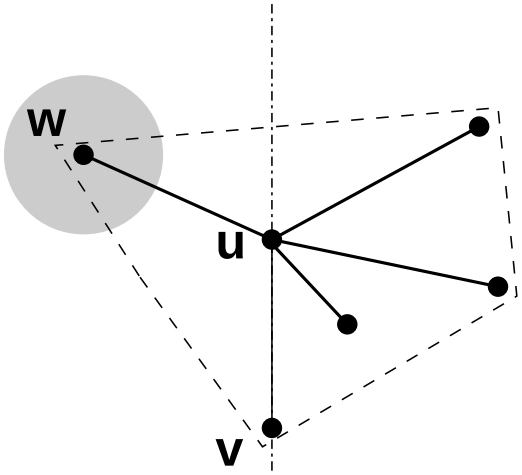
In our analysis we assume that the position of only a single node has been incorrectly estimated. This relatively benign assumption permits a clear demonstration of the effects of error on *lcv* without sacrificing accuracy or completeness. Specifically, all remaining cases may be composed of the cases presented here.

The three cases under consideration by *lcv* are presented in Figure 3.19. For the purpose of demonstration, we consider the *lcv* operation at node u. In our example topologies, neighbours are joined to u with a solid line. The position of some neighbouring node v determines a dash-dot-dashed line that represents a threshold of interest. The neighbour in question has a position estimated by the node labelled w, with a true position that may exist anywhere inside the greyed region. Finally, the dashed poly-line corresponds to the local convex hull under consideration.

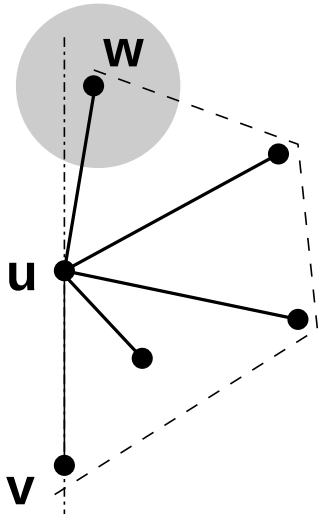
Figures 3.19a and 3.19b depict the two 'good' cases where the *lcv* computation is unaffected by error. In the first case, shown in Figure 3.19a, node u determines it is on the local convex view and declares itself close to the network boundary. Note that the local convex hull consists of the same nodes irrespective of the actual location



(a) Consistent



(b) Consistent



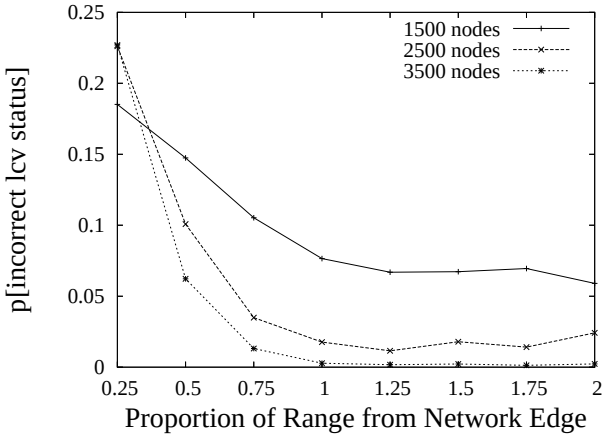
(c) Inconsistent

of w anywhere inside the grey region. The second case shown in Figure 3.19b is of a similar theme. Node w is estimated to have a location that renders node u inside its local convex view. In fact, node w may sit anywhere in the grey region without affecting the local convex view. In both these cases the underlying geometry ensures the resilience of the local convex view method: the convex hull remains consistent so long as the error region of w remains entirely to one side or the other of the threshold determined by (u, v).

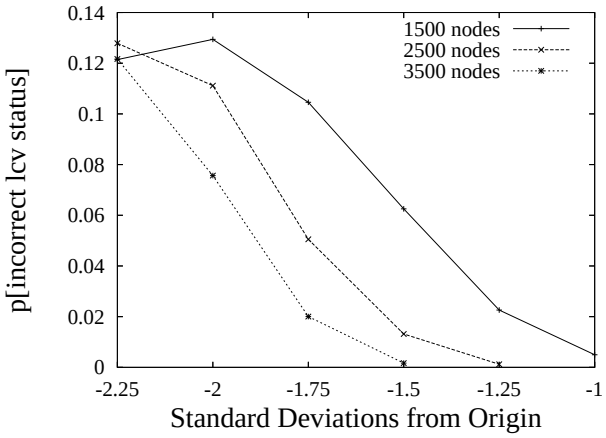
Figure 3.19c depicts the ambiguous case. Node w is estimated to have a position close enough to the threshold that its actual location may exist on either side of the threshold. In the example shown in Figure 3.19c, a w is estimated to have a position that renders u on the local convex view of u. If such a w actually sits on the the other side of the threshold then u has falsely determined that it sits on its local convex view. The reverse may occur if w is estimated to sit close to, but on the other side of, the threshold.

The observation in Section 3.2.4 is that *lcv* seems relatively unaffected by error. From our analysis we have determined that any adverse effect of error to *lcv* is caused by the ambiguous scenario demonstrated in Figure 3.19c. Our hypothesis is that *lcv* is relatively unaffected by error because the ambiguous case occurs very infrequently. To test our hypothesis we evaluate the frequency of false positives and false negatives when error is added to *lcv*. For each type of network the results are partitioned as described in Section 3.2.4 so that we may observe *lcv* performance in each area of the network. We plot for all networks the worst tested case in Figure 3.20, where the variance parameter is equal to 20% of the communication range.

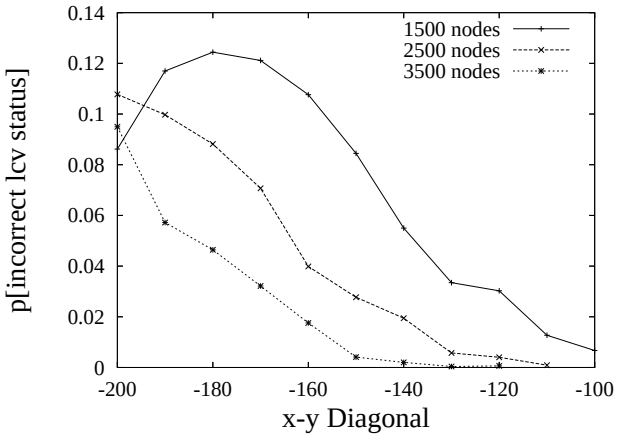
Figure 3.20a reveals that, in uniform networks, the error in the outer-most ring of the network hovers about 20%. Interestingly, the frequency of false positives and negatives in this region climbs as density increases. The reason is that increased densities along the network edge more closely approximate the artificial lines artificially bounding the network. This causes a greater number of w nodes which are the cause



(a) Uniform



(b) Normal



(c) Skewed

Figure 3.20: Frequency of *lcv* false positives and false negatives in the worst tested case, with error variance 20% of communication range.

of the ambiguity that leads to false positives and negatives. As we move deeper into the network where no artificial boundaries exist, the frequency of incorrect responses drops dramatically.

In the outer-most regions of normal and skewed networks, the ambiguous case appears much less frequently. For example, among the outer-most 5% of nodes in normal networks, the rate of false *lcv* responses is approximately 10%. Similarly for skewed networks. This may be explained by the lack of artificial boundaries in normal and skewed networks (ie. unlike the uniformly generated networks, these networks fail to approximate the quadrilateral region that contains them). As we move deeper into the network the rate of false positives quickly drops in both normal and skewed networks.

3.4 Chapter Summary

This chapter has presented and compared methods to identify a subset of nodes on the network boundary (for sensing and localising applications). Its key contribution is a heuristic algorithm that operates locally. A node decides it is close to the network edge if the node finds that it lies on the convex hull of its 1-hop neighbourhood. Where position information is unavailable *lcv* assigns local coordinates to each neighbour so that geometric relationships, and hence the convex hull, may be computed. Coordinate assignment requires that a node view its neighbourhood as 2-connected, ie. if a node removes itself then a single connected component remains. We have determined the maximum possible number of disconnected components to be 5. In such cases we have shown that it is possible for a node to sit on the local convex hull only where removal of the node leaves a maximum of 3 components. A simple probabilistic model was proposed to decide the convex hull of neighbourhoods with disconnected components.

We simulated *lcv*, tent-rule, and 2-hop methods in networks of varying density

constructed from uniform, normal, and skewed (Pareto) distributions. For evaluation we identified two metrics, edge proximity, and regional proportionality. We found that the behaviour of all three methods demonstrated similar trends. According to these metrics the tent-rule was the least well-performing, while the 2-hop method performed best. However, this conclusion is misleading as the 2-hop method also reveals boundary nodes that are tightly clustered, while ignoring other boundary nodes altogether. *lcv* was found to be resilient to the qualitative drawbacks of the 2-hop approach.

We further examined the ability of the local convex view (*lcv*) algorithm to identify network edge nodes in the presence of position estimation error. We engaged in extensive simulations of networks with topologies of varying size and underlying distributions. Position errors were chosen from a normal distribution with a variance up to 20% of the communication range.

Further examination failed to reinforce the assumption that *lcv* would be adversely affected by position estimation error. To explain the disconnect between intuition and observation we enumerated and analysed the three base neighbour configurations that may be seen by a node. In two cases position estimation error changes the shape of the local convex view, but not the nodes that comprise it. In the third case position error leads to ambiguity, where the true position of a neighbour may lead to a false insertion or an omission of the node undergoing the *lcv* computation. Further simulation revealed the frequency of the ambiguous case to be very low, about 10% in the worst case for all networks tested. We conclude that the geometric properties underlying *lcv* are responsible for its resilience to error.

In the next chapter we transform *lcv* into a deterministic algorithm that outputs a descriptive map of the network boundaries.

Chapter 4

A Deterministic and Local Boundary Detection Algorithm

4.1 Introduction

Context-awareness is increasingly important in wireless and sensor networks. When available, knowledge of position, nearby physical obstacles, or topological features, can be exploited to provide better communication protocols and deployment techniques in resource constrained environments.

Intuitively, many pure sensing applications benefit from knowledge of network boundaries ([8, 12, 24, 28, 71, 95, 96, 101, 110, 122, 126]). Nodes along the outer edge of the network, for example, are assumed to be the best candidates for beacons in virtual coordinate constructions. Here the assumption is that the finest resolution in coordinates appear using a set of beacons that are furthest apart. Perceived network edges also may bound holes in the network or other regions of interest. Such regions may indicate physical boundaries or node failures due to environmental effects, so that additional nodes may be deployed. In addition, there are applications that benefit directly. KAT [92], whose success relies in part on accurate knowledge of network

boundaries, is one such example.

In this paper we solve the edge detection problem locally using a geometric structure called the alpha-shape (α-shape) [20]. The α-shape is used to capture the shape of a set of points in space, and is a generalisation of the convex hull. In addition to geometry-related fields of study such as graphics and computational geometry, α-shapes have been used in molecular biology and particle physics [21]. Alpha-shapes offer real world applicability. They may be weighted to reflect degrees of accuracy relative to network positioning [19], and also extended to three-dimensions [21].

Its use is motivated by the hypothesis that within range of many nodes there exists structural information relevant to the network. For a disc of radius $1/\alpha$, the α-shape consists of nodes (and joining edges) that sit on the boundary of the discs that contain no other nodes in the network. For this initial study we restrict ourselves to network graphs with normalised communication range in two-dimensions.

Our contribution, rather than to suggest a new method, is to identify wireless network boundaries by combining previously unrelated methods. It differs from previous methods in that we investigate what might be achieved if relative information - positions of nodes relative to their neighbours - was known or computable. First, each node constructs a local coordinate system. (Alpha-shape computations are unaffected by translations and rotations in space.) Next, each node computes the Delaunay triangulation of its neighbourhood to find the corresponding α-shape. In the simplest terms a node decides it is on a network boundary by asking the following question: "Do I sit on the boundary of a disc of radius $1/\alpha$ that contains no other nodes in the network?" Finally, any boundary node may request a map of the boundary by transmitting a discovery packet along edges of the α-shape using right-hand rule.

The key to localisation is to select the α-parameter appropriately. In the version of the problem faced in this paper it is appropriate to select α so that $1/\alpha = 1/2R$, where R is the normalised communication range. Given such an α, the α-shape derived from local computations is provably correct. Even so, the *alpha*-shape may

expose some unwanted detail. This may be resolved by using either of two refinement methods.

We show via simulation that our algorithm identifies meaningful boundaries even in low-density networks. In addition to varying density, we vary topology by generating networks where node locations are selected from uniform, normal, and skewed (Pareto) distributions. We complete our evaluation with a comparison of local α-shaping over dædal topologies presented in [119]. We find that α-shaping produces similarly favourable results with fewer communications and fewer neighbours.

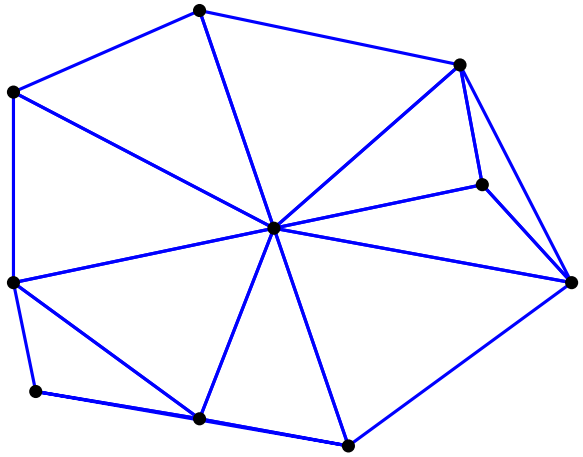
4.2 Preliminaries

Our work exploits the properties of many well known geometric structures. In advance of the presentation of the edge detection algorithm we present some necessary definitions and background for completeness.

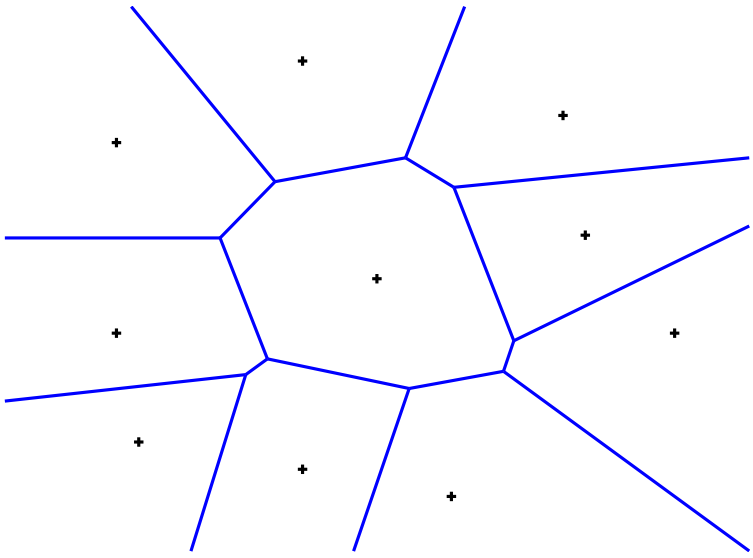
4.2.1 Definitions

Delaunay Triangulation. Given a set of points S the Delaunay triangulation DT_S is one that satisfies the 'empty circle' property, where no point lies inside the circumcircle of any triangle in DT_S. An example of a Delaunay triangulation appears in Figure 4.1a. It is a super set of the convex hull as well as the minimum spanning tree of S. The number of edges in the Delaunay triangulation is on the order of the number of nodes (ie. $|E| = O(|V|)$ for edge set E and node set V). Of its many properties we are most interested to its relationship to Voronoi Diagrams.

Voronoi Diagrams. Given a set of points S the Voronoi diagram VD_S partitions the space occupied by S into convex regions $V(p)$ where, for each p in S, any point in $V(p)$ is closer to p than any other point in S. The Voronoi diagram is the dual graph of the Delaunay triangulation. The Delaunay triangulation may be used



(a) Delaunay Triangulation



(b) Voronoi Diagram

Figure 4.1: The Delaunay triangulation of a set of points and its corresponding Voronoi diagram.

to compute the corresponding Voronoi diagram in linear time. Referring to Figure 4.1 we can see both the Delaunay triangulation and the Voronoi diagram for the same set of points.

4.2.2 α-Shapes

The α-shape provides the foundation of the work in this paper. It is derived from the α-hull, which is a generalisation of the convex hull. To better understand α-shapes we restate the following definitions from the original work in [20] in a manner that better suits our application, and follow with a discussion using examples. Say that we have a set S of points in the plane.

Definition 4.2.1 *The α-hull is defined as the intersection of the complement of all closed discs of radius $1/\alpha$ that contain no points in S.*

We are more interested in a related structure called the α-shape which first requires the following definition.

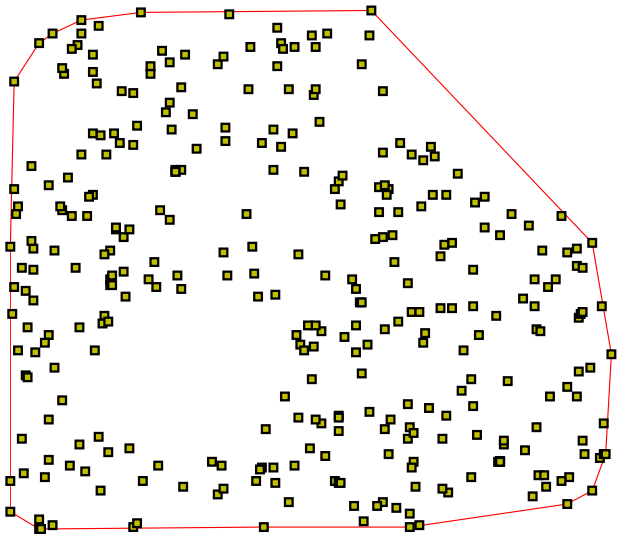
Definition 4.2.2 *A point p in S is said to be α-extreme if p lies on the boundary of a closed disc of radius $1/\alpha$ that contains no other points of S. Two such points p and q that lie on boundary of the same disc are said to be α-neighbours.*

Finally, we may define α-shapes as follows.

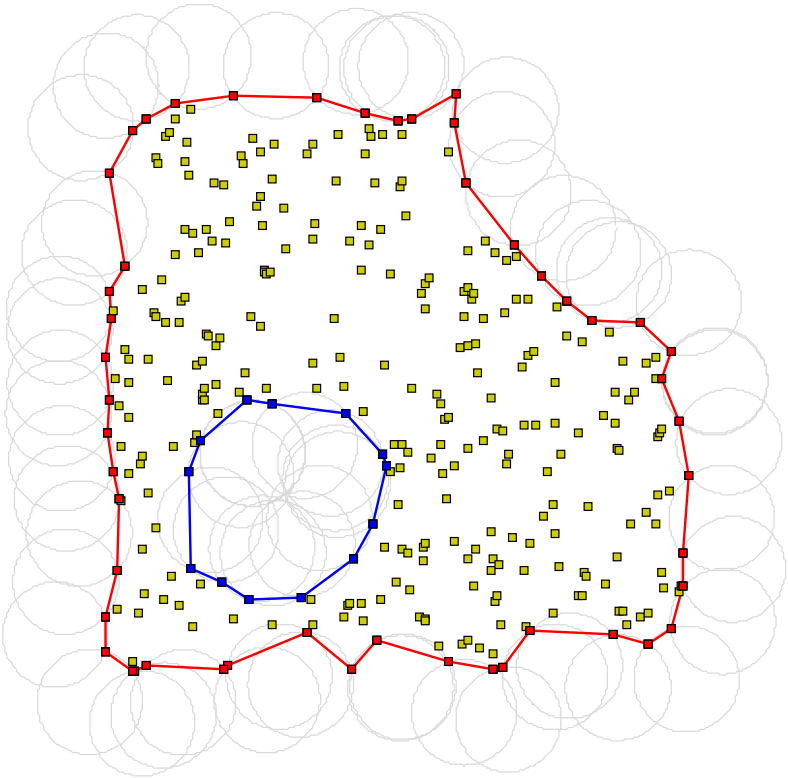
Definition 4.2.3 *For a set S of points in the plane and $\alpha \geq 0$, the α-shape is the graph whose vertices are α-extreme and whose straight line edges connect α-neighbours.*

Note that as α approaches zero the size of the disc approximates a half-plane. At $\alpha = 0$ the α-hull is identical to the convex hull.

The contrast between structures may be seen in Figure 4.2. The convex hull of an example set of points appears in Figure 4.2a. Certainly the convex hull of a



(a) Convex Hull.



(b) α-shape.

Figure 4.2: The convex and α-shapes of a corresponding set of points.

wireless network is one way to define its boundary; unfortunately the convex hull fails to reflect numerous features. Observe, for example, that the curve along the upper-rightmost edge is hidden, as is the large circular gap that appears in the lower-left. Furthermore edges of the convex hull, restricted by range constraints inherent among wireless devices, may be impossible to compute.

Using the same set of nodes we compare the α-shape shown in Figure 4.2b. For demonstration the discs used to produce the α-shape appear using greyed lines. Observe that the revealed shape more accurately reflects the shape of the network (for proper selections of α). The α-shape closely follows variations in the outer-edge. It also reveals the inner gaps. We note that the α-shape can reveals inner as well as outer boundaries, a characteristic missing from the convex hull.

The structures discussed above are well-known, well-studied, and have been extended to 3-dimensions. The algorithm presented in the next section uses these structures to a provide boundary detection localised for wireless and sensor networks.

4.3 Localised Boundary Detection Algorithm

We consider a deployment of a large sensor network where, initially, nodes may lack any knowledge of their positions. We wish to identify, using only local information, those nodes and links that lie on the network boundaries. We propose a localised algorithm that makes only two assumptions. First, that generated or assigned node IDs are unique within each neighbourhood; second, that distance measurements are available should position information be unknown.

Our method relies on the hypothesis that within the local view of each node there exists some structural information relevant to the network. As discussed in Section 4.2.2, it is well known that α-shapes, given a proper selection of the α-parameter, reveals a set of nodes and edges that captures the shape of a set of points in a geometric space. We first outline our algorithm and then discuss each step in

detail. The correctness of the algorithm is reserved for discussion to Section 4.4.

1. (Optional) Each node i constructs a local coordinate system consisting of the nodes $L(i)$ within communication range. Only relative positions are required for the algorithm to succeed. This step is necessary only if position information is unknown.

2. Each node i for its neighbourhood $L(i)$, constructs the Delaunay triangulation $DT_{L(i)}$.

3. Given communication range R, select the α-parameter so that the radius r of the disc is $r = \frac{1}{2}R$. Identify 'α-extreme nodes' of $L(i)$ as determined by r. Each node ascertains its boundary status by asking, "Am I α-extreme?"

4. (Optional) A map of the boundary may be obtained by sending a discovery packets according to right-hand rule along the edges joining α-neighbours.

4.3.1 Establish a Local Coordinate System

In the first step each node constructs a local coordinate system so that the α_L-shape may be computed in the next steps. Though not the focus of this work, we present this step for completeness. It is described for cases where no a priori position information exist. Should position information be available, this step becomes redundant and may be omitted.

Each node in the network, after announcing its presence, begins by sharing with its immediate neighbours a vector of measured distances to 1-hop neighbours. Once received, each node constructs a local coordinate system by placing itself at the origin and, in a depth-first manner, assigns coordinates to remaining neighbours relative to those whose local coordinates have been established. We demonstrate this idea from the perspective of node u in Figure 4.3. Node u places itself at the origin of a

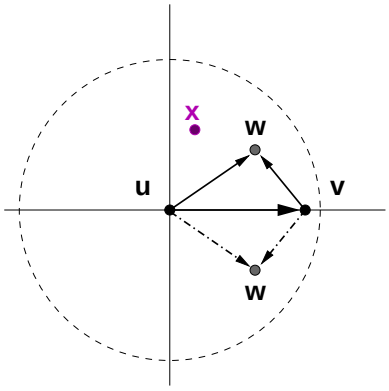


Figure 4.3: Node u may generate a local coordinate system, if needed.

Cartesian space and sits neighbour v on the horizontal axis. (For our purposes we use the furthest neighbour.) The next node, w, may be placed on either side of the horizontal axis since α-shape computations are resilient to rotations and translations. Remaining nodes are placed similarly and with respect to established coordinates.

Clearly this approach is independent of under- and over-lying protocols. Still, its weakness is its dependence on accurate measurement methods. The viewpoint we take is that location-discovery technologies such as [72, 112] or GPS will continue to decrease in size and cost, rendering this argument moot.

Once nodes have discovered their positions relative to their neighbours, each node proceeds to construct the Delaunay triangulation consisting of the nodes in view.

4.3.2 Local Construction of the Delaunay Triangulation

Construction of the α-shape of a set of points in the plane begins with the construction of their Delaunay triangulation. The communication range restricts us to the unit Delaunay triangulation, where edges longer than the communication range are omitted. Localised constructions of the unit Delaunay triangulation are known not to exist (see [4, 37, 80]). Despite this fact we show in Section 4.4.1 that it suffices for our purposes to build only the Delaunay triangulation of the local neighbourhood.

4.3.3 Boundary Recognition using Local α-shapes

In Step 3 each node i independently determines whether it sits on a network boundary by computing its $\alpha_{L(i)}$-shape, the α-shape of its neighbourhood $L(i)$. Recall from Section 4.2.2 the definitions of α-extreme. From the Delaunay triangulation $DT_{L(i)}$ each node i finds the α-extreme points in linear time as follows:

a) Node $p \in L(i)$ lies on the convex hull of $L(i)$; this is the trivial case where node p is α-extreme.

b) Node $p \in L(i)$ is not on the convex hull of $L(i)$; we consider the discs (ie. circumcircles) as defined by the Delaunay triangulation of $L(i)$. Recall from Section 4.2.1 that the points q which define the convex region V_p enclosing p in the Voronoi diagram, are the centres of the circumcircles touching p in the Delaunay triangulation. Thus any p is α-extreme if $r \leq dist(p, q)$ for any $q \in V_p$. (Edges between α-neighbours are similarly identified.)

Finally, each node asks of itself if it is α-extreme. No node may declare any other node as α-extreme. This restriction avoids many of the pitfalls that plague other localised methods such as [60]. We show in Section 4.4 that this decision constraint, in addition to our selection of α, is necessary for correctness.

4.3.4 Mapping the Network Boundaries

We emphasise that, following Step 3, the 'network' has identified all of its boundaries. Collectively, the information stored at *alpha*-extreme nodes constitutes the α-shape of the network for $1/\alpha = r$. Still, it may be advantageous to map the network boundaries. The α-shape, as a subgraph of the Delaunay triangulation, is a planar graph. This fact permits nodes to map network boundaries by routing a discovery packet along edges joining α-neighbours using the right-hand rule : Upon receipt of a discovery packet, an α-extreme node forwards along the next edge in angular order.

The combination of right-hand rule over a planar network graph guarantees the return of a discovery packet to its origin node. In the next section we show that the local α-shape algorithm is correct for certain selections of α.

4.4 Algorithm Correctness

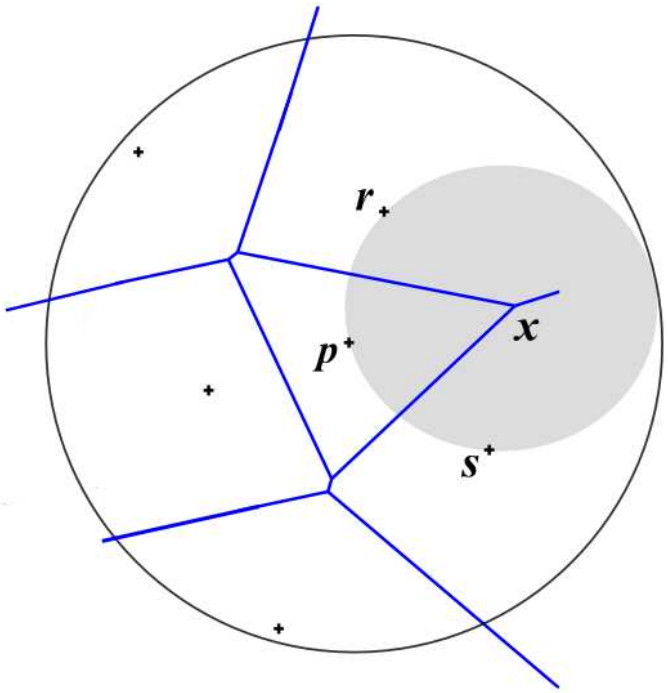
In this section we demonstrate the correctness of the local *alpha*-shape algorithm. We show that when communication range is normalised, local computation is sufficient and reveals the same nodes and edges as if computation was centralised.

4.4.1 Local Delaunay Triangulations Suffice

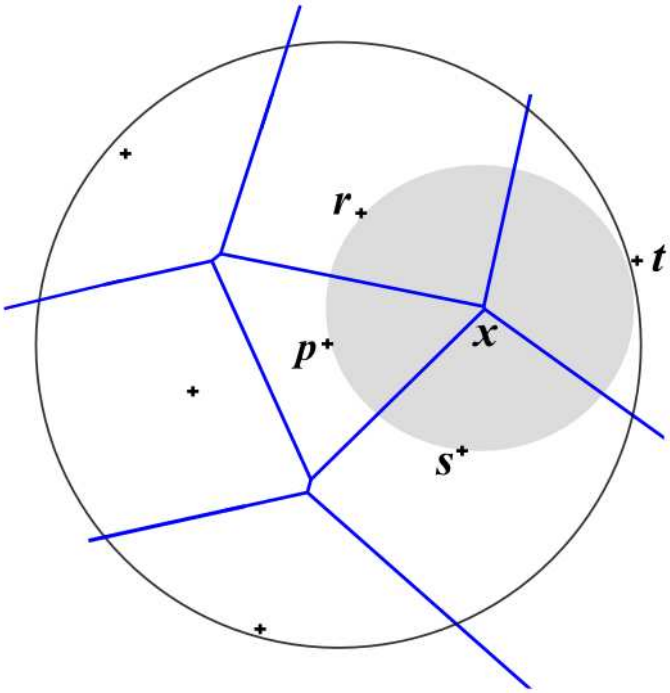
The α-shape of a set of points S may be found using their Delaunay triangulation $DT(S)$ and corresponding Voronoi diagram $VD(S)$. We show by example that, if the communication range is normalised, the events outside of communication range bear no affect on the correctness of local α-shape computations.

We direct our reader to the example in Figure 4.4, which depicts a 'before and after' scenario of a neighbourhood from the perspective of sensor node p. The large circle represents p's communication range; the greyed disc is the disc used to find α-extreme nodes and edges; the straight lines represent Voronoi diagrams. The Voronoi diagram computed by p is shown in Figure 4.4a. The intersection x is the center of the circumcircle of prs and reveals that p is α-extreme. We then add a node t just beyond p's view in Figure 4.4b and show how the Voronoi diagram would change if t was visible to p. Note that x is no closer to p, so p remains *alpha*-extreme. Similar examples may be constructed for any t outside of communication range.

The key to this observation is the bisector property of the Voronoi diagram where any line or point between nodes p and q sits on the line that bisects the space between the points. Next we show that it is possible to ascertain the *alpha*-shape of



(a)



(b)

Figure 4.4: (a) Local information is sufficient to find α-extreme nodes. (b) Point x in the Voronoi diagram can be no closer to p despite node t outside of range.

the network, despite only local information and computation.

4.4.2 On the Proper Selection of Alpha

Generally speaking, the value of an α-shape rests in the selection of the α-parameter. Edge detection is no different. We ask when, if ever, a locally constructed α-shape might be identical to an α-shape that is constructed in a centralised fashion. For a set of points S in the plane, let $\alpha(S)$ be the set of nodes and edges collected from the centralised α-shape algorithm; similarly we label the set of nodes and edges collected from local operations as described in Section 4.3 as $\alpha_l(S)$.

Theorem 4.4.1 *The sets $\alpha_l(S)$ and $\alpha(S)$ are identical sets.*

Proof. It suffices to consider only the edges in a set since, if any node differs, there must be an edge that differs as well. If we assume that the two sets differ then either or both of the following statements must be true:

a. An edge in $\alpha(S)$ fails to appear in $\alpha_l(S)$.

b. An edge fails to appear in $\alpha(S)$ that appears in $\alpha_l(S)$.

We show both statements to be false. The key is in the selection of α which, in the unit disc graph, is selected so that the radius of the disc $r = 1/2$. When finding the global α-shape and setting α so that $r = 1/2$, only Voronoi neighbours (or their dual, Delaunay edges) whose distance is ≤ 1 may be inserted into $\alpha(S)$. We can satisfy this restriction by searching $UDel(S)$, the Delaunay triangulation with edges greater than 1 unit removed. So, in $\alpha(S)$ we have no edge greater than 1 and only edges whose endpoints sit on an empty circumcircle with $r = 1/2$.

The reverse statement presents a greater challenge since there is no way to construct $UDel(S)$ locally [4, 37, 80]. Instead we can effectively find all edges that join α-neighbours. Observe in Step 3 of our algorithm that any edge e can only be inserted in $\alpha_l(S)$ by endpoints of e. We label the endpoints u and v. Node u has a

complete view of its neighbourhood. This is sufficient information to find incident circumcircles since its radius is set so that its diameter matches the communication range. It is important to note that only incident circumcircles are inserted into $\alpha_l(S)$: Node u is prohibited from making decisions on behalf of v or any other node in its neighbourhood. This idea is demonstrated in Figure 4.5 in which node u can only insert edges that it knows belong in $\alpha_l(S)$. Finally, the argument is symmetric in that if u inserts uv then v inserts vu. ∎

Finally, we note that the localised α-hull algorithm is resilient to non-uniform range so long as $r = \frac{1}{2}R_{min}$, where R_{min} is the minimum possible communication range for all nodes. Accurate boundaries emerge so long as R_{min} is sufficiently large to reflect a disc that is able to connect α-neighbours according to Definition 4.2.2.

4.5 Refinement

The selection of disc radius $r = 1/2R$ for communication range R guarantees correctness. Our initial investigation shows that this selection of disc radius exposes some unwanted detail, necessitating further refinement.

We present example misleading geometries in Figure 4.6. The quadrilaterals' edges represent communicating nodes and boundaries as determined by local α-shaping. The greyed discs used to establish the local α-shape have been included for clarity. We can see that sets of nodes located in close proximity may produce boundaries where, effectively, there is none. Unrefined, the local α-shaping process produces boundaries such as those seen in Figure 4.8. We provide two methods to further refine the α-shape should the default be too fine-grained.

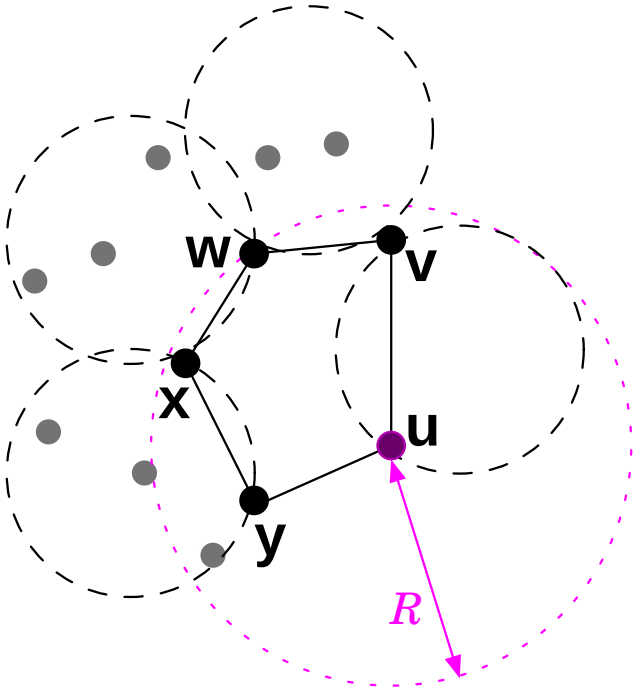


Figure 4.5: Node u may insert in $\alpha_l(S)$ only directed incident edges uv and uy.

4.5.1 Increasing the Disc Radius

The first method we investigate is to increase the radius of the disc in Step 3. The outcome is a shape that is more coarse. Under these circumstances the relationship between the α-shapes generated in a local and in a centralised fashion is unclear. The question remains open whether setting $r > 1/2R$, ie. setting the disc diameter strictly greater than communication range, connects local α-edges in a planar configuration amenable to a right-hand-rule mapping.

Increasing the available information to multi-hop neighbourhoods fails to resolve the issue at question. This idea is demonstrated using Figure 4.7. Referring first to Figure 4.7a, we present a configuration where the disc diameter is increased beyond the communication range R. Node v is out of the communication range of node u but, unbeknownst to u, inside the disc of radius r. This means that he the segment of the α-shape that would appear if computed centrally, shown in Figure 4.7b, fails

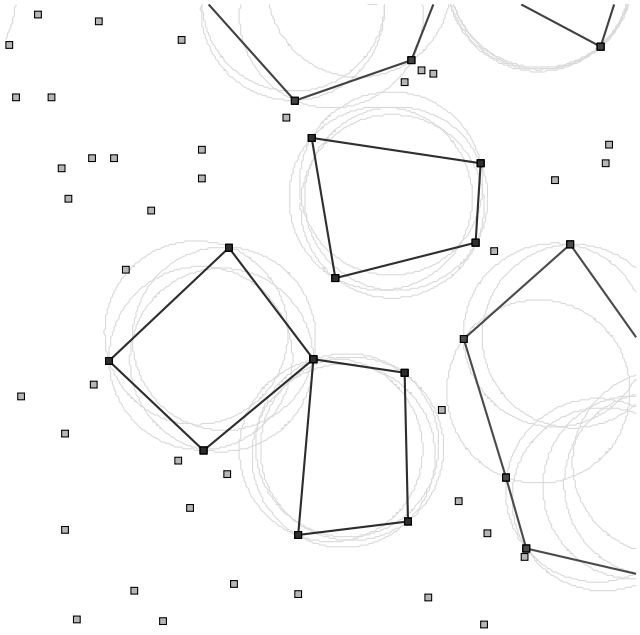


Figure 4.6: Nodes in close proximity may expose unwanted detail.

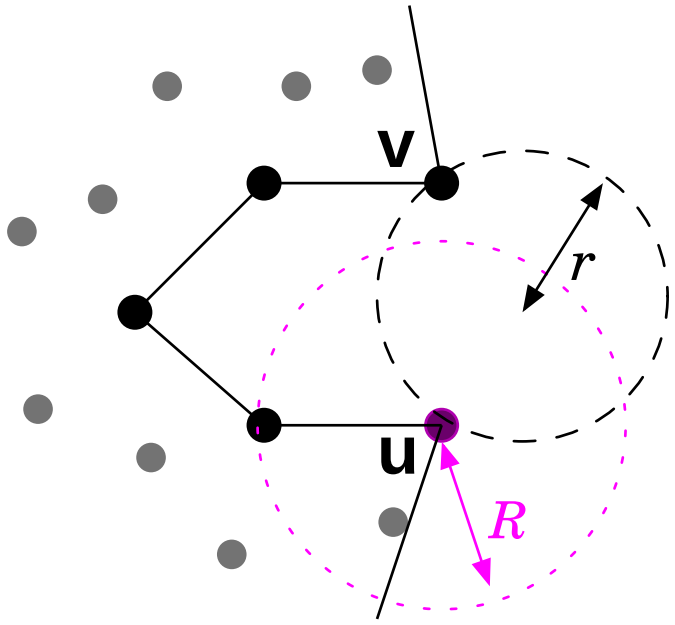
to appear when computed locally.

Hence, by increasing the disc radius an *alpha*-node can successfully determine whether it is located on the boundary; it is unclear if increasing the disc diameter beyond the communication range allows a boundary node to correctly communicate with all other nodes on the same boundary or how the local and global α-shapes might differ. In the following section we present a more effective refinement method.

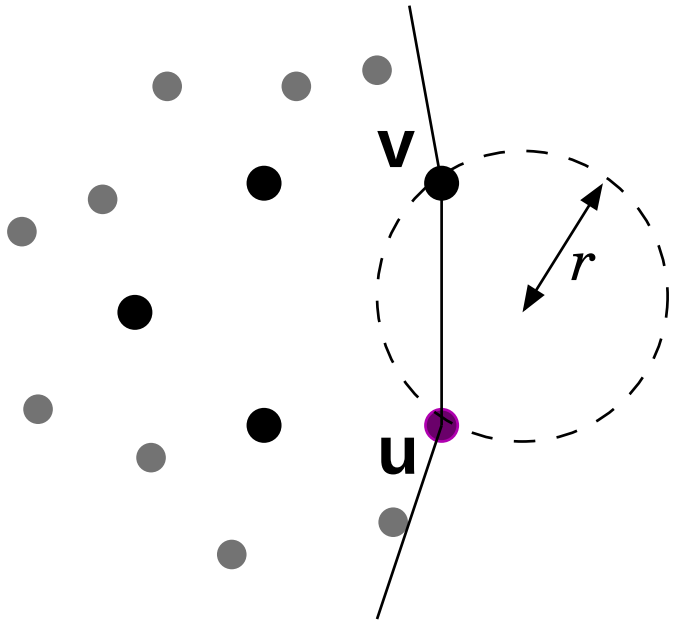
4.5.2 Refinement by Omission

Alternatively, network boundaries may be further refined following Step 4 by disregarding those boundaries found to be greater than some pre-determined number of hops in length. Our measurements show that in all but the sparsest networks tested, the length of most boundaries was found to be quite small, measuring 10 or fewer hops. This observation is reinforced by previous study([25]).

Ultimately it is the need to identify boundary nodes versus edges that dictates



(a) Local α-shape.



(b) Actual α-shape.

Figure 4.7: By setting $r > \frac{1}{2}R$, the local α-shape may differ from the actual α-shape of the network.

the refinement method. We present simulations in the next section to compare both methods, and evaluate the local α-shaping process in general.

4.6 Simulation Results

In the previous section we presented a complete algorithm to identify network boundaries without the need for broadcasting. Here we use a broad set of simulations to evaluate the algorithm performance with respect to density and distribution. We begin with a description of the networks tested.

4.6.1 Experimental Design

To evaluate the performance of the local α-shaping we simulate networks of varying density, distribution, and topology. Network nodes are distributed in a 200x200 unit space, each node with a fixed range of 8 units. We vary node density by changing the network size. Note that by changing size instead of communication range we can vary the density without affecting the maximum network diameter. Network sizes are 3500, 2500, and 1500 nodes. (In the uniform networks this results in average neighbourhood sizes of $\tilde{1}7$, 12, and 7 nodes.) To obtain results unbiased by isolated nodes we tabulate and experiment over the largest connected component of each network as described by Table 4.1.

Nodes locations are chosen from a normal or skewed (Pareto) distribution in addition to the uniform distribution traditionally used to generate wireless network topologies. Uniformly distributed networks may be sufficient to provide insight yet are poor representations of many real deployments. Normal coordinates are generated with an average of 100 (the center) and a standard deviation of 40. Skewed coordinates are chosen from the Pareto distribution with scale parameter 1.0 and shape parameter 100.5.

Table 4.1: Largest connected components in tested networks with 99% confidence intervals.

Initial Network Size	Size of lcc		
	Uniform	Normal	Skewed
3500	3499.9 ± 0.2	3450.8 ± 5.3	3403.8 ± 12.3
2500	2499.8 ± 0.4	2433.8 ± 4.9	2382.2 ± 10.9
1500	1490.0 ± 7.5	1406.7 ± 7.7	1359.0 ± 12.9

4.6.2 Refinement Phases Compared

We begin our evaluations by comparing local alpha-shaping with and without the refinement phases proposed in Section 4.5. The outcome of the unaltered localised α-shape algorithm is presented in Figure 4.8. Small 'empty' pockets appear in the densest networks, growing in number and size in the sparse networks. The less than satisfactory results in Figures 4.8b and 4.8c stem from the selected value of the α parameter. Recall that the localised algorithm reveals the same boundaries as the global algorithm so long as the disc diameter is restricted to the communication range. Despite this fact we explore the effect of an increase in α in the first refinement method.

The first attempt at refinement appears in Figure 4.9 using the same networks as in Figure 4.8. In these networks the α parameter has been increased by 10% so that $r = 1.10 * \frac{1}{2}r$. Few pockets appear in the denser networks shown in Figures 4.9a and 4.9b; unfortunately, the improvement in the sparse network of 1500 nodes shown in Figure 4.9c, while apparent, is marginal at best. Also, as discussed in Section 4.5.1, there is no guarantee that planar edges are found when the disc diameter is increased beyond communication range. Routing and mapping, then, becomes a challenge. Still, this approach provides the benefit of remaining entirely local to each node, needing no added communication.

In Figure 4.10 we show the boundaries revealed by omitting long paths as discussed in Section 4.5.2. Using this approach, nodes forward a discovery packet to their α-neighbours according to right-hand rule. In our simulations, packets that return to their origin having travelled less than 15 hops are omitted from contention. The improvement is clear. In all but the sparsest network, our algorithm reveals a single network boundary. And in the sparse network shown in Figure 4.10c only the largest empty regions remain.

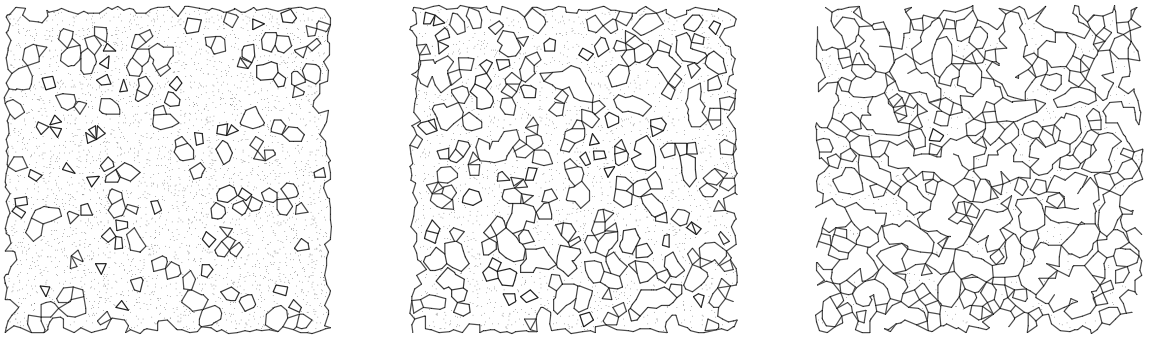
α-neighbours cooperate to omit unlikely boundaries and produce the most accurate results. The communication required is far less than is required for global broadcasts in comparable methods. We apply this refinement in remaining simulations.

4.6.3 Distribution and Density of Sensors

In this set of simulations we alter the distribution of nodes in addition to the network density. Distribution parameters are described in Section 4.6.1. Normally distributed sensor nodes, intended to better approximate aerial deployments, are shown in Figure 4.11; skewed sensor distributions, intended to better approximate ground projectile deployment, are shown in Figure 4.12. In all non-uniform simulations presented, α-shapes are shown refined using the mapping method described in Section 4.5.2. The α-shaping method performs well in all tested networks and, but for increasingly jagged boundaries, seems relatively unaffected by density.

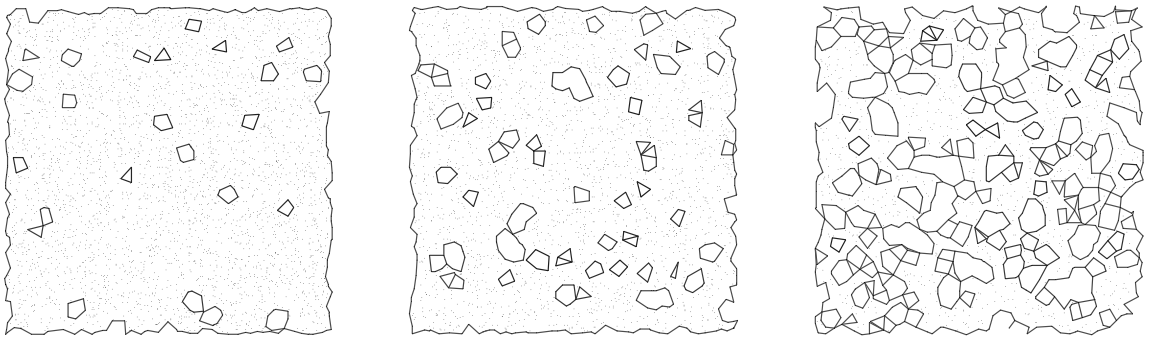
4.6.4 Additional Examples

Finally, we test local α-shaping using example networks featured in [119]. In that study the authors produced favourable results using topological means needing many network-wide communications. Neighbourhoods in their study ranged from 18 to 22 nodes in size. Our method produced only slightly improved boundaries, and can be



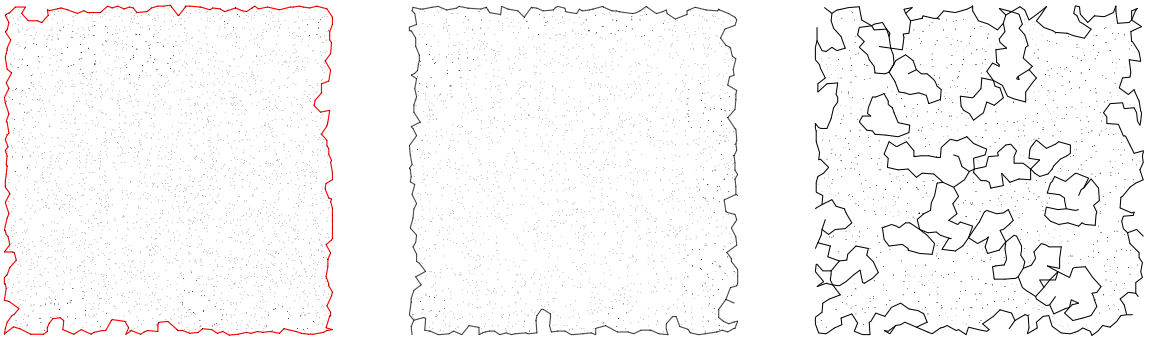
(a) 3500 nodes. (b) 2500 nodes. (c) 1500 nodes.

Figure 4.8: Unrefined boundaries determined using local α-shapes; network sizes reflect average neighbourhood sizes of 17, 12, and 7, respectively.



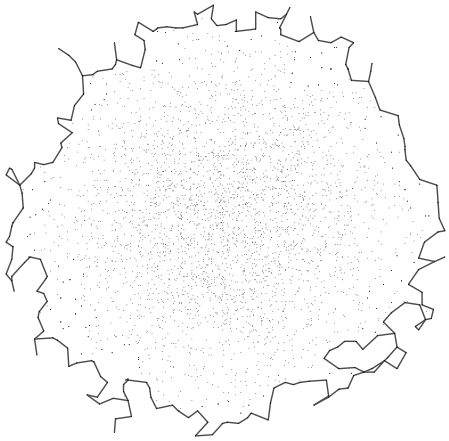
(a) 3500 nodes. (b) 2500 nodes. (c) 1500 nodes.

Figure 4.9: Disc radius increased by 10%; network sizes reflect average neighbourhood sizes of 17, 12, and 7, respectively.

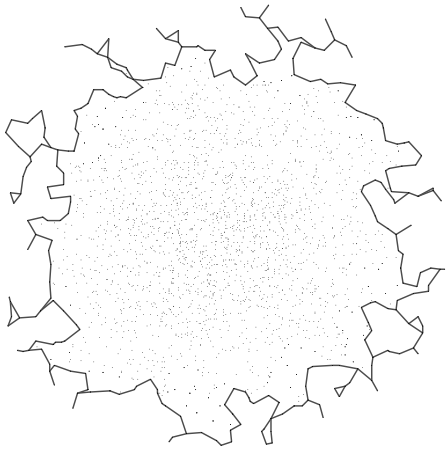


(a) 3500 nodes. (b) 2500 nodes. (c) 1500 nodes.

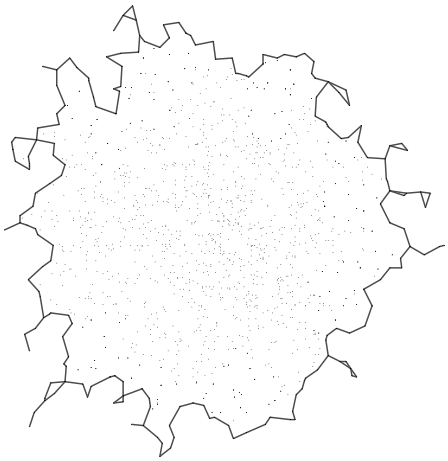
Figure 4.10: Boundaries are mapped and omitted if greater than 15 hops; network sizes reflect average neighbourhood sizes of 17, 12, and 7, respectively.



(a) 3500 nodes.

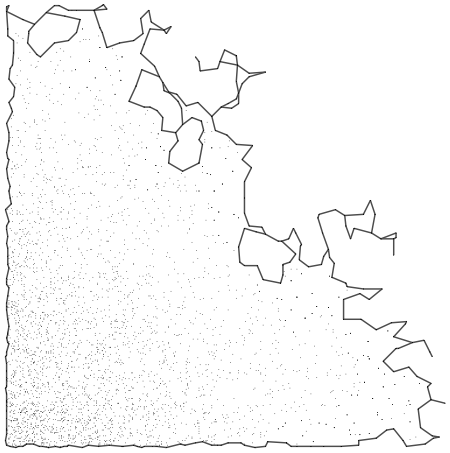


(b) 2500 nodes.

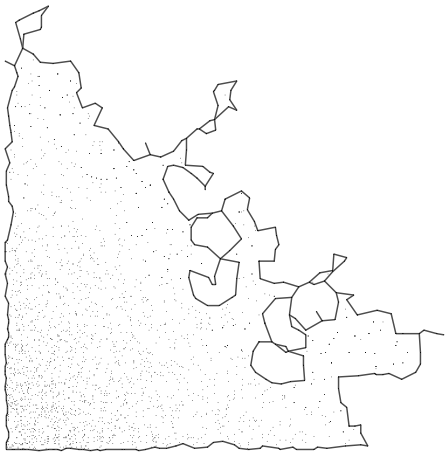


(c) 1500 nodes.

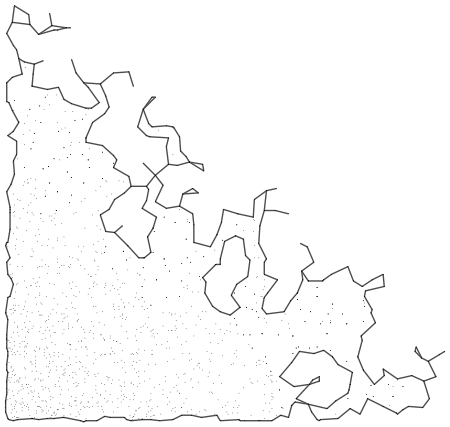
Figure 4.11: Results over networks with nodes distributed according to a normal distribution.



(a) 3500 nodes.



(b) 2500 nodes.



(c) 1500 nodes.

Figure 4.12: Results over networks with nodes distributed according to a skewed (Pareto) distribution.

seen in Figure 4.13. However, we emphasise that local α-hull yields slightly more accurate results using far fewer communications and neighbourhoods ranging from 7 to 10 nodes. This is less than half of the size of neighbourhoods in [23].

4.7 Chapter Summary

In this paper we have developed an algorithm to identify nodes and links along network boundaries. It is useful to place our work within the context of previous methods.

Edge detection algorithms and protocols have demonstrated considerable potential in the past. Such methods offer the benefit of relaxing the unit disc model, and needing no position information a priori. However, the success of previous methods has relied on the global cooperation of all sensor nodes in the network. The level of cooperation often requires packets to record information revealed over multiple broadcasts. This may be an unacceptable drain in such an energy constrained environment where channel contention and collision is at issue. Besides this fact a bootstrap node is sometimes assumed to exist, generally at the center or at the edge of the network. The origin of these nodes is unclear. This is restrictive behaviour: many networks, such as those quickly deployed in an emergency, may be unable to tolerate delays in network setup or proper placement of bootstrap nodes.

In contrast we assume that partial location information is available or obtainable. Specifically we investigate what might be achieved if relative information - positions of nodes relative to their neighbours - was known or computable. From this standpoint we can investigate geometric options from which we might otherwise be restricted. In this paper we have managed to detect network boundaries by provably localising the α-shape algorithm; the centralised version of which has been successful in capturing the shape of a set of points in many disciplines. The key is to set α so that the disc in use has a diameter equal to the communication range. Then each node may independently decide if it sits on the α-shape of the network and find the

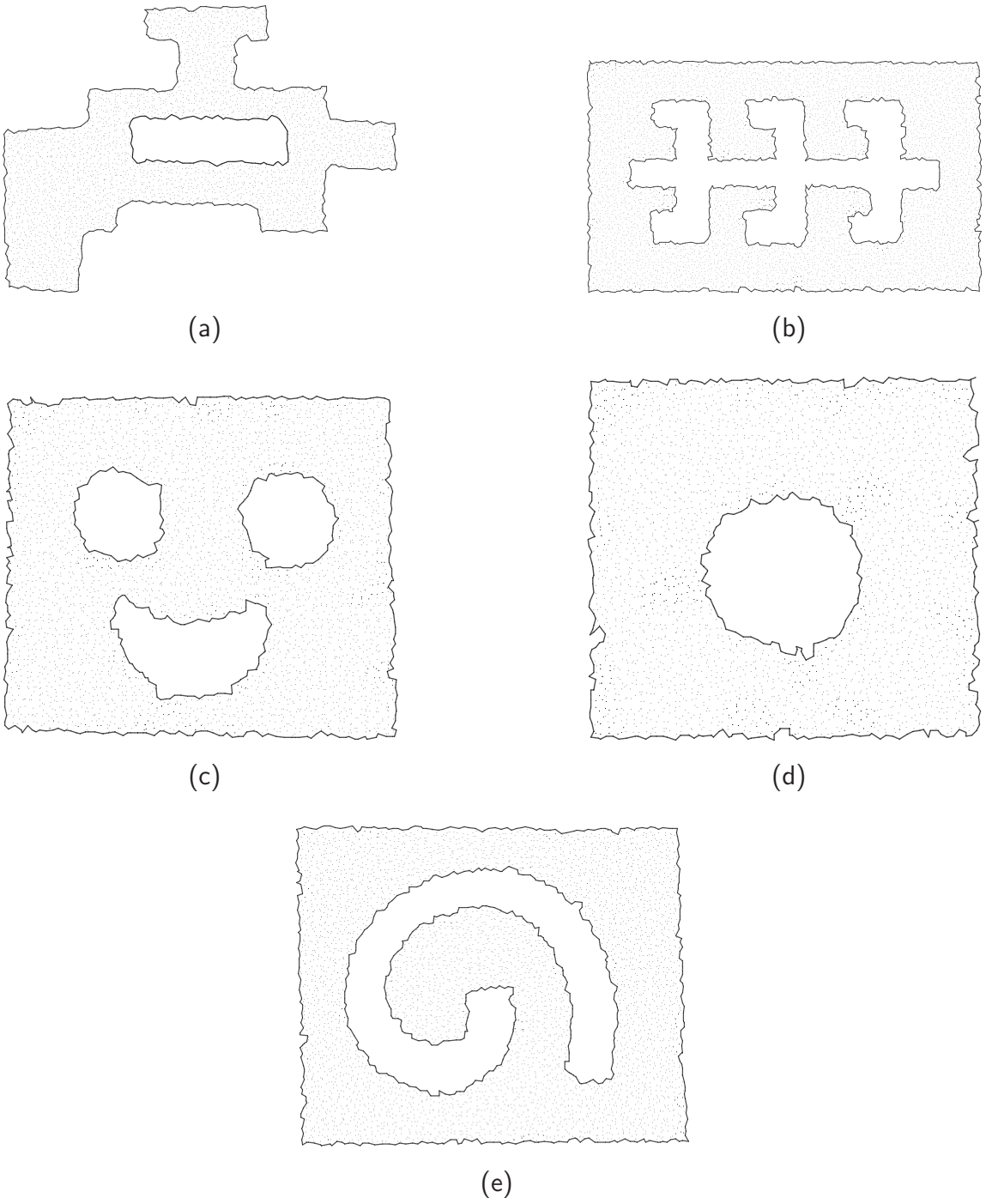


Figure 4.13: Dædal examples featured in [119]. They include (a) a building floor plan with 3420 nodes and average degree of 8; (b) a cubicle shaped office space with 6833 nodes and average degree of 7; (c) a happy face with 4050 nodes and average degree 8; (d) a network with 3443 nodes and average degree of 8; (e) a spiral shape with 5040 nodes and average degree of 10.

correct incident edges. By using our algorithm and selected α the collection of locally computed nodes and edges provably matches the set of nodes and edges of the α-shape computed centrally.

To validate our approach we tested the local α-shape algorithm in simulated networks of varying density where nodes were distributed according to uniform, normal, and skewed distributions. We found the algorithm output could be further refined if nodes mapped their boundaries by forwarding a discovery packet according to right-hand-rule; boundaries exceeding some length could be omitted from consideration.

Compared to results from previous studies the local α-shape algorithm performs favourably. Presently, we are looking at statistical methods to resolve inaccuracies associated with position estimation error.

Chapter 5

Guaranteed Geographic Routing with Intersections

5.1 Introduction

The construction of network subgraphs appropriate for position-based (or geographic) routing protocols has, to date, remained a complex problem. These subgraphs are needed to recover from the local minima problem (see [15]) that prevents delivery and plagues position-based protocols. Network subgraphs constructed for recovery using only 1-hop information risk inaccuracies that cause routing failures([60, 107]). If permitted to cooperate, nodes may construct a network subgraph that remedies any inaccuracies([59, 107]). Yet the energy needed to power many rounds of communication risks being prohibitive in such a resource-constrained environment. The ideal wireless network subgraph would guarantee successful delivery while a) needing only 1-hop information and b) be able to acquire such information passively.

Traditionally, position-based routing protocols construct subgraphs (herein referred to as just 'graph') from available links in somewhat of a bottom-up fashion. Generally the idea is to extract a specific type of graph. During the setup of the

graph each node evaluates potential links to find those that preserve some global properties. Planar graphs([30]) and k-spanners([106]) are two such examples. The analogous question would be to ask, "what is the minimum set of edges that must be preserved to guarantee so-and-so feature in the graph?"

Our work is motivated by the opposite question, "What is the minimum set of edges that must be *deleted* while still providing guarantees?" Without sacrificing the scalability and success of position-based routing, the goal of this work is to disturb the network as little as possible. To this end it is necessary to understand the causes for a position-based routing protocol to fail to recover from local minima and deal with those causes, directly. We believe the work in this chapter is the first work in that direction.

In this chapter we investigate routing according to left- or right-hand rule (LHR). Using LHR, a node upon receipt of a message will forward to the neighbour that sits next in counter-clockwise order in the network graph. (Alternatively, clockwise order if using right-hand rule.) When used to recover from greedy routing failures, LHR guarantees success if implemented over planar graphs; for this reason it is often called 'face-routing'. We note, however, that if planarity is violated then LHR is only guaranteed to eventually return to the point of origin. Our work seeks to understand and correct the underlying causes.

We have chosen LHR for three reasons. First, it is most prevalent in position-based routing literature and hence well-studied. Second, it is a simple rule. Finally, the ideal network graph remains elusive. To re-iterate, we envision the ideal graph as overcoming the inaccuracies that lead to routing failures; as one that results from knowledge of the 1-hop neighbourhood; as one where each node transmits a constant number of messages.

We begin by with a provable enumeration of the possible types of intersections in a unit-disc graph, within which any two nodes are neighbours if separated by a maximum distance of one unit. We show that only three types of intersections are

possible and that in only one of these configurations does LHR fail to recover.

5.2 Links that Prohibit Routing Success

In the previous section we noted that routing according to right- or left-hand rule (LHR), alone, fails to provide a guarantee of success. Though this fact is well known, the reasons and circumstances under which delivery may fail are poorly understood. In this section we seek to investigate the limits imposed by intersections on face-based recovery.

Our goal is to maximise the number of active edges in a wireless network graph while providing routing guarantees using LHR. In an attempt to relax planarity, currently required to guarantee the success of a traversal between two nodes, we must identify the causes for failure in an arbitrary graph. We focus this work on the unit disc graph (UDG), where all communication ranges are normalised. The UDG is appropriate since it limits potential routing options yet still poses a challenge to LHR routing. Our investigation begins with an enumeration of all of the types of intersections that may appear in the UDG.

5.2.1 An Enumeration of Intersection Types

Consider any two intersecting edges. We provide the edges ac and bd in Figure 5.1 for reference. The nodes a, b, c, d at the end points of these edges form a 4-gon (shown in Figure 5.1 using dashed lines). The question we ask is, which of the edges of the 4-gon may or may not be communicating links in the unit-disc graph? In order for at least one such edge to exist, we need to show that *all four sides cannot be greater than both diagonals.*

Using cosine rule we know,

$$(ac)^2 = (ad)^2 + (dc)^2 - 2(ad)(dc)\cos D. \tag{5.2.1}$$

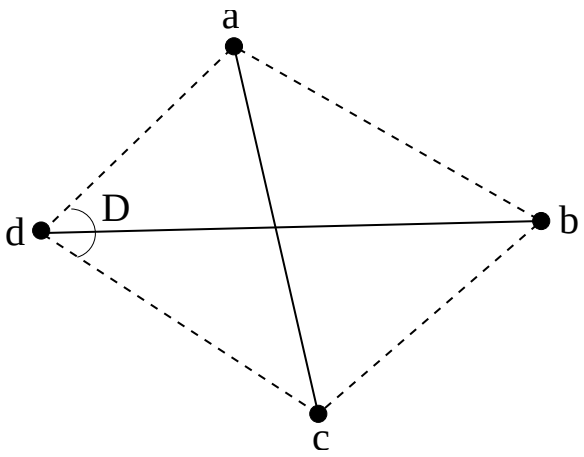


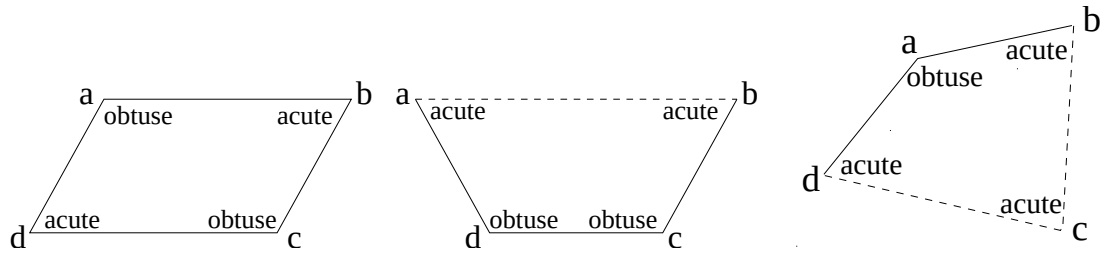
Figure 5.1: Intersecting links between two pairs of nodes may impose any or all edges in a 4-gon.

If $|ac|$ is less than or equal to 1, then

$$(ad)^2 + (dc)^2 - (ad)(dc)\cos D \leq 1. \tag{5.2.2}$$

When $D \geq \frac{\pi}{2}$, then $\cos D \leq 0$. In this case, $(ad)^2 + (dc)^2 \leq 1$, which means $(ad) \leq 1$ and $(ad) \leq 1$. Thus, if an angle of the 4gon is right or obtuse, then both incident edges must exist in the UDG. (By contrast, incident edges when $D < \frac{\pi}{2}$ may or may not exist.)

This implies and restricts the possible configurations that allow intersections to three, all shown in Figure 5.2. The two cases where the nodes of intersecting edges produce a 4-gon with two obtuse angles is shown in Figures 5.2aand 5.2b, while the 4-gon containing a single obtuse angle is shown in Figure 5.2c. (Note that it is impossible for a 4-gon to be constructed with three obtuse edges; and that edges incident to an acute angle may or may not appear in the unit-disc graph.)



(a) All edges of the 4-gon must exist.

(b) Edges bc, cd, da must exist.

(c) Edges ab, ad must exist.

Figure 5.2: Possible 4-gons when edges ac and bd intersect in the UDG with dashed lines indicating edges that may or may not appear.

5.2.2 The Prohibitive Link

A trace over each configuration reveals that LHR routing recovers from all but one configuation. The 'bad' configuration occurs during a traversal of the intersection in Figure 5.2c where there are three acute angles and when only the edges of the 4-gon incident to the obtuse angle appear. This represents the case where network node a communicates with b, c, d, and b with d; node c communicates only with a. We call this case the *umbrella* configuration for its appearance, as shown in Figure 5.3.

Referring to Figure 5.3 there are two ways in which LHR may fail. The first is demonstrated by the dashed-dot-dash line originating at node d. (Entry at nodes a and b are analogous.) A traversal using left- or right-hand rule will never traverse edge ac while travelling through this intersection. Supposing c must be traversed in order to reach the destination, LHR will fail. The second possible failure occurs when a LHR traversal encounters the umbrella intersection first via node c in Figure 5.3 using the dashed line. LHR traverses the inside of the triangle $\triangle abd$ and exits without ever seeing edges that protrude from the outside of the triangle. As before, any such edges leading to the destination may be overlooked by an LHR traversal.

The cause of both failures lies in the relationship between $\triangle abd$ and ac in Fig-

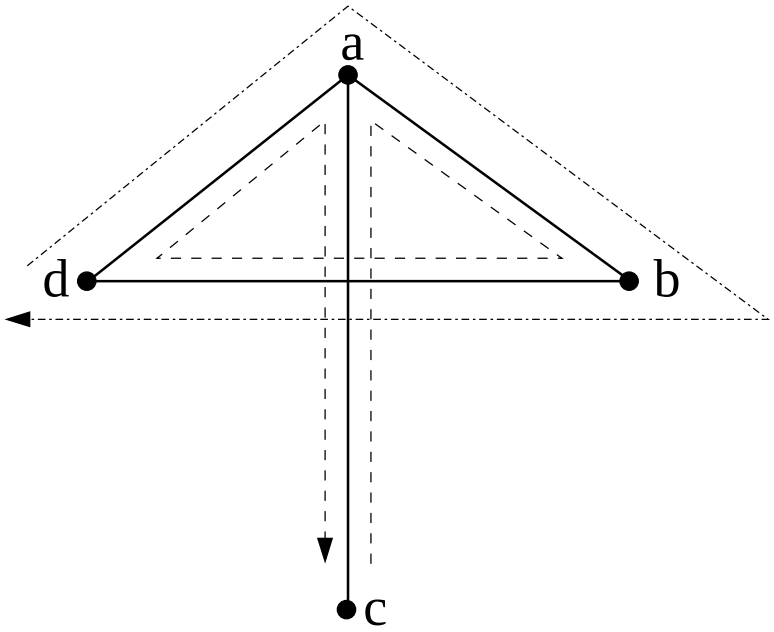


Figure 5.3: The 'bad' umbrella configuration prevents the success of LHR two ways.

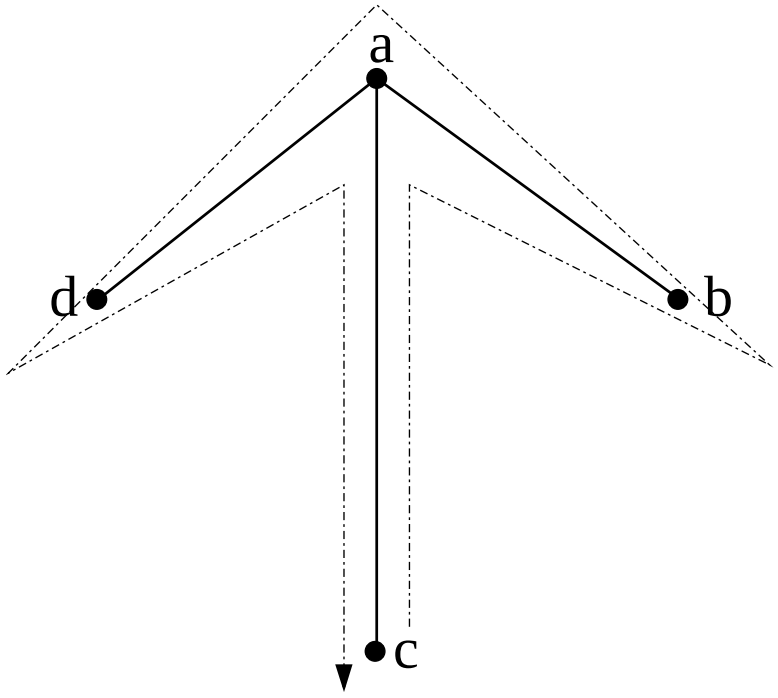


Figure 5.4: Removing prohibitive link *bd* allows LHR to traverse all edges.

ure 5.3: There exists an edge from the triangle that is accessible only from inside the triangle. In other words, a traversal around the inside or the outside of the triangle fails to encounter all edges leading to the triangle. Both failures are solved by removing any of the edges that form the triangle. The easiest of these to identify and remove from the network graph is the edge of the triangle that forms the intersection in the umbrella configuration. We call this the *prohibitive link*.

We revisit this subject and build a networking protocol in the next section. Before closing this section the outcome following a removal of the prohibitive link from the umbrella configuration is demonstrated in Figure 5.4. The configuration that was an intersection is reduced to a planar set of edges easily navigated by left- or right-hand rule.

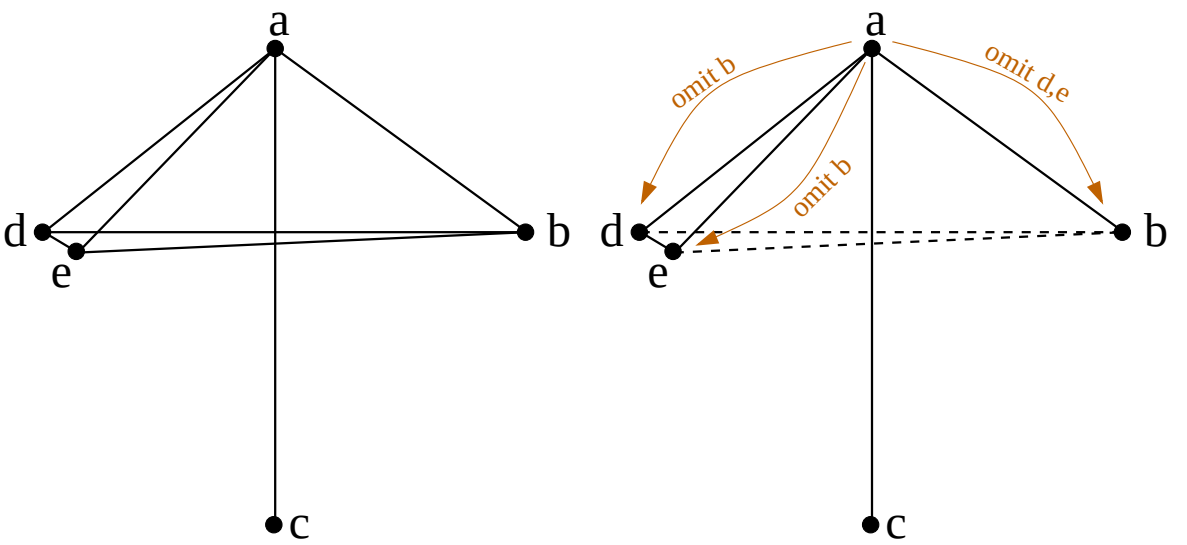
5.3 Prohibitive-link Detection and Routing Protocol (PDRP)

We have enumerated all possible intersections in the unit-disc graph and identified the type of intersection with the link that prohibits successful delivery when routing according to right- or left-hand rule. In this section we present a prohibitive-link detection and routing protocol (PDRP) and show its correctness.

5.3.1 PDRP Overview

Most wireless protocols construct network graphs to guarantee delivery. The goal of PDRP is to remove from the network graph as few links as needed to guarantee routing. To better describe PDRP we assume a static graph where each node is assigned a coordinate in a 2-dimensional Euclidean system. We assume that the graph is connected and that all links are bi-directional. PDRP functions adequately in a mobile space provided that changes in position occur over a greater time-frame

than is required to re-evaluate local prohibitive links and transmit local updates. In this work communication range is fixed and uniform across all nodes; a relaxation of this requirement is the subject of future work.



(a) Before detection phase.

(b) Following detection phase links db and eb omitted.

Figure 5.5: Local neighbourhood from viewpoint of node c, before and after the PDRP detection phase.

PDRP consists of two phases, detection and routing. The routing component operates similarly to the routing component in many established position-based routing protocols: PDRP routing consists of a greedy phase and a recovery phase. During normal operation nodes route in a greedy fashion and forward messages to the neighbour that most reduces the distance to the destination. Where no such neighbour exists a message is deemed 'stuck' in a local minima and is forwarded according to left- (or right-)hand rule. (The node initially selected is the first to appear left, or right, of the line segment from the current location to the destination.) The first node found that sits closer to the destination than the 'stuck' location returns to the greedy

forwarding phase. The correctness of PDRP and its ability to guarantee delivery to the destination is discussed in Section 5.3.2.

During the PDRP detection phase each node inspects its neighbourhood. Using reported neighbour positions each node evaluates intersections within range and flags any three neighbours that compose an umbrella configuration, as described in Section 5.2.2. Once sufficient information is compiled a node sends a notification packet to the neighbours that anchor prohibitive links. Within this packet are a list of prohibitive links to be avoided during the recovery phase.

The detection phase is demonstrated in Figure 5.5. In Figure 5.5a node a determines that two intersections in its vicinity contain prohibitive links, those links being bd and eb. Nodes b, d, and e have no knowledge of node c's existence. The responsibility falls on node a to inform neighbours of their prohibitive links. Moving to Figure 5.5b node a instructs each of d and e to ignore their links to b during recovery; similarly node a instructs b to omit links to d and e.

By definition, the number of messages required to 'destruct' the network graph are fewer in number than current efforts such as the mutual witness protocol in GPSR. Alternatively notifications may be avoided entirely by either i) passing and evaluating over 2-hop neighbourhoods or ii) compactly embedding (in a fixed space) prohibitive links within recovery packets. We intend to solve this in the future by use of Bloom filters to compress prohibitive link information into recovery packets. In the next section we prove that PDRP functions correctly.

5.3.2 Statement of Correctness

Having identified removed prohibitive links in umbrella configurations, we show in this section that PDRP will successfully route a message between two nodes if a path exists. We remind our reader that during the routing phase of PDRP, any standard position-based routing technique consisting of greedy + face-routing recovery may be

implemented.

The following argument progresses first by defining the graph embedding so that we might state our claim. We then show correctness by tracing a face-routing traversal within intersections of the embedding defined.

Definition 5.3.1 *Let G be an embedding of a graph. We define $G_{unu} = UDG(G) \cap NUmI(G)$, where $UDG(G)$ is the unit-disc graph over G and $NUmI(G)$ is the sub-graph of G where umbrella intersections are removed.*

Proposition 5.3.2 *We claim that in G_{unu} a traversal, T, consisting of left-hand rule with memory, will find and traverse a unique face.*

Proof. We prove by induction on the neighbourhoods witnessed by T. Consider the first neighbourhood, k_0, visible to starting node v. If no intersection is visible to v then the next edge in T is trivial. If, however, an intersection exists in k_0 then it must be in the form depicted in either of Figures 5.6a or 5.6b (see Section 5.2.1 for proof).

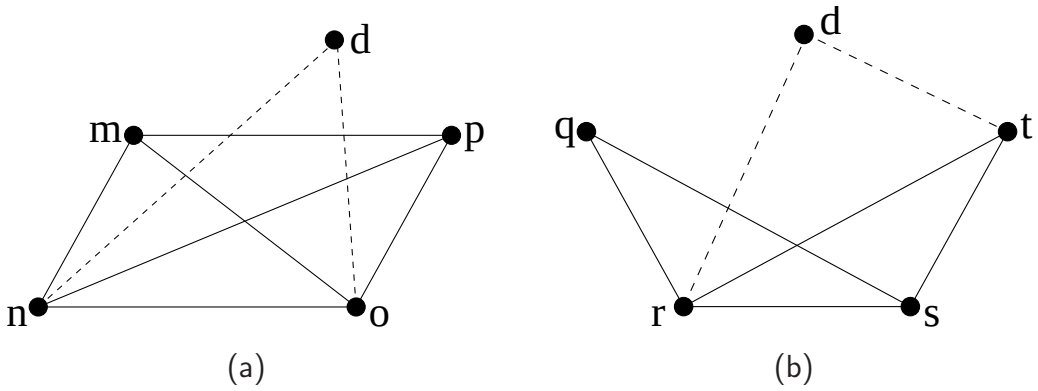


Figure 5.6: Once prohibitive links are omitted, two possible contentious configurations remain.

Case 1. Consider the intersection in Figure 5.6a. For any $v \in \{m, n, o, p\}$ and destination d, if $\overline{vd}$ intersects with no local edges (ie. $\overline{vd}$ does not pass through

quadrilateral ($mnop$)) then the next left edge - and thus first edge in the current face - is trivial. If, however, $\overline{vd}$ does pass through ($mnop$) as shown in Figure 5.6a then there are two cases:

$v = n$ The starting vertex is situated in the quadrilateral such that a single vertex sits left of $\overrightarrow{vd}$ and two vertices sit on the right. In Figure 5.6a this case is represented by $v = n$. Node n forwards to m. Both mo and mp intersect with nd, the line segment from the destination to the point where T started, so T will escape this neighbourhood when m chooses the next ccw edge from $\overrightarrow{mp}$.

$v = o$ The starting vertex is situated in the quadrilateral such that a single vertex sits right of $\overrightarrow{vd}$ and two vertices sit on the left. In Figure 5.6a this case is represented by $v = o$. Here, too, o forwards to m and m chooses the next ccw edge from $\overline{mp}$.

In either case, the face of interest begins at vertex m where a cycle, if traversed, will be declared.

Case 2. Consider the intersection in Figure 5.6b. Let starting node be $v \in q, r, s, t$ and destination d sit such that $\overline{rd}$ intersects $\overline{qs}$ and $\overline{sd}$ intersects $\overline{rt}$. The trivial case is $v = q$. Three cases remain:

$v = r$ The starting vertex is situated in the quadrilateral such that a single vertex sits left of $\overrightarrow{vd}$ and two vertices sit on the right. In Figure 5.6b this case is represented by $v = r$. r forwards to q where T will escape the neighbourhood. (Recall that when T intersects with $\overline{vd}$, T switches faces.) In this case the cycle will be detected at r.

$v = s$ The starting vertex is situated in the quadrilateral such that a single vertex sits right of $\overrightarrow{vd}$ and two vertices sit on the left. In Figure 5.6b this case is represented by $v = s$. s forwards along $\overline{sq}$ where T escapes the

neighbourhood. On its return, T will detect a cycle at s since node t will avoid the edge $\overline{sq}$ since it was traversed previously.

$v = t$ The starting vertex is situated in the quadrilateral such that all three vertices sit left of $\overrightarrow{vd}$. In Figure 5.6b this case is represented by $v = t$. Here, T traverses $\overline{tr}, \overline{rq}$ before its escape from q. (Note that $\overline{qs}$ is not a valid edge since it intersects the edge previously traversed, $\overline{tr}$. T will detect this cycle at node t.

Assume now that for any neighbourhood, k_i, traversal T exits on the same face on which it enters. We show that for neighbourhood k_{i+1} traversal T exits on the same unique face on which it enters.

Referring once more to Figure 5.6, there are two types of neighbourhoods to consider. Those intersections whose endpoints join into a quadrilateral such as in Figure 5.6a require little consideration. For any entry point m, n, o, p on the quadrilateral, T will exit on the outside of this neighbourhood.

Similarly in Figure 5.6b, traversals entering on $\{q, r, s\}$ are trivial. We focus on traversals of T that reach node t. From t the next ccw edges in T are $\{\overline{tr}, \overline{rq}\}$ since $\overline{qs}$ intersects $\overline{tr}$. From q, T is forwarded along the next ccw edge. ∎

Corollary 5.3.3 *For any G_{unu}, a traversal, T, consisting of left-hand rule with memory, guarantees a path will be found provided a path exists, or complete the face if no path exists.*

*Proof.*We know that for a set of unique faces (ie. no intersections) in an embedding, that a left-hand traversal from source to destination is guaranteed to find a path provided one exists. Thus T, which finds sets of unique (non-intersecting) faces will find a path if it exists, or complete the face where no path exists. ∎

Finally, we note that traversal T requires no memory to succeed. A trace with no memory through all examples reveals that T will escape from any intersecting neighbourhood via the same egress links, albeit by traversing a few extra links.

PDRP overcomes drawbacks of existing methods. We show in the next section that an implementation of PDRP is also competitive with these methods.

5.4 Simulation Results

The previous section describes the PDRP protocol and asserts its correctness. We demonstrate the practical performance of PDRP via simulation in the sections that follow.

5.4.1 Experimental Design

We have implemented PDRP into the CLDP suite of position-based routing protocols [58] available for TinyOS [46, 116]. TinyOS is an event-driven operating system deployed on many commercial sensor networking products. Code written for TinyOS may be executed directly within TOSSIM [78], a simulator designed for debugging and evaluation of protocols before installation to sensor devices. The CLDP suite is a natural fit: In addition to CLDP it implements GPSR and face-routing with a variety of options; its implementation helps to ready PDRP for testing and improvement in real-world environments.

In our evaluation we compare PDRP versus CLDP and GPSR-MW. Recall from Section 2.3.1 that GPSR constructs a planar graph from available links. CLDP achieves planarity by destructing the network graph by probing links to identify and remove intersections. Likewise, PDRP destructs the network graph yet requires no probing and seeks to minimally disturb the graph.

GPSR simulations use the Gabriel Graph option for planarization. GPSR design and accomplishments serve as the foundation on which much more complex efforts have been built; it has long been considered the baseline for benchmark performance. In our simulations we use GPSR-MW which includes the mutual witness (MW) protocol for additional robustness. CLDP was developed much more recently to avoid the pitfalls that occur when the unit-disc graph assumptions are violated in real-world networks. PDRP, GPSR, and CLDP are similar in construction and operation yet differ in their design goals. A comparison in performance between these protocols is most appropriate.

Simulations are composed of 200 nodes placed uniformly at random within a square two-dimensional space. The communication range for all nodes is fixed at 180 units. Node density varies as we scale the network space by 100 units in the range from 1300 to 2000 units squared. Each simulation tests protocols over 50 runs using the same 50 topologies generated from non-overlapping random streams. This is sufficient to guarantee a 95% confidence interval. Our two primary performance measures are success rate and average path stretch. We additionally investigate the messaging overhead during setup. Each measure is defined and discussed within their relevant sections below.

5.4.2 Routing Setup and Success

The success rate is measured as the fraction of messages that are correctly transmitted between each pair of source-destination nodes. Figure 5.7 shows the success rate of each protocol as a function of network density described by average number of neighbours.

All protocols perform well though consistency appears to be highest among PDRP simulations. Note that our simulation results of CLDP differ slightly from results reported in [59]. In addition both CLDP and PDRP fall short of the expected

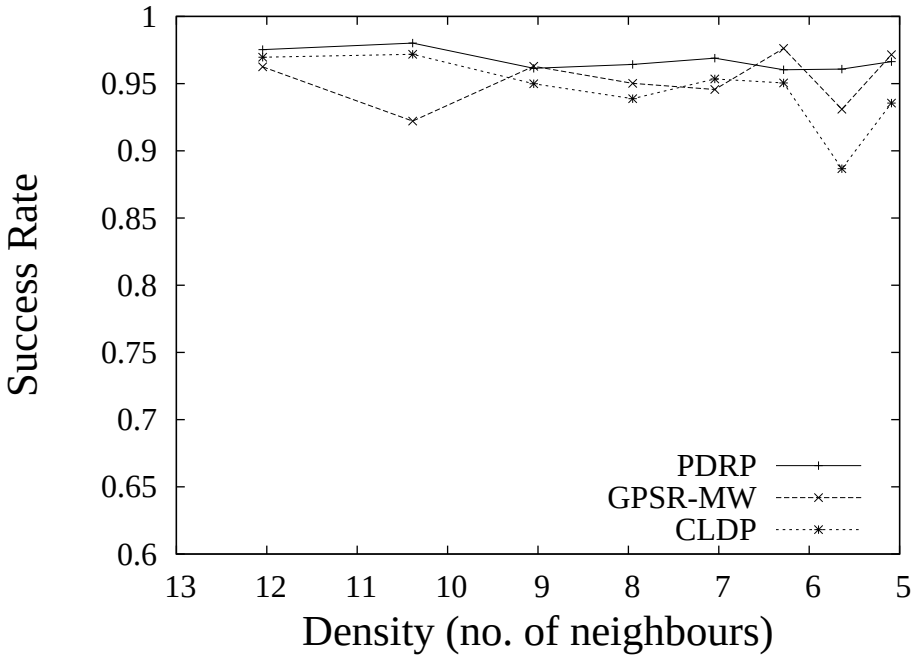


Figure 5.7: Routing Success.

100% delivery rate that is predicted by their proofs of correctness. This is attributable to two phenomena. The first is collinearity. Left uncorrected, collinear links have the potential to confuse left- or right-hand rule in a manner that removes routing guarantees. In Figure 5.7 collinearity is directly responsible for the dip that occurs in the GPSR-MW plot at $\tilde{1}0.5$ neighbours, as well as at $\tilde{6}$ neighbours in GPSR-MW and CLDP.

Referring again to Figure 5.7, PDRP seems less affected by collinear links. Further investigation reveals this to be an unexpected side-effect of the design goal underlying PDRP: to preserve the original network graph in a manner that allows delivery using position-based routing. In preserving as many links as possible PDRP maintains links that are omitted from the GPSR and CLDP network subgraphs. Hence there appears fewer collinear links since a greater number of links are seen by left- or right-hand rule in angular order in the PDRP subgraph.

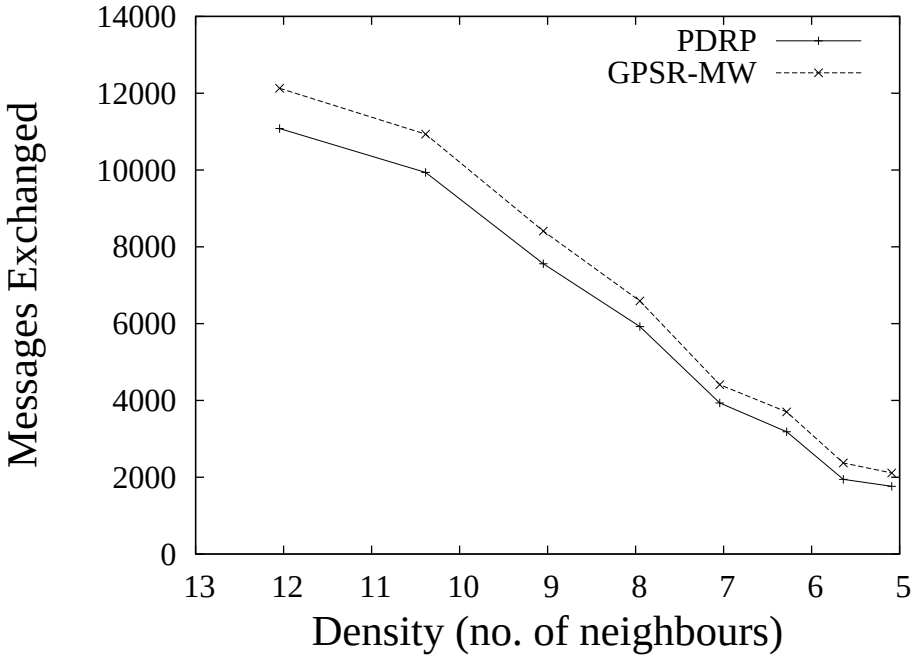


Figure 5.8: The number of messages exchanged during setup.

The second reason we see a slightly less than 100% success rate is in the TinyOS implementation. There appears to be slight bugs about which we are in contact with the authors of the position-based routing suite for TinyOS.

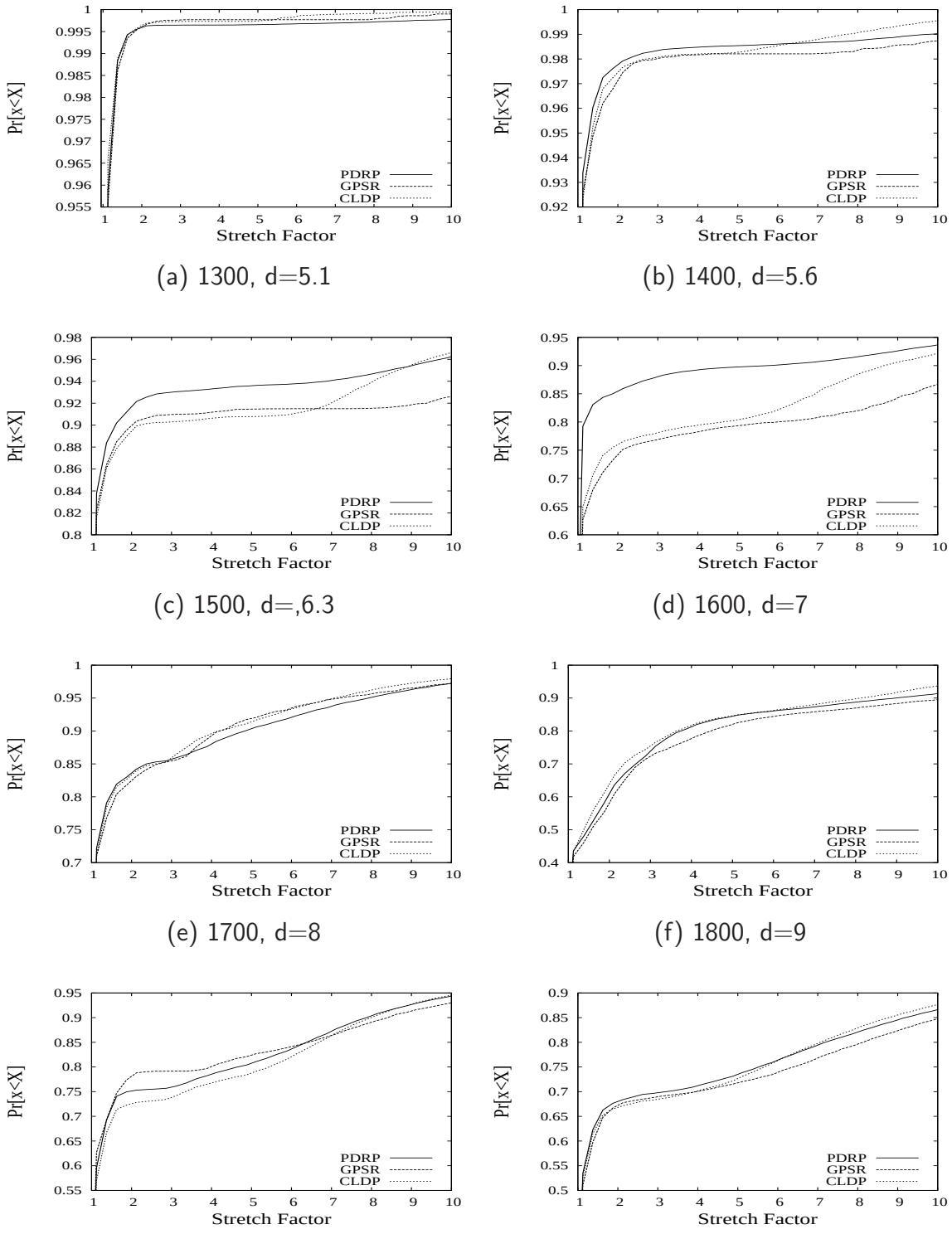
An additional evaluation appears in Figure 5.8 which shows the number of messages that are exchanged during the setup phase of both PDRP and GPSR with the mutual witness protocol. (CLDP data has been omitted since the number of messages exchanged is many times higher than PDRP and GPSR.) In the current implementation both protocols search for relevant network features upon receipt of each new 'HELLO' message. Updates are sent whenever there is a change in the local topology that affects routing. While there is room for optimisation, we can see that the messaging complexity of PDRP is on the order of GPSR-MW. In the future we intend to reduce this complexity further by compactly embedding prohibitive link information in the packet headers.

5.4.3 Routing Quality

We measure routing quality using path stretch factor. The stretch factor describes the ratio between the number of hops in the shortest path and the number of hops traversed using the routing scheme in question. The cumulative distribution functions generated by stretch factors appear in Figure 5.9. Subplots show the cdf as revealed over networks of increasing density. We emphasize to our reader the differences in scale (over the y-axes) between plots so that appropriate levels of detail may be viewed for each set of simulations.

The difference in performance between GPSR, CLDP, and PDRP in the sparsest networks, shown in Figures 5.9a and 5.9b, is near indistinguishable. The reduced likelihood of intersecting links that occurs with sparse networks means that all three protocols are more likely to choose the same path. Furthermore node sparsity reduces the number of potentially good routing choices so each protocol is more likely to route along the shortest path. Referring now to Figures 5.9e through 5.9h we can make two observations. The first observation is that routing quality amongst all three protocols continues to to behave in a manner that is (statistically) indistinguishable. Second, we can see a clear degradation in path quality that is directly attributed to the increase in frequency of intersections and their potential to separate the routing path from the shortest path during recovery modes.

There does appear, however, a gap of particular interest that separates PDRP from GPSR and CLDP. It is visible in Figure 5.9d which depicts the cdf of path stretch factor where the number of neighbours per node is approximately 7. Further investigation reveals that the improved routing quality provided by PDRP stems from its propensity to leave intersecting links intact during the recovery phase: PDRP often routes along direct links between neighbours whereas the GPSR-family of protocols will route along multiple links between the same neighbours. Why this separation is much more clear when neighbourhoods are approximately 7 nodes in number rather



(a) 1300, d=5.1

(b) 1400, d=5.6

(c) 1500, d=,6.3

(d) 1600, d=7

(e) 1700, d=8

(f) 1800, d=9

(g) 1900, d=10.4

(h) 2000, d=12

Figure 5.9: Edge proximity distributions reveal the proximity to the network edge of nodes that declare boundary status.

than the more dense or sparse nodes remains an open question and requires further investigation.

5.5 Chapter Summary

In this chapter we have explored an new approach to graph construction for successful forwarding in position-based routing. It is instructive to compare this approach with previous work.

Like the face-routing family of protocols, the success of PDRP relies upon a recovery phase that implements a form of left-hand rule. Traditionally, the success of face-routing schemes relies on the assumption that the underlying graph is planar. The planar graph was chosen because, using left-hand rule, a path to the destination will always be found (if a path exists). This is restrictive; local constructions of planar graphs risk inaccuracies, while co-operative (or global) constructions are resource intensive. In either case there has yet to appear an examination of the challenges that face left-hand rule in the presence of intersections.

By contrast, the approach taken in this work was to enumerate the configurations that form an intersection in the network graph. We then scrutinised each with a left-hand rule traversal so as to isolate the 'bad' configurations from which left-hand rule is unable to recover. In doing so we recognised the existence of a prohibitive link that has the potential to conceal other viable links from a left-hand rule traversal. We then presented PDRP, a protocol that detects and avoids the prohibitive link to successfully deliver packets. It operates locally and, unlike planarization methods, omits only nonessential links.

Our simulation results demonstrate that PDRP performance is similar to, and in some cases out-performs, current face-routing schemes. Compared against CLDP and GPSR with the mutual witness protocol, the success rate of PDRP appeared more consistent across networks of varying density. The availability of additional

edges to left-hand rule during the recovery phase allowed PDRP to avoid the trends common to both GPSR and CLDP. Moreover, in all cases the stretch factor of paths generated by PDRP was competitive with both GPSR and CLDP. In networks where neighbouring nodes averaged 7, PDRP shows noticeable improvement over GPSR and CLDP; the underlying cause requires further analysis.

We have implemented PDRP into the geographic routing suite for TinyOS and are pleased to make it available upon request. In the future we hope to remove the unit-disc assumption. Then, using the approach presented in this chapter we expect to augment PDRP for general case networks where communication error and non-uniform range is commonplace.

Chapter 6

Conclusion

6.1 Thesis Summary

In this thesis we have adapted and designed simple geometric constructs for the boundary detection and routing problems in large wireless networks. We have shown that analyses of the geometric properties of the underlying network graphs may lead to solutions that are better tailored to their applications: we have presented local convex view, a locally generated alpha-hull, and a position-based routing scheme whose goal is to preserve available links with low communication complexity.

The local convex view (*lcv*) is a heuristic algorithm that operates locally. A node decides it is close to the network edge if the node finds that it lies on the convex hull of its 1-hop neighbourhood. We found that necessary information may be unavailable unless a neighbourhood is 2-connected. An analysis of the possible geometries in such cases allowed us to propose a simple probabilistic model to decide the convex hull of such neighbourhoods. *lcv* performance was found to be consistent with competing methods and resilient to their qualitative drawbacks. Moreover, simulations failed to reinforce the assumption that *lcv* would be adversely affected by position estimation error. Further examination of local geometries revealed three possible neighbourhood

configurations caused by position estimation error. *lcv* was shown to be immune to two of them.

We then approach boundary detection by provably localising the α-shape algorithm; the centralised version of which has been successful in capturing the shape of a set of points in many disciplines. The key is to set α so that the disc in use has a diameter equal to the communication range. Then each node may independently decide if it sits on the α-shape of the network and find the correct incident edges. By using our algorithm and selected α the collection of locally computed nodes and edges provably matches the set of nodes and edges of the α-shape computed centrally. Compared to results from previous studies the local α-shape algorithm performs favourably. This is due, in part, to the use of partial location information.

Finally we presented the Prohibitive-link Detection and Routing Protocol, PDRP, a guaranteed routing algorithm that follows the same principles as current face-routing schemes. It operates locally and, unlike planarisation methods, omits only prohibitive rather than nonessential links. The approach taken in this work was to enumerate the configurations that form an intersection in the network graph. In doing so we recognised the existence of a prohibitive link that has the potential to conceal other links from a left-hand rule traversal. We implemented PDRP in TinyOS; in TOSSIM simulations PDRP performance was found to be similar to, and in some cases better than, current face-routing schemes.

6.2 Conclusive Remarks

From our research, we make the following remarks:

- Error and inconsistency is known to generally affect geometries negatively. Despite this trend, there appears to be certain structures that can be designed to be resilient to such errors.

- It is possible to construct (at least one) global structure using only local communication and computation.
- Planarity of the network graph is not a requirement for successful delivery using left- (or right-)hand rule.

We conclude this document with some notes on future directions.

6.3 Future Work

Wireless sensor networks provide an ample new set of networking problems with unique geometric properties. A thorough understanding of these properties, as exemplified in the design of *lcv*, localised α-hulls, and PDRP, permits the design of better solutions.

In the future this work will take five directions. We intend first to investigate the performance of the localised α-hull in the presence of position-estimation error, and explore statistical and optimisation techniques to resolve the expected degradation. In addition we hope to adapt α-hulls for boundary detection in 3-dimensional networks such as those underwater.

By design, PDRP demands either that nodes must communicate to omit prohibitive links, or embed those links in recovery packets. The former case increases the messaging complexity, yet the latter increases packet size. We hope to encode prohibitive links into the packet using Bloom filters, thereby providing a version of PDRP that reduces resources while providing routing guarantees with high probability.

The enumeration of intersections studied in this document is in the context of unit-disc graphs. We intend to study intersections in general graphs with no such constraint. LHR has so far eluded delivery guarantees in general networks. We hope our approach helps to break this trend. Finally, we expect that other routing protocols may be improved by a better understanding of the challenges they face. We hope

to continue to remove obstacles faced, rather than design protocols to overcome the obstacles as they are encountered.

References

[1] A. E. Abdallah, Thomas Fevens, and Jaroslav Opatrny. High delivery rate position-based routing algorithms for 3d ad hoc networks. *Elsevier Computer Communications*, 31(4):807–817, 2008.

[2] Nadeem Ahmed, Salil S. Kanhere, and Sanjay Jha. The holes problem in wireless sensor networks: a survey. *SIGMOBILE Mobile Computer Communication Review*, 9(2):4–18, 2005.

[3] Ian Akyildiz, Welljan Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A Survey on Sensor Networks. *IEEE Communications Magazine*, August 2002.

[4] Filipe Araújo and Luís Rodrigues. Fast localized delaunay triangulation. *Springer LCNS Principles of Distributed Systems*, 3544/2005:81–93, 2005.

[5] Abdalkarim Awad, Christoph Sommer, Reinhard German, and Falko Dressler. Virtual Cord Protocol (VCP): A Flexible DHT-like Routing Service for Sensor Networks. In *5th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2008)*, Atlanta, GA, USA, September 2008. to appear.

[6] Hari Balakrishnan, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Looking up data in p2p systems. *Communications of the ACM*, 46(2):43–48, 2003.

[7] 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 (MobiCom, pages 76–84, 1998.

[8] Mirela Ben-Chen, Craig Gotsman, and Steven Gortler. Routing with guaranteed delivery on virtual coordinates. In *Proceedings of the 18th Canadian Conference on Computational Geometry (CCCG'06)*, pages 117–120, 2006.

[9] Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with guaranteed delivery in *ad hoc* wireless networks. In *Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DialM)*, 1999.

[10] 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.

[11] Jehoshua Bruck, Jie Gao, and Anxiao Jiang. Map: medial axis based geometric routing in sensor networks. *Kluwer/Springer Wireless Networks*, 13(6):835–853, 2007.

[12] Qing Cao and Tarek Abdelzaher. A scalable logical coordinates framework for routing in wireless sensor networks. *Proceedings of the 25th IEEE Real-Time Systems Symposium (RTSS)*, pages 349–358, 2004.

[13] Niklas Carlsson and Derek L. Eager. Non-euclidian geographic routing in wireless networks. *Elseiver Ad Hoc Networks*, 5(7):1173–1193, 2007.

[14] Dazhi Chen and Pramod K. Varshney. On-demand geographic forwarding for data delivery in wireless sensor networks. *Elsevier Computer Communications*, 30(14-15):2954–2967, 2007.

[15] Dazhi Chen and Pramod K. Varshney. A survey of void handling techniques for geographic routing in wireless networks. *Communications Surveys & Tutorials, IEEE*, 9(1):50–67, First Quarter 2007.

[16] Min Chen, Victor C. M. Leung, Shiwen Mao, and Yong Yuan. Directional geographical routing for real-time video communications in wireless sensor networks. *Elsevier Computer Communications*, 30(17):3368–3383, 2007.

[17] Douglas S. J. De Couto, Daniel Aguayo, Benjamin A. Chambers, and Robert Morris. Performance of multihop wireless networks: shortest path is not enough. *SIGCOMM Compututer Communication Review (CCR)*, 33(1):83–88, 2003.

[18] Luke Demoracski and Dimiter R. Avresky. Performance analysis of fault-tolerant beacon vector routing for wireless sensor networks. In *MSWiM '05: Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 40–44, New York, NY, USA, 2005.

[19] Herbert Edelsbrunner. Weighted alpha shapes. Technical Report UIUCDCS-R-92-1760, University of Illinois at Urbana-Champaign, Champaign, IL, USA, 1992.

[20] Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *Information Theory, IEEE Transactions on*, 29(4):551–559, July 1983.

[21] Herbert Edelsbrunner and Ernst P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[22] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: scalable coordination in sensor networks. In *Proceedings of*

the fifth annual ACM/IEEE international conference on Mobile computing and networking, pages 263–270, Seattle, WA USA, 1999.

[23] Qing Fang, Jie Gao, and Leonidas Guibas. Locating and Bypassing Routing Holes in Sensor Networks. In *Proceedings of IEEE/ACM Infocom*, Hong Kong, China, March 2004.

[24] Qing Fang, Jie Gao, Leonidas J. Guibas, Vin de Silva, and Li Zhang. Glider: gradient landmark-based distributed routing for sensor networks. In *24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 339–350, Miami, FL, USA, March 2005.

[25] Marwan Fayed and Hussein T. Mouftah. Characterizing the Impact of Routing Holes on Geographic Routing. In *Proceedings of Systems Communications 2005 (ICW / ICHSN / ICMCS / SENET 2005)*, pages 401 – 406, August 2005.

[26] Sandor P. Fekete, Alexander Kröller, Dennis Pfisterer, Stefan Fischer, and Carsten Buschmann. Neighborhood-based topology recognition in sensor networks. In *Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*, volume 3121 of *Lecture Notes in Computer Science*, pages 123–136. Springer, 2004.

[27] G.G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISI/RR-87-180, Information Sciences Institute (ISI), March 1987.

[28] Rodrigo Fonseca, Sylvia Ratnasamy, Jerry Zhao, Cheng Tien Ee, David Culler, Scott Shenker, and Ion Stoica. Beacon vector routing: Scalable point-to-point routing in wireless sensornets. In *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI '05)*, Boston, MA, USA, May 2005.

[29] Hannes Frey and Daniel Görgen. Planar graph routing on geographical clusters. *Elsevier Ad Hoc Networks*, 3(5):560–574, 2005.

[30] Hannes Frey and Ivan Stojmenovic. On delivery guarantees of face and combined greedy-face routing in ad hoc and sensor networks. In *the 12th annual international conference on Mobile computing and networking (Mobicom)*, pages 390–401, 2006.

[31] Stefan Funke. Topological hole detection in wireless sensor networks and its applications. In *Proceedings of the 2005 joint workshop on Foundations of mobile computing (DIALM-POMC '05)*, pages 44–53, New York, NY, USA, 2005.

[32] Stefan Funke and Christian Klein. Hole detection or: "how much geometry hides in connectivity?". In *Proceedings of the* 22^{nd} *annual symposium on Computational Geometry (SGC)*, pages 377–385, 2006.

[33] Stefan Funke and Nikola Milosavljevic. Guaranteed-delivery geographic routing under uncertain node locations. In *26th IEEE International Conference on Computer Communications (INFOCOM).*, pages 1244–1252, Anchorage, AK, USA, May 2007.

[34] Holger Fler, Jrg Widmer, Michael Ksemann, Martin Mauve, and Hannes Hartenstein. Contention-Based Forwarding for Mobile Ad-Hoc Networks. *Elsevier's Ad Hoc Networks*, 1(4):351–369, November 2003.

[35] L. Galluccio, A. Leonardi, G. Morabito, and S. Palazzo. A mac/routing cross-layer approach to geographic forwarding in wireless sensor networks. *Elsevier Ad Hoc Networks*, 5(6):872–884, 2007.

[36] Deepak Ganesan, Deborah Estrin, and John Heidemann. Dimensions: why do we need a new data handling architecture for sensor networks? *SIGCOMM Computer Communication Review (CCR)*, 33(1):143–148, 2003.

[37] Jie Gao, Leonidas J. Guibas, John Hershberger, Li Zhang, and An Zhu. Geometric spanner for routing in mobile networks. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing (MobiHoc*, pages 45–55, 2001.

[38] Venkata C. Giruka and Mukesh Singhal. A self-healing on-demand geographic path routing protocol for mobile ad-hoc networks. *Elseiver Ad Hoc Networks*, 5(7):1113–1128, 2007.

[39] Young-Jin Kim Ramesh Govindan, Brad Karp, and Scott Shenker. Lazy cross-link removal for geographic routing. In *the 4th international conference on Embedded networked sensor systems (SenSys)*, pages 112–124, 2006.

[40] Benjamin Greenstein, Deborah Estrin, Ramesh Govindan, Sylvia Ratnasamy, and Scott Shenker. Difs: A distributed index for features in sensor networks. *Elsevier Journal of Ad Hoc Networks*, 2003.

[41] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, and Tarek Abdelzaher. Range-free localization schemes for large scale sensor networks. In *Proceedings of the 9th annual international conference on Mobile computing and networking (Mobicom)*, pages 81–95, 2003.

[42] Wendi Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking (Mobicom)*, pages 174–185, 1999.

[43] Marc Heissenbüttel, Torsten Braun, Markus Wälchli, and Thomas Bernoulli. Evaluating the limitations of and alternatives in beaconing. *Elsevier Ad Hoc Networks*, 5(5):558–578, 2007.

[44] Marc Heissenbttel, Torsten Braun, Thomas Bernoulli, and Markus Wlchli A. Blr: Beacon-less routing algorithm for mobile ad-hoc networks. *Elseviers Computer Communications Journal (Special Issue*, 27:1076–1086, 2004.

[45] Jeffrey Hightower and Gaetano Borriello. Location Systems for Ubiquitous Computing. *IEEE Computer Magazine*, pages 57–66, August 2001.

[46] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. *SIGPLAN Notices*, 35(11):93–104, 2000.

[47] Ting-Chao Hou and Victor Li. Transmission Range Control in Multihop Packet Radio Networks. *IEEE Transactions on Communications*, 34(1):38–44, 1986.

[48] Lingxuan Hu and David Evans. Localization for mobile sensor networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking (MobiCom*, pages 45–57, 2004.

[49] Zhen Jiang, Junchao Ma, Wei Lou, and Jie Wu. An information model for geographic greedy forwarding in wireless ad-hoc sensor networks. In *IEEE 27th Conference on Computer Communications (INFOCOM).*, pages 825–833, Pheonix, AZ, USA, April 2008.

[50] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*. Kluwer Academic Publishers, 1996.

[51] Sam Bass Warner Jr. *The Urban Wilderness: A History of the American City*. University of California Press, 1995. ISBN-10: 0520202244.

[52] Hanna Kalosha, Amiya Nayak, Stefan Rührup, and Ivan Stojmenovic. Select-and-protest-based beaconless georouting with guaranteed delivery in wireless

sensor networks. In *IEEE 27th Conference on Computer Communications (INFOCOM).*, pages 346–350, Pheonix, AZ, USA, April 2008.

[53] Holger Karl and Andreas Willig. *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, 2005.

[54] Brad Karp. Challenges in geographic routing: Sparse networks, obstacles, and tra ffic provisioning. presented at DIMACS Workshop on Pervasive Networking, May 2001.

[55] Brad Karp and H.T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of ACM MobiCom*, Boston, MA, September 2000.

[56] Yongjin Kim, Jae-Joon Lee, and Ahmed Helmy. Modeling and analyzing the impact of location inconsistencies on geographic routing in wireless networks. *SIGMOBILE Mobile Computer Commununication Review*, 8(1):48–60, 2004.

[57] Yongjin Kim, Jae-Joon Lee, and Ahmed Helmy. Modeling and analyzing the impact of location inconsistencies on geographic routing in wireless networks. *SIGMOBILE Mobile Computer and Communication Review (MC2R)*, 8(1):48–60, 2004.

[58] Young-Jin Kim. CLDP+GFR ver-0.2 for the motes. http://enl.usc.edu/software.html, 2005.

[59] Young-Jin Kim, Ramesh Govindan, Brad Karp, and Scott Shenker. Geographic routing made practical. In *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Boston, MA, USA, May 2005.

[60] 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 (DIALM-POMC*, pages 34–43, 2005.

[61] Robert Kleinberg. Geographic routing using hyperbolic space. In *26th IEEE International Conference on Computer Communications (INFOCOM).*, pages 1902–1909, Anchorage, AK, USA, May 2007.

[62] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom*, pages 66–75, 1998.

[63] Evangelos Kranakis, Harvinder Singh, and Jorge Urrutia. Compass Routing on Geometric Networks. In *Proceedings of the* 11^{th} *Canadian Conference on Computational Geometry*, pages 51–54, Vancouver, August 1999.

[64] Alexander Kröller, Sándor P. Fekete, Dennis Pfisterer, and Stefan Fischer. Deterministic boundary recognition and topology extraction for large sensor networks. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm (SODA)*, pages 1000–1009, New York, NY, USA, 2006.

[65] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Unit Disk Graph Approximation. In *ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, Philadelphia, PA, USA, October 2004.

[66] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric Ad-Hoc Routing: Of Theory and Practice. In *22nd ACM Symposium on the Principles of Distributed Computing (PODC)*, Boston, Massachusetts, USA, July 2003.

[67] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Asymptotically Optimal Geometric Mobile Ad-Hoc Routing. In *6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM)*, Atlanta, Georgia, September 2002.

[68] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Ad-Hoc Networks Beyond Unit Disk Graphs. In *1st ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, San Diego, CA, USA, September 2003.

[69] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Worst-Case Optimal and Average-Case Efficient Geometric Ad-Hoc Routing. In *4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, Annapolis, Maryland, USA, June 2003.

[70] Sungoh Kwon and Ness B. Shroff. Geographic routing in the presence of location errors. *Elsevier Computer Networks*, 50(15):2902–2917, 2006.

[71] Koen Langendoen and Niels Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. *Computer Networks*, 43(4):499–518, 2003.

[72] Loukas Lazos and Radha Poovendran. Hirloc: high-resolution robust localization for wireless sensor networks. *IEEE Journal on Selected Areas in Communications.*, 24(2):233–246, February 2006.

[73] Seungjoon Lee, Bobby Bhattacharjee, and Suman Banerjee. Efficient geographic routing in multihop wireless networks. In *the 6th ACM international symposium on Mobile ad hoc networking and computing (MOBIHOC)*, pages 230–241, 2005.

[74] Ben Leong, Barbara Liskov, and Robert Morris. Geographic routing without planarization. In *Proceedings of the 3rd USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, USA, May 2006.

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

[76] Fang-Yie Leu and Guo-Cai Li. A scalable sensor network using a polar coordinate system. *Elsevier Signal Processing*, 87(12):2978–2990, 2007.

[77] Philip Levis, Eric Brewer, David Culler, David Gay, Samuel Madden, Neil Patel, Joe Polastre, Scott Shenker, Robert Szewczyk, and Alec Woo. The emergence of a networking primitive in wireless sensor networks. *Communications of the ACM*, 51(7):99–106, 2008.

[78] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *1st International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 126–137, 2003.

[79] Teng Li, Anthony Ekpenyong, and Yih-Fang Huang. A Location System Using Asynchronous Distributed Sensors. In *Proceedings of IEEE/ACM Infocom*, Hong Kong, China, March 2004.

[80] Xiang-Yang Li, G. Calinescu, Peng-Jun Wan, and Yu Wang. Localized delaunay triangulation with application in ad hoc wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(10):1035–1047, October 2003.

[81] Xin Li, Young Jin Kim, Ramesh Govindan, and Wei Hong. Multi-dimensional range queries in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys)*, pages 63–75, 2003.

[82] Menghow Lim, Adam Greenhalgh, Julian Chesterfield, and Jon Crowcroft. Landmark guided forwarding. In *the 13TH IEEE International Conference on Network Protocols (ICNP).*, pages 169–178, 2005.

[83] Nathan Linial, Laszlo Lovasz, and Avi Widgerson. Rubber Bands, Convex Embeddings, and Graph Connectivity. *Combinatorica*, 8(1):91–102, 1988.

[84] Cong Liu and Jie Wu. Destination-region-based local minimum aware geometric routing. In *IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems (MASS).*, pages 1–9, October 2007.

[85] Ke Liu and Nael Abu-Ghazaleh. Stateless delivery guaranteed geometric routing for virtual coordinates on wsn (short paper). In *5th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, Atlanta, GA, USA, September 2008. to appear.

[86] Christian Lochert, Martin Mauve, Holger Fler, and Hannes Ha rtenstein. Geographic Routing in City Scenarios. *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R)*, 9(1):69–72, January 2005.

[87] Xiaoli Ma, Min-Te Sun, Xiangqian Liu, and Gang Zhao. Improving geographical routing for wireless networks with an efficient path pruning algorithm. In *IEEE 3rd Annual Sensor and Ad Hoc Communications and Networks Conference (SECON)*, pages 246–255, Reston, VA, USA, September 2006.

[88] 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 *NSDI*, Cambridge, MA, USA, April 2007.

[89] Kousha Moaveninejad, Wen-Zhan Song, and Xiang-Yang Li. Robust position-based routing for wireless ad hoc networks. *Elsevier Ad Hoc Networks*, 3(5):546–559, 2005.

[90] Jongkeun Na and Chong kwon Kim. Glr: a novel geographic routing scheme for large wireless ad hoc networks. *Elsevier Computer Networks*, 50(17):3434–3448, 2006.

[91] Radhika Nagpal, Howard Shrobe, and Jonathan Bachrach. Organizing a global coordinate system from local information on an ad hoc sensor network. In *the 2nd International Workshop on Information Processing in Sensor Networks (IPSN)*. Springer Verlag Lecture Notes in Computer Science LNCS 2634, April 2003.

[92] Hidehisa Nakayama, Nirwan Ansari, Abbas Jamalipour, and Nei Kato. Fault-resilient sensing in wireless sensor networks. *Computer Communications*, 30(11-12):2375–2384, 2007.

[93] James Newsome and Dawn Song. Gem: graph embedding for routing and data-centric storage in sensor networks without geographic information. In *Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys)*, pages 76–88, 2003.

[94] Lionel M. Ni, Yunhao Liu, Yiu Lau, and Abhishek P. Patil. LANDMARK: Indoor Location Sensing Using Active RFID. In *Proceedings of IEEE International Conference on Pervasive Computi ng (PerCom)*, Fort Worth, TX, March 2003.

[95] Dragos Niculescu and Badri Nath. Ad hoc positioning system (aps. In *the IEEE Global Communications Conference (GLOBECOM)*, pages 2926–2931, San Antonio, TX, USA, 2001.

[96] Dragos Niculescu and Badri Nath. Position and orientation in ad hoc networks. *Elsevier Ad Hoc Networks*, 2(2):133–151, 2004.

[97] Joseph O'Rourke. *Computational Geometry in C*. Cambridge University Press, New York, NY, USA, 1998.

[98] Charles Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[99] Charles E. Perkins and Elizabeth M. Royer. Ad hoc on-demand distance vector routing (aodv). In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999.

[100] Anuj Puri Rahul Jain and Raja Sengupta. Geographical routing using partial information for wireless ad hoc networks. *IEEE Personal Communications*, 8(1):48–57, 2001.

[101] Ananth Rao, Sylvia Ratnasamy, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic routing without location information. In *Proceedings of ACM MobiCom*, San Diego, CA, September 2003.

[102] Sylvia Ratnasamy, Brad Karp, Scott Shenker, Deborah Estrin, Ramesh Govindan, Li Yin, and Fang Yu. Data-centric storage in sensornets with ght, a geographic hash table. *Mobile Network Applications*, 8(4):427–442, 2003.

[103] Juan A. Sanchez, Rafael Marin-Perez, and Pedro M. Ruiz. Boss: Beacon-less on demand strategy for geographic routing in wireless sensor networks. In *IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems (MASS).*, pages 1–10, October 2007.

[104] Sandor P. Fekete and Michael Kaufmann and Alexandor Krller and Katharina Lehmann. A new approach for boundary recognition in geometric sensor networks. In *Proceedings 17th Canadian Conference on Computational Geometry (CCCG)*, pages 82–85, 2005.

[105] Andreas Savvides and Wendy L. Garber. An analysis of error inducing parameters in multihop sensor node localization. *IEEE Transactions on Mobile Computing*, 4(6):567–577, 2005. Senior Member-Randolph L. Moses and Senior Member-Mani B. Srivastava.

[106] Christian Schindelhauer, Klaus Volbert, and Martin Ziegler. Geometric spanners with applications in wireless networks. *Computational Geometry: Theory and Applications*, 36(3):197–214, 2007.

[107] Karim Seada, Ahmed Helmy, and Ramesh Govindan. On the effect of localization errors on geographic face routing in sensor networks. In *Proceedings of the 3rd international symposium on Information processing in sensor networks (IPSN)*, pages 71–80, 2004.

[108] Karim Seada, Ahmed Helmy, and Ramesh Govindan. Modeling and analyzing the correctness of geographic face routing under realistic conditions. *Elsevier Ad Hoc Networks*, 5(6):855–871, 2007.

[109] Rahul C. Shah, Adam Wolisz, and Jan M. Rabaey. On the performance of geographical routing in the presence of localization errors. In *IEEE International Conference on Communications (ICC).*, volume 5, pages 2979–2985, May 2005.

[110] Yi Shang, Wheeler Ruml, Ying Zhang, and Markus P. J. Fromherz. Localization from mere connectivity. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing (MobiHoc)*, pages 201–212, New York, NY, USA, 2003.

[111] Scott Shenker, Sylvia Ratnasamy, Brad Karp, Ramesh Govindan, and Deborah Estrin. Data-centric storage in sensornets. *SIGCOMM Computer Communication Review*, 33(1):137–142, 2003.

[112] Francesco Sottile and Maurizio A. Spirito. Robust localization for wireless sensor networks. In *5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON).*, pages 46–54, Rome, Italy, June 2008.

[113] Ivan Stojmenovic, Mark Russell, and Bosko Vukojevic. Depth first search and location based localized routing and qos routing in wireless networks. In *Proceedings International Conference on Parallel Processing (ICPP)*, page 173, Washington, DC, USA, 2000. IEEE Computer Society.

[114] Sundar Subramanian, Sanjay Shakkottai, and Piyush Gupta. On optimal geographic routing in wireless networks with holes and non-uniform traffic. In *26th IEEE International Conference on Computer Communications (INFOCOM).*, pages 1019–1027, Anchorage, AK, USA, May 2007.

[115] Hideaki Takagi and Leonard Kleinrock. Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals. *IEEE Transactions on Communications*, 32(3):246–257, 1984.

[116] Tinyos source, software, and documentation. http://www.tinyos.net.

[117] Ming-Jer Tsai, Hong-Yen Yang, and Wen-Qian Huang. Axis-based virtual coordinate assignment protocol and delivery-guaranteed routing protocol in wireless sensor networks. In *26th IEEE International Conference on Computer Communications (INFOCOM).*, pages 2234–2242, Anchorage, AK, USA, May 2007.

[118] Serdar Vural and Eylem Ekici. Hop-distance based addressing and routing for dense sensor networks without location information. *Elsevier Ad Hoc Networks*, 5(4):486–503, 2007.

[119] Yue Wang, Jie Gao, and Joseph S.B. Mitchell. Boundary recognition in sensor networks by topological methods. In *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 122–133. ACM, 2006.

[120] Kamin Whitehouse, Chris Karlof, Alec Woo, Fred Jiang, and David Culler. The effects of ranging noise on multihop localization: an empirical study. In *Proceedings of the 4th international symposium on Information processing in sensor networks (IPSN)*, 2005.

[121] Alec Woo, Terence Tong, and David Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *the 1st international conference on Embedded networked sensor systems (SenSys)*, pages 14–27, 2003.

[122] Sungwon Yang, Jiyoung Yi, and Hojung Cha. Hcrl: A hop-count-ratio based localization in wireless sensor networks. In *4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON).*, pages 31–40, San Diego, CA, USA, June 2007.

[123] Fenghui Zhang, Hao Li, Anxiao Jiang, Jianer Chen, and Ping Luo. Face tracing based geographic routing in nonplanar wireless networks. In *26th IEEE International Conference on Computer Communications (INFOCOM).*, pages 2243–2251, Anchorage, AK, USA, May 2007.

[124] Gang Zhao, Xiangqian Liu, Min-Te Sun, and Xiaoli Ma. Energy-efficient geographic routing with virtual anchors based on projection distance. *Elsevier Computer Communications*, 31(10), 2008.

[125] 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 (SenSys)*, pages 1–13, 2003.

[126] 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 (ICNP)*, pages 179–190, 2005.

[127] Zizhan Zheng, Kai-Wei Fan, Prasun Sinha, and Yusu Wang. A distributed roadmap aided geographic routing protocol for sensor networks (short paper). In *5th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, Atlanta, GA, USA, September 2008. to appear.

[128] Xianjin Zhu, R. Sarkar, Jie Gao, and J.S.B. Mitchell. Light-weight contour tracking in wireless sensor networks. In *IEEE 27th Conference on Computer Communications (INFOCOM).*, pages 1175–1183, Pheonix, AZ, USA, April 2008.

Appendix

A-1 Confidence Intervals

Simulated measures throughout this work such as success rate, path stretch, and probability distributions, etc, are calculated by taking the mean values over a succession of n runs of simulated networks. All runs are identical and independent, with random values generated from non-overlapping streams. We represent simulation runs by $R_1, R_2, R_3, \ldots, R_n$. The mean value $\overline{R}$ in question is determined by,

$$\overline{R} = \frac{1}{n} \sum_{i=1}^{n} R_i \tag{A-1}$$

Since the summation $\overline{R}$ is only an estimate of the expected value $E[R] = \mu$, we need to determine the accuracy of $\overline{R}$ by calculating its confidence interval. We first compute the variance, V_R^2, by

$$V_R^2 = \frac{1}{n-1} \sum_{i=1}^{n} (R_i - \overline{R})^2 \tag{A-2}$$

A small variance indicates a tight clustering of measurements around the simulated value $\overline{R}$. This corresponds to an increase in confidence that $\overline{R}$ is close to $E[R]$. Conversely, a large variance indicates a wider spread of measured values, reducing the confidence that $\overline{R}$ is indicative of $E[R]$. The degree of confidence is determined by calculating the interval of values in which the true $E[R]$ is contained, and expressing the interval as a proportion of the measured value.

Confidence is determined first by selecting some high probability $1 - \alpha$ for some small α. Then the interval of lower and upper values for μ, L_R and U_R respectively,

may be expressed by the probability

$$P[L_R \leq \mu \leq U_R] = 1 - \alpha \tag{A-3}$$

In other words, the true value of μ is contained in the interval $P[L_R \leq \mu \leq U_R]$ with a probability of $1 - \alpha$. This is said to be the $100(1 - \alpha)$ confidence interval.

For example, the 95% confidence interval for a set of observations may be determined using the standard distribution and the student's t-distribution table as follows. For

$$\begin{aligned}
\alpha &= 0.05 \\
n &= \text{number of observations} \\
\overline{R} &= \text{average of sample observations} \\
\sigma &= \text{sample standard deviation} \\
&= \sqrt{V_R^2} \\
&= \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(R_i - \overline{R})^2}
\end{aligned}$$

the lower and upper values in the confidence interval have been calculated as,

$$\text{lower limit: } L_R = \overline{R} - \frac{\sigma t_{(1-\frac{\alpha}{2},n-1)}}{\sqrt{2}}, \text{ and} \tag{A-4}$$

$$\text{upper limit: } U_R = \overline{R} + \frac{\sigma t_{(1-\frac{\alpha}{2},n-1)}}{\sqrt{2}} \tag{A-5}$$

The observations in preceding chapters are 50 in number and fall from a true distribution that has two tails. Hence, from the t-distribution table, the two-tail value $t_{0.05,49}$ is found to be 2.010. All measurements in this work are presented using 95% or better confidence intervals.