Geometric routing over virtual coordinate systems: algorithms, protocols and applications

Wijesiriwardana Samarasinghe, Kasun

This publication URL: https://archive-ouverte.unige.ch/unige:91697
Publication DOI: 10.13097/archive-ouverte/unige:91697

UNIVERSITÉ DE GENÈVE FACULTÉ DES SCIENCES

Département d'Informatique Professeur Pierre Leone

Geometric Routing over Virtual Coordinate Systems: Algorithms, Protocols and Applications

THÈSE

présentée à la Faculté des sciences de l'Université de Genève
pour obtenir le grade de Docteur ès sciences, mention Sciences informatiques

par

Kasun WIJESIRIWARDANA SAMARASINGHE

de

Galle, Sri Lanka

Thèse N^o

GENÈVE
2016

To my parents. . .

Acknowledgements

I would like to thank Professor Pierre Leone, for his guidance and advices given through out my thesis. He has given me the freedom to explore new things and ideas as well as given me the directions whenever needed to make my work a success. Also I would like to thank Professor Jose Rolim for giving me the opportunity to work with his team. I thank all my colleagues and friends (past and present) in Theoretical Computer Science laboratory for helping me to complete my thesis, as well as for making a friendly and productive environment to work. I would also like to thank Professor Thiemo Voigt at Uppsala University, Sweden for introducing me to the research community and providing me support through out my post graduate studies. Last but not least, I am grateful for all my teachers and advisors at the University of Colombo, School of Computing, Sri Lanka and Richmond College, Galle, Sri Lanka, for setting me up the basis to pursue further studies in Computer Science

Geneva, 2016 Kasun Wijesiriwardana Samarasinghe

Abstract

Advancements in hardware development have given rise to a novel paradigm of computing called ubiquitous computing. In ubiquitous computing, applications provide various services across heterogeneous hardware platforms, over standard protocols. These hardware can range from mobile devices to small devices with wireless communication capabilities, which can be embedded into appliances utilized in daily life.

Wireless ad-hoc networks is an important ingredient in the deployment of ubiquitous computing applications. It acts as the communication backbone, which enables the flexible deployment of devices required by most of the ubiquitous applications. In general, wireless ad-hoc networks do not have a pre-defined topology, hence form a mesh network over wireless communication medium. Routing in wireless ad-hoc networks is a challenging task compared to fixed infrastructure networks. It has to deal with the dynamics of the network topology, due to the unpredictable wireless communication behavior.

Geographic routing is a routing paradigm, especially proposed for wireless ad-hoc networks. In geometric routing, routing decisions are made based on the geometric position of the nodes. Initially geometric routing tries to forward a packet to the closest node towards the destination, namely greedy routing. If greedy routing fails, it explores the graph systematically to reach the destination, namely face routing. As routing is based on the geometric coordinates of the nodes, it avoids building and maintaining the routing tables as in classical routing protocols. Hence, it provides scalability and robustness in networks with dynamic topologies.

Despite the ingenuity of seminal proposal in geometric routing, it has not thrive to become a standard for wireless ad-hoc networks. This is mainly due to the unrealistic assumptions of the algorithms. Also as geometric routing requires the nodes to be equipped with geometric coordinates, an auxiliary location service has to be in place. Such a service can be expensive and unrealistic in most of the wireless ad-hoc network deployments.

In this thesis, geometric routing is investigated on virtual coordinate systems. In the first contribution of the thesis, an efficient coordinate system called virtual raw anchor coordinates (VRAC) has been investigated and geometric routing algorithms are formulated on it. Further more, it establishes results on the possibility of greedy routing with delivery guarantees on VRAC. Second contribution of the thesis is the proposal of greedy zone routing protocol considering the reliability and scalability aspects of large scale wireless ad-hoc networks.

Abstract

Greedy zone routing protocol design not only free of unrealistic assumptions but also based on simple techniques which can be applied in real network settings. In the third contribution, an application layer framework is presented over existing routing protocols for wireless ad-hoc networks. This framework is then applied to solve a smart building use case.

Key words: Geometric Routing, Planar Graphs, Greedy Embedding

Résumé

Les progrès dans le développement de matériel ont donné lieu à un nouveau paradigme de l'informatique appelé l'informatique omniprésente. Dans l'informatique omniprésente, les applications offrent divers services à travers des plates-formes matérielles hétérogènes, sur des protocoles standards. Ces matériels peuvent varier de dispositifs mobiles à petits appareils dotés de capacités de communication sans fil, qui peuvent être intégrés dans des appareils utilisés dans la vie quotidienne.

Les réseaux ad-hoc sans fil sont un ingrédient important dans le déploiement d'applications informatiques omniprésentes. Ils agissent comme l'épine dorsale de communication, qui permet le déploiement flexible des dispositifs requis par la plupart des applications omniprésentes. D'une manière générale, les réseaux sans fil ad-hoc ne disposent pas d'une topologie prédéfinie, par conséquent, forment un réseau maillé sur support de communication sans fil. Routage dans les réseaux ad-hoc sans fil est une tâche difficile par rapport aux réseaux d'infrastructures fixes. Il doit faire face à la dynamique de la topologie du réseau, en raison du comportement de communication sans fil imprévisible.

Le routage géographique est un paradigme de routage, en particulier proposé pour les réseaux ad-hoc sans fil. Dans le routage géométrique, les décisions de routage sont effectués en fonction de la position géométrique des noeuds. Initialement, le routage géométrique tente de transmettre un paquet au noeud le plus proche vers la destination, à savoir le routage glouton. Si le routage glouton échoue, il explore le graphique systématiquement pour atteindre la destination, le face routage. Que le routage soit basé sur les coordonnées géométriques des noeuds, cela évite la construction et l'entretien des tables de routage, comme dans des protocoles de routage classiques. Par conséquent, il offre une évolutivité et de robustesse dans les réseaux avec des topologies dynamiques.

Malgré l'ingéniosité de la proposition séminal dans le routage géométrique, il n'a pas prospéré pour devenir une norme pour les réseaux ad-hoc sans fil. Ceci est principalement dû aux hypothèses irréalistes des algorithmes. Aussi comme le routage géométrique exige aux noeuds d'être équipés avec des coordonnées géométriques, un service de localisation auxiliaire doit être en place. Un tel service peut être coûteux et irréaliste dans la plupart des déploiements de réseau sans fil ad-hoc.

Dans cette thèse, le routage géométrique est étudiée sur des systèmes virtuels de coordonnées.

Dans la première contribution de la thèse, un système de coordonnées efficace appelé coordonnées d'ancrage premières virtuels (VRAC) a été étudié et des algorithmes de routage géométriques sont formulés à ce sujet. De plus, il établit les résultats sur la possibilité de routage glouton avec des garanties de livraison sur VRAC. La deuxième contribution de la thèse est la proposition de protocole glouton de routage par zones tenant compte des aspects de fiabilité et d'évolutivité des réseaux ad-hoc sans fil à grande échelle. La conception de ce protocole non seulement est libre d'hypothèses irréalistes, mais il est aussi basé sur des techniques simples qui peuvent être appliquées dans les paramètres de réseaux réels. Dans la troisième contribution, un cadre de la couche d'application est présenté sur les protocoles de routage existants pour les réseaux ad-hoc sans fil. Ce cadre est ensuite appliqué pour résoudre un cas d'utilisation d'un bâtiment intelligent.

Mots clefs : Geometric Routing, Planar Graphs, Greedy Embedding

Contents

List of Figures

List of Tables

1 Introduction

Rapid development in hardware technologies has drastically changed the face of computing in last couple of decades. Emergence of ubiquitous computing is an important realization of these developments. In ubiquitous computing, applications provide services across heterogeneous hardware platforms, over standard protocols. This hardware can range from mobile devices to small devices with wireless communication capabilities.

Internet of Things (IOT) is an important application domain in ubiquitous computing. It allows various devices in our ambient environment to connect to the Internet via standard protocols. These devices serve different purposes, such as sensing a particular phenomenon or acting as an actuator to control the device. This enables various applications, which involve collecting sensor data, processing them and control different appliances. For instance, a smart building application would collect the data from the sensors deployed in a building and process them. Based on the data, it controls the utilities in the building, such as heating and lighting. In order to support these applications, a robust and scalable network infrastructure has to be deployed. In such an infrastructure, data routing is an important service, which enables the whole information exchange.

Routing in communication networks is the process of forwarding information between the source of information and the intended destination. In a packet switched network like Internet, data is transmitted in the form of data packets, which are chunks of data bundled according to a certain protocol. For instance in TCP/IP networks, data is transmitted in the form of IP packets. A routing protocol has to discover the route (or a path) in the form of a sequence of intermediary nodes and then forward the packet to the next hop towards the destination. For the purpose of computing the routes, a network can be mathematically abstracted by a graph (referred to as the communication graph, see Section 2.1.2) with vertices representing network nodes and edges representing communication links. Given that the complete topology of the network is known (communication graph), a path can be computed by employing a classical shortest path algorithm on the communication graph.

The design of routing protocols depends on the characteristics of the underlying networks. For instance, wired links are more reliable and robust, compared to wireless links. Therefore, routing protocols for wired networks can assume a network topology, which does not change over time, whereas wireless network topology is highly dynamic. Moreover the size of the network is also important, as protocols have to scale in large scale networks. For instance, in local area networks (LAN), a spanning tree (see Section 2.1.1) of a communication graph is constructed to perform routing between the nodes.

In an abstract view point, a fixed infrastructure network organizes itself into manageable sized sub-networks. Routing within these sub-networks is performed by protocols local to these networks. In order to route globally within the network, protocols are established depending on the hierarchical relationships between the sub-networks. Due to its hierarchical nature, such a scheme is highly scalable in terms of number of nodes.

Wireless ad-hoc networks are a viable candidate, which provides the required flexibility for most of the ubiquitous computing scenarios. They comprise of devices that are capable of communicating over wireless medium. These networks do not have a specific network architecture or topology, rather forms a mesh network of devices.

Geometric routing is proposed as an alternative to the problem of scalable routing in wireless ad-hoc networks. It avoids building routing tables, instead uses geometric coordinates of the nodes to determine the routes. Most importantly, a node only requires the local neighborhood information (geometric coordinates of the neighbors) and the geometric coordinates of the destination. Hence, geometric routing is considered to be a *local routing* mechanism. This is very important, because with the size of the network, it does not require to gather the knowledge of the topology like in classical routing schemes.

Geometric routing has two main challenges, when it comes to the application in network protocols. Firstly, an auxiliary location service has to be in place to provide the geometric coordinates to the nodes. Depending on the network of interest, such a service can be expensive. For instance, one obvious choice is to equip the nodes with Global Position System (GPS) devices, which can be quite expensive in terms of energy consumption and cost. Also for indoors, GPS is not an option, therefore alternative localization mechanisms are required.

Secondly, assumptions made by geometric routing algorithms makes them inapplicable in real world wireless ad-hoc network conditions. For instance, assumption of ideal radio conditions (hence uniform communication range) is impractical in real networks, leads the algorithms to fail in practice.

In order to make geometric routing practical, it is important to design algorithms without the unrealistic assumptions. Furthermore, underlying coordinate system has to be computed in a cost effective manner.

1.1 Thesis Overview

In chapter 2 a background on the problem of routing in wireless ad-hoc networks presented. Chapter 3 presents a literature review on routing in ad-hoc networks, including a presentation of the main contributions in geometric routing research. Chapter 4.2 introduces the Virtual Raw Anchor Coordinate (VRAC) system and the first main contributions of the thesis. The second main set of contributions are presented in chapter 5, where a scalable and robust geometric routing protocol is introduced. From an application perspective, chapter 6 presents an Internet of things application scenario. It illustrates a service oriented framework for IOT applications over existing IPv6 routing protocol. This is followed by the summary of the thesis and concluding remarks in chapter 7.

2 Background

In this chapter, we develop the required background in routing in wireless ad-hoc networks. An overall introduction to the routing problem is presented in section 2.1. In section 2.1.1, we introduce the concepts from graph theory to model networks as graphs. Finally, classical algorithmic techniques for route discovery along with the communication complexity aspects of them are briefed in section 2.1.2.

2.1 Routing in Communication Networks

Data routing in packet switched networks is a fundamental service in the network stack. In a classical network setting, it is performed by special devices called routers. In a general view point, a router has two components namely; control plane and forwarding plane[1]. The control plane maintains the route information (the next device to which the packet must be forwarded in order to reach a particular destination) in a routing table. In order to construct and maintain the routing table, control plane employs various algorithms and protocols (See section 2.1.1 and 2.1.2 for a general overview) depending on the network. The forwarding plane uses the routing table and performs the packet forwarding. In order to construct and maintain a routing table, control plane must gather information about the network topology. This is usually done with message passing between the routers according to the protocol employed.

2.1.1 Communication graph models

In network algorithm design, a network is mathematically abstracted as a graph.

Definition 1. *A graph is a pair $G = (V, E)$ consists of a set of vertices V and a set of edges E, such*

[1]Note that there are different naming conventions in the literature for the control plane and forwarding plane

that $E \subseteq V \times V$.

According to the definition, a graph is a combinatorial object, which represents the formation of the graph in terms of the sets of vertices and edges. As per the purpose of geometric routing (as well as for various other applications), a graph has to be represented in a geometric surface. Such a representation is often called as a *greedy embedding* or a *greedy drawing* of a graph. A greedy embedding can be defined as follows.

Definition 2. *Given a graph* $G(V,E)$, *an embedding of the graph onto a geometric surface* Σ *is a function* $\phi : V(G) \to \Sigma$.

In other words, the function ϕ maps every vertex into a geometric position on Σ, hence assigning it a coordinate. For instance, Σ can be the *Euclidean plane*, where as non-euclidean spaces are also studied in various different instances[2].

The communication graph is the graph representation of the network connectivity. Therefore a physical communication link between two nodes[3] in the network is represented as an edge. When a link is considered symmetric (both the nodes at the end points can communicate with each other) it is represented by an *undirected* edge, where as for an asymmetric link is represented as a *directed* edge. Note that in networks like wireless ad-hoc networks, there are asymmetric links due to the wireless communication irregularities,

In this thesis, we consider several classes of graphs. It is possible to distinguish them as geometric and non-geometric graphs, especially considering the overall problem of geometric routing. Non-geometric graphs does not contain any geometric information, but the abstract connectivity. Geometric graphs can be considered as graphs embedded on a geometric surface, hence containing geometric information.

Planar graphs

Planar graphs is an important class of graphs encounters in geometric routing, which is defined as below.

Definition 3 (Planar Graph)**.** *A planar graph is a graph, which can be drawn on the plane* $\mathbb{R}^2$ *without edge crossings.*

A planar graph drawing introduces the notion of *faces* in the graph. A face is a bounded or an unbounded region in a planar graph drawing. Bounded regions are called *inner-faces*,

[2]A graph with an embedding is referred to as an embedded graph in this thesis.

[3]In this thesis, node is referred to a physical node in the network, while a vertex is the graph theoretic representation of it.

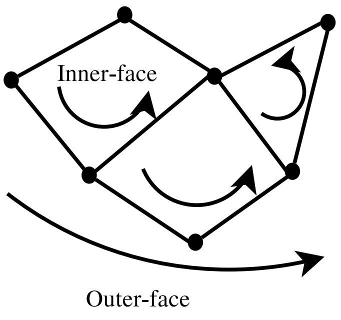


Figure 2.1 – Drawing of a planar graph and the faces of the graph

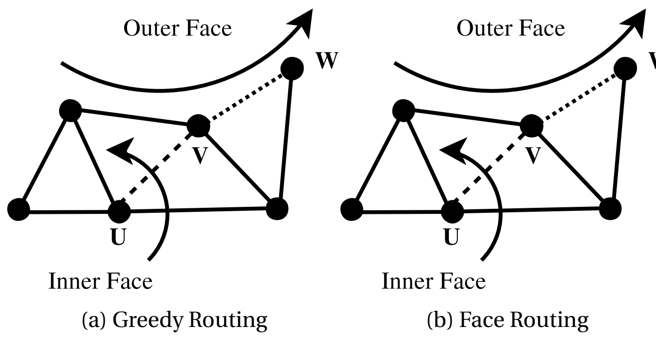


Figure 2.2 – Two phases of geometric routing

while the unbounded region is called the *outer-face*. A special instance of a planar graph is a maximal planar graph, where all the inner-faces of the graph are triangles. More formally, it can be defined as follows.

Definition 4 (Maximal Planar Graph)**.** *A maximal planar graph (triangular graph), is a planar graph, in which no more edges can be added such that the planarity is preserved.*

Another special instance of a maximal planar graph is a maximal outer-planar graph which is defined as follows.

Definition 5 (Maximal Outer-planar Graph)**.** *A maximal outer-planar graph is a maximal planar graph, where all the nodes lie on the outer-face of the graph.*

It is important to note that, even a geometric characterization is inherent in the definitions of these graph classes, they can also be defined completely in a combinatorial form. In other words, there are combinatorial conditions to be satisfied by planar graphs [Diestel, 1997], without a reference to its embedding.

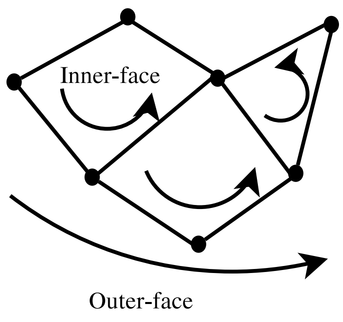


Figure 2.3 – Unit disk graph connectivity

Geometric Graphs

A geometric graph is a graph which is embedded on a geometric surface. In geometric routing, routing decisions are based on geometric information of nodes, hence the underlying communication graph has to be an embedded graph. As described before, connectivity in wireless ad-hoc networks depends on various factors such as radio communication characteristics. In order to design routing algorithms, simplified models of connectivity had been proposed, isolating the complex radio communication characteristics.

Definition 6 (Unit Disk Graph)**.** *[Huson and Sen, 1995] A unit disk graph (UDG) is a graph $G = (V, E)$, where two nodes in the network u and v are connected if the $d(u, v) \leq 1$ ($d(.)$ is the metric function of the respective geometric space).*

UDG model assumes a uniform radio communication range. Thus in a network embedded on a plane, a node is connected to all the other nodes within a circular region around itself. This assumption does not hold in practice, hence a more realistic model is proposed incorporating a probabilistic element in the connectivity.

Definition 7. *[Penrose, 2003] A random geometric graph (RGG) is a graph $G = (V, E)$, where the nodes are distributed on $\mathbb{R}^2$ or $\mathbb{R}^3$ at random and two nodes are connected if $d(u, v) \leq R$. (R is the communication radius of a node)*

This is considered to be a more realistic representation of a wireless ad-hoc network. The associated geometric assumptions can be used in designing algorithms and protocols with statistically motivated performance criteria.

2.1.2 Algorithms for route computation

Given a graph $G(V, E)$, problem of computation of a route between two nodes u and v is to find a path (see Appendix A.1) between u and v on $G(V, E)$. A route is optimal if the path is the shortest path between u and v. In some instances, sub optimal routes are computed, for which a quantity called *route stretch* is defined as follows.

Definition 8. *Route stretch of a route r between $u, v \in V$ is the quantity; $\frac{l_r}{l_s p}$, where l_r is the length of the path r and l_{sp} is the length of the shortest path between u and v.*

In a distributed setting to perform a route computation, every node needs to gather the information on the topology. In other words, communication graph has to be globally known. It is important to define a communication abstraction to study the complexity of such distributed algorithms. A synchronous communication model assumes that the nodes are synchronized in time and communication is performed in rounds (or time slots). In a given communication round, a node can send or receive messages of fixed length. This is an ideal model of message passing, where realistic concerns such as message collisions are disregarded. Nevertheless based on this model, the communication complexity of a distributed algorithm can be expressed in communication rounds required to terminate the algorithm.

Shortest path computation

Given a graph $G(V, E)$, to find a route between two nodes $u, v \in V$, the classical shortest path algorithm can be applied, where $|V| = n$. In protocols like OSPF and RIP, shortest path algorithms like Dijkstra and Bellmon-Ford [Cormen et al., 2001] are applied. The communication complexity of these algorithms depends on the cost of gathering the topology information. In general to gather the complete topology information, it takes $\mathcal{O}(n)$ rounds as it is directly proportional to the diameter of the graph. Moreover, once the routes are calculated, in these protocols it takes $\mathcal{O}(n)$ memory.

Spanning trees and tree routing

Tree based routing is another important concept in routing. The spanning tree protocol (STP) is a classical example of tree based routing, which is used in local area networks [Perlman, 1985]. The basic idea is to construct a tree structure rooted at an arbitrary node in a distributed manner and route along the paths of the tree. A node keeps track of the parent and the descendants in the tree structure. RPL is a protocol specifically designed for low power lossy networks, which uses a similar structure. More precisely, it maintains a directed acyclic graph (DAG).

To construct a tree in a network it takes $\mathcal{O}(n)$ communication rounds. Also every node has to maintain the list of its descendants, hence a memory of $\mathcal{O}(n)$ is required. Even though the worst case is as such, in practice, this requirement can be much less. Further more, route stretch of a tree routing can be arbitrarily long, resulting in an unbounded route stretch.

Nevertheless, in a practical setting tree based routing has attracted lot of attention due to the simplicity and efficiency.

3 Scalable Routing in Wireless Ad-hoc Networks

In this chapter, a concise literature review is presented on routing in wireless ad-hoc networks. After an introduction to the challenges of wireless ad-hoc network routing, classical protocols are reviewed. Section 3.2 introduces the concept of local and stateless routing followed by a review on related work on geometric routing in section 3.3. Section 3.4 presents more specific developments in geometric routing, especially on providing geometric coordinates to the nodes.

3.1 Routing in Wireless Ad-hoc Networks

Advancements in hardware technologies have made the communication and computation more ubiquitous. Not only with widely available hand-held devices with high end processors but also with various tiny artifacts with communication capabilities, have given rise to the computing paradigm called ubiquitous computing. It opens up the possibility for plethora of applications in various domains, ultimately enhancing the communication and computation experience of the end users.

Wireless ad-hoc networks is a class of networks, which plays a major role in providing the underlying communication infrastructure for ubiquitous systems. These networks are comprised of devices connected over the wireless medium, without a predefined infrastructure. Devices within the wireless communication range of each other connect and establish a link between them. Communication between two such devices are refereed to as single hop communication. The overall network is the collection of these communication links, which forms a wireless mesh between the devices. Communication between two devices which are not in each one's vicinity has to be done in multiple hops.

The connectivity of the network is subject to various different factors such as; wireless hardware of the devices and the quality of wireless links between the devices. In turn, quality of wireless links depends upon radio communication characteristics like, interference and muli-path fading. As a result wireless links can be disrupted and re-established over time. Hence not only the connectivity mesh is ad-hoc, but also it changes over time.

Typically the devices used in wireless ad-hoc networks are constrained by computation and power supply. Thus the operating system as well as the applications have to consider these limitations in their design. Despite the challenges posed by characteristics of wireless medium, wireless ad-hoc networks offers a great flexibility in terms of the deployment. This very reason lead various applications of wireless ad-hoc networks in various domains. A classical example of a wireless ad-hoc network is environmental monitoring, where a set of wireless devices with sensory capabilities are deployed to gather real time environmental data. Another application is to augment the environment with devices to create smart environments with certain capabilities to improve the quality of life of the inhabitants.

It is important to note that the success of fixed infrastructure networks is due to the extent of human administration along with the hierarchical routing structure. In contrast, human intervention in wireless ad-hoc network deployments has to be minimized. Therefore a possible hierarchical routing structure in this context has to be self organizing. Cluster based routing is a hierarchical routing scheme proposed in the general context of wireless ad-hoc networks. It partitions the network into clusters and constructs a routing structure between the clusters. Clustering is a well studied problem, where there are various schemes proposed for different networking scenarios. Therefore with an efficient clustering scheme, a cluster based routing can be realized.

3.1.1 On-demand Routing

One of the alternative paradigms for dynamic networks is on-demand routing. As the name suggests, instead of pre-computing the routes they are discovered upon the routing request. Ad-hoc On-Demand Vector routing (AODV) [Perkins and Royer, 1999] is a seminal work on this line of research. In AODV, routes are discovered per routing request.

3.1.2 Hierarchical Routing

A generic cluster based routing scheme, works in two layers. In the bottom layer, routing can be performed based on a tree-like structure, where all the nodes in the cluster include in a tree rooted at the cluster head (can be arbitrary node). In the top level, a routing scheme has to be maintained between the clusters to support inter cluster routing. A straight forward choice is to perform a shortest path algorithm on between the cluster level and maintain a routing table. This approach is proposed in [Haas, 1997], where they perform classical link state routing in cluster level. Link state protocol needs to acquire the complete network topology knowledge by flooding the network between the clusters.

3.2 Local and Stateless Routing

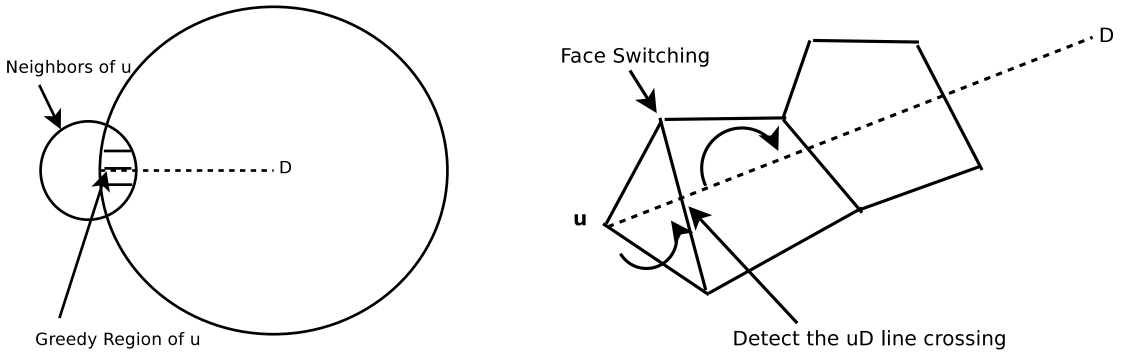
In a routing algorithm, it is important to quantify the amount of knowledge on the communication graph required by a routing algorithm. When the algorithm has the complete knowledge of the communication graph, it is considered to be a global algorithm. More formally, a routing decision is made at a given node based on $\theta(n)$ neighborhood. In another perspective, it suggests to perform routing only by use of local neighborhood information, namely a *local algorithm*. Accordingly, a local algorithm (or 1-local algorithm) executes by only using the $1 - hop$ neighborhood information of the network graph. Therefore algorithms which require *k-hop* neighborhood information when k is a constant, can still be considered as a local algorithm. Nevertheless, when k grows with the number of nodes in the network ($k = O(n)$) it can no longer be called as a local algorithm. An important restriction on local routing algorithms were presented in [Bose et al., 2013], which is summarized in Theorem 1.

Theorem 1. *[Bose et al., 2013] For every $k < \lfloor n/2 \rfloor$, every k-local routing algorithm fails on some connected graph.*

Therefore given an arbitrarily connected graph, it is not possible to route only with a constant neighborhood information. Nevertheless, if the given graph is a geometric graph and certain connectivity conditions are met, local routing algorithms exists (see Section 3.3).

Another important quantification of a routing algorithm is the routing state (information stored in the memory) maintained by a routing algorithm. The routing state can be in two forms, as the state stored at a node and the state maintained in the message header. At a given node state can be maintained per routing request (like in AODV [Perkins and Royer, 1999], where details of routes are stored) or to maintain a distributed routing structure like a spanning tree. For instance, in a classical routing protocol like RIP (see section 2.1.2), the state maintained at a node is $\mathcal{O}(n)$. Since a message header is limited, ideally it should carry a constant number of node information.

In terms of the scalability of a routing protocol, it is important to consider both locality and the routing state maintained. In particular, for large scale ad-hoc networks a routing protocol has to trade-off between these two quantities in order to achieve scalability.



(a) Greedy Routing (b) Face Routing

Figure 3.1 – Two phases of geometric routing

3.3 Geometric Routing

Geometric routing [Bose et al., 2001; Karp and Kung, 2000] is a routing paradigm, specially designed for wireless ad-hoc networks. It avoids constructing and maintaining routing tables, instead uses the geographic coordinates of nodes to make routing decisions. Nodes make the routing decisions only using their neighborhood information (coordinates of the neighbors) and the coordinates of the destination. Therefore it keeps the communication overheads minimal, in turn saving energy consumption of the nodes. There are two phases in geometric routing, namely *greedy routing* and *face routing*. It is important to note that, geometric routing requires the nodes to be embedded in a geometric surface.

3.3.1 Greedy Routing

Greedy routing (also refereed to as greedy forwarding) is the simplest form of geometric routing. It makes a local *greedy decision*, when forwarding a packet, simply choosing the geographically closest neighbor towards the destination as the successor. Let S be the source node and D be the destination node. At a given node u starting from S, it decides the next node v to which the packet has to be forwarded according to the following criterion. $\mathcal{N}_v$ is the set of *1-hop* neighbors of v and $d(.)$ is the underlying distance function in the geometric surface.

$$v = \underset{v \in \mathcal{N}_v}{\operatorname{argmin}} \; d(v, D) \tag{3.1}$$

Depending on the graph embedding, distance function can be either Euclidean or non-Euclidean. These distance function (or metrics) usually needs to follow the metric criteria.

Even more abstract conditions on a distance function for greedy routing was proposed in [Li et al., 2010]. It showed that in order to form a greedy path, following conditions must be satisfied by the distance function.

1. Transitivity: if node y is greedy for x and z is greedy for y then z is greedy for x
2. Anti-symmetry: if y is greedy for x then x is not for y

These properties can be used to form greedy routing algorithm, when metric functions are not available (See section 4.6).

3.3.2 Face Routing

Greedy routing may stuck in a node, where greedy routing is no longer possible. Such nodes are referred to as *local minima* or *routing voids*. Such void regions are unavoidable in wireless ad-hoc network deployments. In order to recover the routing process from a local minimum, GFG [Bose et al., 2001] and GPSR [Karp and Kung, 2000] introduced the concept of face routing. Face routing is a systematic exploration of the graph, which guarantees either to reach a node where greedy routing can be resumed or to reach the destination.

Face routing requires the underlying communication graph to be a planar graph. It uses the classical maze solving strategy called *left/right hand rule*. Initially it applies that rule and explores the current face, where greedy routing got stuck. Along the exploration, it performs a face switching, which guarantees a progress towards the destination (see Figure 3.1(b)).

Face routing guarantees the delivery of a message. A comprehensive analysis of delivery guarantees for different face switching strategies on different planar sub graphs are discussed in [Frey and Stojmenovic, 2010]. The strategy used in GFG is guaranteed to be delivered in arbitrary planar sub graph. Even though, face routing guarantees delivery, due to exhaustive exploration of the faces of the graph, routing stretch can be arbitrarily high. A worst case optimal face routing strategy was introduced in Kuhn et al. [2003] assuming a unit disk communication graph model.

In a practical perspective, local planarization of the communication graph is challenging. In GFG and GPSR they obtain a planar sub graph assuming an ideal radio communication characteristics (*unit disk graph*). Each node drops disregard links in the routing process, which could lead to crossing edges (hence non-planar) obtaining a planar sub graph. More specifically, in GPSR nodes locally apply a condition for planarity introduced in [Gabriel and Sokal, 1969]. Relative neighborhood graph [Toussaint, 1980] is also an alternative to gabrial graph, where the properties of it can be used to extract a connected planar sub graph.

Nevertheless, by the time of this writing, there is no planarization algorithm (which is local) known for arbitrarily connected graphs. An alternative proposal is to avoid the planarization [Leong et al., 2006], instead maintain a distributed data structure to cover the void region. This data structure is then used to overcome the dead end in greedy forwarding. Since this approach maintains a certain amount of state across some nodes in the network, it does not meet the original goal of geographic routing.

3.4 Coordinate Systems for Geometric Routing

In a practical perspective, providing geometric coordinates to wireless network nodes is a challenging problem. Especially with the energy constrains of the nodes, standard techniques such as Global Position System (GPS) is both infeasible and expensive. This has driven the wireless ad-hoc and sensor networks community to pursue viable localization protocols, not only for geometric routing, but also for various other location aware applications.

3.4.1 Localization in wireless ad-hoc networks

An alternative solution for expensive localization techniques is to equip small number of devices (so called anchor nodes) with location information (with GPS or other locating mechanism), and let the rest of the nodes (non-anchor nodes) to derive their location [Bulusu et al., 2000]. In order to calculate their position, non-anchor nodes have to measure the distances from anchor nodes (from minimum of three anchors in a two dimensional surface) and utilize basic geometric principles to arrive on their coordinates. This is called trilateration in the literature, while measurements from more than three anchors are used to get more accurate positions in the presence of erroneous range measures, which is called multilateration. This is only feasible when it is possible to equip anchor nodes with location information, which may be cumbersome in large scale deployments.

A localization algorithm takes a network deployment in two or three dimensional space as the input. It assigns the absolute or relative set of coordinates to the nodes, depending on the algorithm. For instance, if the nodes are equipped with GPS devices, they will get the absolute geographic coordinates. It can also assign relative coordinates. In either case, coordinate assignment must preserve the distances between nodes, if not it leads to localization errors. Such errors can affect geometric routing, leading the nodes to make incorrect routing decisions.

3.4.2 Virtual Coordinate Systems

Dependencies to specific hardware and manual configuration of anchor nodes hinder the practicality of most localization protocols [Bulusu et al., 2000; Leone et al., 2006]. An alternative is to compute the coordinates given the communication graph and inter distance between the nodes. This is a well studied problem called the *metric embedding* problem, which is proven to be NP-complete [Aspnes et al., 2004]. It is similar to the graph embedding problem, with the inter distance constraints. In geometric routing literature, such coordinate systems are referred as virtual coordinate systems.

Rao et.al [Rao et al., 2003] introduced the idea of using virtual coordinates in geometric routing in NoGeo protocol. NoGeo computes an embedding of the network on the Euclidean space, starting from an initial coordinate assignment at each node. It models the coordinate assignment problem as a mass-spring model and performs an iterative relaxation algorithm to achieve an approximation of the optimum coordinate assignment. In each iteration of the protocol, every node broadcast its coordinate to its neighboring nodes. Therefore in order to Shang et.al have incorporated inter distances between nodes into the coordinate construction problem, hence to compute an embedding of the network preserving the topological structure of the network [Shang et al., 2003]. This solution was based on a technique borrowed from psychometrics called multi dimensional scaling and based on an iterative approach. Initial algorithm was a centralized algorithm, but later distributed versions were proposed more applicable in real time network protocols [Shang and Ruml, 2004].

When considering a real time network protocol, most of the localization schemes proposed are far from being applicable. Mainly this is due to the communication overheads involved in computing the coordinates. An alternative direction on localization for geometric routing proposed to use the raw distances from a set of pre-designated anchor nodes (also referred to as beacons). Unlike in anchor based localization, these anchor nodes does not have any location information, rather acts as reference points for the non-anchor nodes. GLIDER [Fang et al., 2005], BVR [Fonseca et al., 2005] and VRAC [Huc et al., 2010] use this concept to localize nodes merely for the purpose of geometric routing. Both GLIDER and BVR use the hop distances from anchor nodes as the coordinates and define a heuristic distance function, which is used to perform greedy forwarding. When greedy forwarding reaches a *local minima*, both these protocols perform a scooped flooding to recover and reach the destination.

3.4.3 Greedy Embeddings

Constructing a greedy embedding is a solution to the problem of graph planarization. A greedy embedding always finds a closer neighbor towards the destination. It was conjectured [Papadimitriou and Ratajczak, 2004] and later proven that every 3-connected graph admits a greedy embedding [Leighton and Moitra, 2010] on the Euclidean plane. This is not favorable in real settings, where a communication graph can be simply connected. Further more, construction of such an embedding in a distributed fashion is not trivial. Another proposal in the spirit of greedy embedding was proposed in [], where they construct the embedding such that holes in the communication graph are drawn such that the greedy forwarding guarantees delivery. Despite the ingenuity in terms of the geometric construction, an implementation would be impractical due to the communication it may require to perform the computation. A greedy embedding is defined as follows.

Definition 9. *A greedy embedding is an embedding of a graph on a respective geometric space such that, greedy routing always succeeds. In other words, between every node pair u, v there is another node w adjacent to u, such that $d(u, v) > d(w, v)$, where $d(.)$ is the underlying metric on the geometric space.*

Papadimitrou & Ratajczak [Papadimitriou and Ratajczak, 2004] were the first to study the existence of *greedy embeddings*. In particular, they constructively proved the existence of a greedy embedding of a 3-connected graph in $\mathbb{R}^3$. Moreover, it is *conjectured* that any 3-connected planar graph admits a greedy embedding in $\mathbb{R}^2$.

Conjecture 1. *[Papadimitriou and Ratajczak, 2004] Every 3-connected finite graph has a greedy embedding in the Euclidean plane.*

The conjecture was proved affirmatively for different classes of graphs. In [Leighton and Moitra, 2010] it is proved for 3-connected graphs, in [Bose and Morin, 2004] for Delaunay triangulations, in [Chen et al., 2007] for graphs that satisfy conditions with respect to the power diagram. In [Kleinberg, 2007], a greedy embedding in the hyperbolic plane of a connected finite graphs is constructed. More related to our approach, in [Dhandapani, 2010] the conjecture is proven for planar triangulations (maximal planar graphs), see also [Angelini et al., 2010; He and Zhang, 2010, 2011]. Another direction of greedy embedding research considers the efficient representation of coordinates. In [Eppstein and Goodrich, 2009] authors proposed a greedy embedding on a hyperbolic space with $\mathcal{O}(log(n))$ bit complexity. Such a coordinate representation is called *succinct*, which is important for the design of scalable routing schemes. In subsequent research, succinct greedy routing schemes are proposed in [Goodrich and Strash, 2009; He and Zhang, 2010].

For arbitrarily connected graphs, existence of greedy embeddings on a non-Euclidean space is proven in [Kleinberg, 2007].

Theorem 2. *[Kleinberg, 2007] Every connected finite graph has a greedy embedding in the hyperbolic plane.*

They extract a spanning tree of the network and embed it on a hyperbolic space such that the tree distances are preserved. When considering the non-Euclidean greedy embedding proposals, a common procedure can be observed. Initially, a spanning tree is extracted and isometrically embedded onto the respective geometric space. An *isometric embedding* preserves the graph distances on the tree.

Lemma 1. *Let $T(V,E\prime)$ be a spanning tree of $G(V,E)$. An isometric embedding of $T(V,E\prime)$ is a greedy embedding of $G(V,E)$.*

Proof. Consider the R the root of the tree and an isometric embedding of the thee. Since there is only one path in a tree between any two nodes, for any given node u, either its parent or one of its descendants is closer to every node, hence a greedy embedding. □

It is important to note that even though the embedding is based on a tree, greedy routing strategy can consider all the edges of the network, when taking greedy forwarding decisions.

One drawback of tree-based greedy routing is the *unbounded path stretch* due to the underlying tree structure: shorter paths may be ignored and longer paths on the tree given preference. There are several attempts to overcome this by introducing multiple trees embedded instead of a single spanning tree [Herzen et al., 2011]. This suggests a node to hold multiple coordinates, corresponding to different spanning trees. Therefore a node has multiple choices to forward the packet. If the spanning trees are extracted accordingly, this approach could lead to shorter paths. Recently in [Houthooft et al., 2015] the authors propose the use of multiple spanning tree based greedy embeddings and focus on load balancing aspects. Nevertheless, neither of the two proposals provide guarantees on routing stretch. In contrast [Flury et al., 2009], proposes the use of multiple tree embeddings, where a constant factor stretch bound is proven for combinatorial unit disk graphs by computing a structure of trees which covers the graph with constant stretch. This is the only case for which a stretch bound is known. For arbitrary graphs no such bound is known.

4 Geometric Routing on Virtual Raw Anchor Coordinates

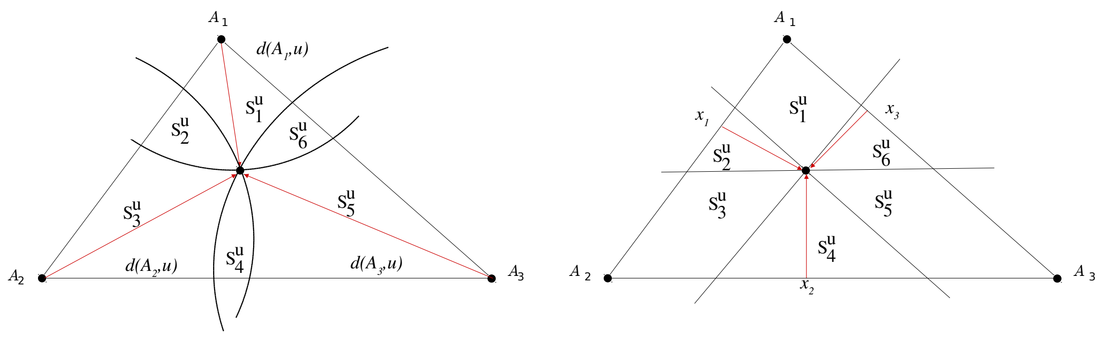
In this chapter, geometric routing algorithms on a coordinate system called Virtual Raw Anchor Coordinates(VRAC) is presented. VRAC is an efficient coordinate system devised for geometric routing in wireless ad-hoc and sensor networks. It is important to emphasize that algorithms presented are local algorithms, hence given a VRAC deployment no additional overhead is incurred. In the first part of this chapter, construction of VRAC and related definitions are presented. In section 4.4, a geometric routing algorithm is presented, where greedy and face routing is combined to guarantee the delivery. In section 4.6 a greedy routing algorithm is presented, when the graph connectivity follows certain conditions. Finally in section 4.7, a generic result on greedy embeddings is presented.

4.1 Why another coordinate system?

Localization of nodes in wireless ad-hoc networks is not a trivial task. Hardware dependent solutions, where a set of anchor nodes have to be designated, which requires human intervention and incur higher initialization costs. Furthermore, such solutions only work in outdoor environments, as the anchor nodes are equipped with GPS hardware. The alternative paradigm of virtual coordinates are inherently computationally intensive, since most of these mechanisms require iterative computations. As a result associated communication overheads are overwhelming especially in large scale networks.

Identifying these discrepancies, GLIDER Fonseca et al. [2005] and BVR Fang et al. [2005] have independently proposed an anchor based virtual coordinate scheme, specially for geometric routing. Given a pre-designated set of anchor nodes, both these protocols assign coordinates to the nodes, simply as a hop-count vector from the anchors. Therefore it does not require any further computation or complicated communication between nodes.

Virtual Raw Anchor Coordinate (VRAC) system is a coordinate system specially defined for



(a) Basic coordinate assignments with raw distances from anchors

(b) Coordinate assignment with perpendicular distances from edges of the triangles

Figure 4.1 – Two VRAC variants

geometric routing. Coordinate construction in VRAC is conceptually similar to GLIDER and BVR, where it assigns raw distances from anchors as the coordinates. In contrast to GLIDER and BVR, it uses the Euclidean distance from anchor nodes, instead of the hop distance.

4.2 Virtual Raw Anchor Coordinate System

In order to define the VRAC system, consider a network deployed along with three anchor nodes A_1, A_2 and A_3. If the nodes are distributed within the triangle formed by three anchors $\widehat{A_1 A_2 A_3}$, coordinates are defined as follows.

Definition 10. *A node u is assigned* $(u_1, u_2, u_3) = (d(u, A_1), d(u, A_2), d(u, A_3))$*, where* $d(u, A_i)$ *is the Euclidean distance from the anchor* A_i*.*

In this chapter, two versions of VRAC constructions are investigated. First version is the basic construction of coordinates according to the definition 10 (see Figure 4.1(a)). On this version, algorithms are based on the combinatorial structure of the coordinate system Samarasinghe and Leone [2014], which is presented in section 4.4. In the second version, we further assume that anchors can estimate the distances between them, hence the lengths of the sides of the triangle $\widehat{A_1 A_2 A_3}$ are known to each node. This leads to a slightly different version of VRAC (see Figure 4.1(b)), where perpendicular distances from triangle edges are assigned as coordinates. Algorithms on this coordinate system is presented in section 4.5.

Virtual Raw Anchor Coordinate space by definition purely is a virtual coordinate system, which does not correspond to the physical coordinates. Moreover, there is no *metric* associated in this coordinate system, hence it is not possible to perform the geometric computations required by geometric routing. This motivates us to explore combinatorial properties of VRAC which can be used in geographic routing.

Inspired by Schnyder characterization of planar graphs Schnyder [1989], we can define three order relations on the set of nodes V.

Definition 11. *The three order relations $<_i$, $i = 1,2,3$ on $V \times V$ are defined by*

$$\forall u, v \in V \quad u <_i v \Longleftrightarrow d(u, A_i) > d(v, A_i) \Longleftrightarrow u_i > v_i.$$

The three order relations are total[1], hence it is possible to define the minimum of a set with respect to the three orders. We will denote this by $\min_i$ for $i = 1,2,3$.

These three orders permit the definition of sectors associated with a node u.

Definition 12. *We define the following sectors associated to a node $u \in V$, see Figure 4.1(a). Note that the reference node u does not belong to the sectors.*

$$s_1^u = \{v \mid u <_1 v,\ u >_2 v,\ u >_3 v\} \cap \widehat{A_1 A_2 A_3}.$$
$$s_2^u = \{v \mid u <_1 v,\ u <_2 v,\ u >_3 v\} \cap \widehat{A_1 A_2 A_3}.$$
$$s_3^u = \{v \mid u >_1 v,\ u <_2 v,\ u >_3 v\} \cap \widehat{A_1 A_2 A_3}.$$
$$s_4^u = \{v \mid u >_1 v,\ u <_2 v,\ u <_3 v\} \cap \widehat{A_1 A_2 A_3}.$$
$$s_5^u = \{v \mid u >_1 v,\ u >_2 v,\ u <_3 v\} \cap \widehat{A_1 A_2 A_3}.$$
$$s_6^u = \{v \mid u <_1 v,\ u >_2 v,\ u <_3 v\} \cap \widehat{A_1 A_2 A_3}.$$

In the following, we refer to these sectors as sector number $1, 2, \ldots, 6$ respectively, i.e. for instance, s_4^u is sector number 4 of u. Notice that the node in the network are assumed to belong to the interior of the triangular region $\widehat{A_1 A_2 A_3}$, defined by the three anchors. This is necessary to the application of Schnyder's planarity criterion Schnyder [1989], that we use to extract a planar subgraph of the communication graph (see conditions 4.1 and definition 14).

Definition 13. *Given a node D, we also use the convenient notation s_D^u to denote the sector j of u such that $D \in s_j^u$, i.e. $D \in s_D^u$.*

[1]A total order is a binary relation which is valid for *all the pairs* in a set.

4.3 Graph planarization on virtual raw anchor coordinate system

For combined greedy face routing, a planar sub graph of the communication graph has to be extracted. Given the VRAC coordinates, it is not possible to extract a gabriel graph [Gabriel and Sokal, 1969] or relative neighborhood graph [Toussaint, 1980]. Nevertheless, with the combinatorial properties available in VRAC, it is possible to use the classical Schnyder characterization [Schnyder, 1989] defined below to formalize a planarization algorithm.Given a planar graph $G = (V, E)$, it is proven in Schnyder [1989] that there exists three total order relations on $V \times V$, denoted $<_1, <_2, <_3$ such that;

$$\begin{aligned} &a)\ \bigcap_{i=1,2,3} <_i = \emptyset, \text{ and} \\ &b)\ \forall (x,y) \in E, \forall z \notin \{x,y\}\ \exists i \in \{1,2,3\} \\ &\text{s.t. } x <_i z \text{ and } y <_i z. \end{aligned} \tag{4.1}$$

This is called a *(3-dimensional) representation* of a planar graph. Such representation of a planar graph does not use a (planar) embedding and applies to an abstract graph. Based on this characterization, Huc et al [Huc et al., 2012] defined a planar sub graph on the VRAC system as follows, in order to formulate a local planarization algorithm.

Definition 14. *Given a graph $G(V,E)$, a planar sub graph $\widetilde{G}(\widetilde{V}, \widetilde{E})$ can be defined such as*

1. $\widetilde{V} = V$
2. $\widetilde{E} = \left\{ (u,v) \middle| \ v \in s^u_{2k-1} \text{ and } v = \min_k(s^u_{2k-1})\ k = 1,2 \text{ or } 3 \text{ and } (u,v) \in E \right\}$

According to the definition, if a node can locally determine the minimum edge with respect to the order relation (in turn, with respect to the sector), it is possible to planarize the graph. Huc et al [Huc et al., 2012] showed that it is possible, if the connectivity graph follows the unit disk graph assumption. Thereby, they formulate a local planarization algorithm, where a node keeps the minimum edge and ignore all the other edges, resulting in a planar sub graph.

However, a node u can have many neighboring nodes in the sectors $v \in s^u_2, s^u_4$ or $v \in s^u_6$. In [Huc et al., 2012] we call the edges (u,v) with v in s^u_1, s^u_3, or s^u_5 *outgoing* edges. With this terminology, a node has at most three outgoing edges and possibly many *ingoing* edges (and edge (u,v) such that $v \in s^u_2, s^u_4$ or $v \in s^u_6$). We emphasize that this is a useful denomination but the graph $\widetilde{G}$ is not oriented.

There are several important properties of the obtained planar graph following the definition 14, which are used in the formulation of routing algorithms through out the rest of this chapter.

Property 1. *A node u has at most one neighboring node in each of s_1^u, s_3^u, s_5^u (the closest with respect to the corresponding order relation).*

This follows immediately from the definition 14. Indeed, if $v \in s_1^u$, $v \in s_3^u$, or $v \in s_5^u$ then $u \in s_4^v$, $u \in s_6^v$, or $u \in s_2^v$ respectively and then, the only possibility for an edge $(u, v) \in \widetilde{E}$ is that v is minimal with respect to $<_1, <_3$ or, $<_5$ respectively. Due to the elimination of edges in the planarization process, empty regions around an edge can be observed.

Property 2. *Given that there is an edge between node u and v and $v \in s_{2i-1}^u$, then there are no other nodes in the region defined by;*

$$s_{2i-1}^u \cap \{z \mid d(A_i, z) > d(A_i, v)\}$$

Proof. By the construction of the planar graph, v is the minimal node in s_{2i-1}^u, with respect to the order relation $<_i$ (defined by $d(A_i, z)$). Therefore to ensure planarity, by definition there is no $w \in s_{2i-1}^u$ such that $d(A_i, w) > d(A_i, v)$ or similarly $w <_i v$ (see figure 4.2(a)). □

An immediate implication of property 2 is an even finer empty region illustrated in figure 4.2(b) as presented in property 3.

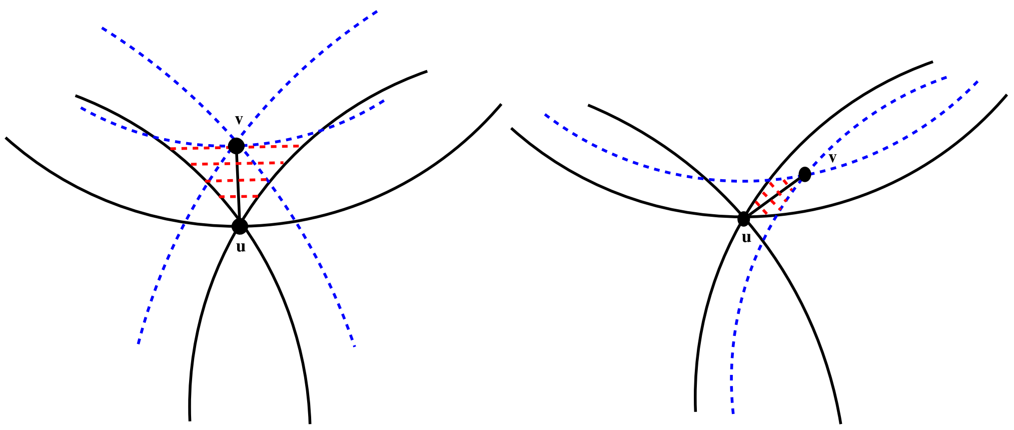
Property 3. *Given the nodes u and v such that $(u, v) \in \widetilde{E}$, then the region defined by $s_v^u \cap s_u^v = \emptyset$*

Proof. Without loss of generality we can assume that $u >_1 v, u >_2 v, u <_3 v$, $v \in s_5^u$ (the proof is the same if we permute the indices). Because u and v are connected it must be that $v = min_3\{z \mid u >_1 z, u >_2 z, u <_3 z\}$. Then sector s_u^v is defined by $\{z \mid z >_1 v, z >_2 v, z <_3 v\}$ and, the intersection is $s_v^u \cap s_u^v = \{z \mid u >_1 z >_1 v, u >_2 z >_2 v, u >_3 z <_3 v\}$ the last inequality shows that if it were a node in the intersection u should be connected to that node instead of v. □

As per these properties, if there is an edge (u, v) in $\widetilde{E}$ and $v \in s_{2i-1}^k$, then there are no nodes $w \in s_{2i-1}^u$ such that $w <_i v$ and $(u, w) \notin E$ i.e., the node w should be connected to u to ensure planarity. A local algorithm has to overcome a pathological situation which occurs due to the radio communication irregularities, where as a node w exists but residing out of the range of u. In such a situation u erroneously keeps a longer edge. Huc et al proved that this situation can be overcome if the planarization algorithm has two hop neighborhood knowledge.

4.4 Combined greedy face routing with delivery guarantees

In this section, a combined greedy face routing algorithm on VRAC system is presented. This algorithm is a variant of the combined greedy-face routing algorithm [Bose et al., 2001].



(a) Illustration of property 2 (b) Illustration of property 3

Figure 4.2 – Empty regions around an edge (u, v)

However, the main difference is that in VRAC it is not possible to compare the angles, hence it is not possible to implement face routing independently of greedy routing (see Proposition 2).

In face routing, basic primitives are the implementation of the left or right hand traversal rule to explore a face of the planar graph and, the detection of the intersection of an edge of the path with the source destination line (see Section 3.3.2). Nevertheless, in VRAC it is not possible to detect line segment intersection. Therefore it is required to consider a (*greedy*) region that contains the source and destination nodes (see equation (4.10)) and to detect when the routing path crosses this region. If the packet is reached a node in the greedy region, it switches back to greedy routing (face routing is no longer implementable in this case, see Proposition 2).

4.4.1 Greedy routing primitives

As mentioned earlier, VRAC does not have an underlying metric. Hence it is not possible to decide the greedy neighbors as in the Euclidean space based on the distance function. Instead, an abstract characterization of a greedy path [Li et al., 2010](see section 3.3.1 for details) is used to define a greedy routing criteria. Accordingly, a greedy path on VRAC is defined as below.

Definition 15. *For destination node D, a path* $\{u^i\}_{i=1,\dots,k}$ *is a greedy path if*

$$u^{i+1} \in s_D^{u^i} \bigcap s_{u^i}^D. \tag{4.2}$$

The region defined by the intersection $s_D^{u^i} \cap s_{u^i}^D$ is considered to be the *greedy region* of the node u^i with respect to the destination D. Moreover, u^{i+1} is said to be greedy for u^i with respect to D. When considering a greedy path established according to the above definition, it ensures the *transitivity* property described in section 3.3. This is presented in proposition 1, which is then used to prove the convergence of a greedy path in corollary 1.

Proposition 1. *(***transitivity***) If u^{i+1} is in the greedy region of u and u^{i+2} is in the greedy region of u^{i+1}, then u^{i+2} is in the greedy region of u.*

Proof. What has to be proven is that, if $u^{i+1} \in s_D^{u^i} \bigcap s_{u^i}^D$ and $u^{i+2} \in s_D^{u^{i+1}} \bigcap s_{u^{i+1}}^D$ then $u^{i+2} \in s_D^{u^i} \bigcap s_{u^i}^D$ For concreteness, we consider $s_D^{u^i} = s_5^{u^i} = \{z \mid z <_1 u^i, z <_2 u^i, z >_3 u^i\}$ and, then $s_{u^i}^D = \{z \mid z >_1 D, z >_2 D, z <_3 D\}$ (we reverse the signs of the inequalities). The assumption $u^{i+1} \in s_D^{u^i} \bigcap s_{u^i}^D$ leads to $D <_1 u^{i+1} <_1 u^i$, $D <_2 u^{i+1} <_2 u^i$, $D >_3 u^{i+1} >_3 u^i$. We then conclude that $s_D^{u^{i+1}} = \{z \mid z <_1 u^{i+1}, z <_2 u^{i+1}, z >_3 u^{i+1}\} \subset s_D^{u^i}$ and that $s_{u^i}^D = s_{u^{i+1}}^D$. This proves the proposition. □

Algorithm 1 contains a pseudo-code of the implementation of the greedy routing primitive.

Algorithm 1 Implementation of the routine *greedy(u,D)*

```
1:  procedure GREEDY($u$,$D$)
2:      Determine the sector $s_D^u$
3:      Determine the sector $s_u^D$                ▷ by reversing the signs of the inequalities
4:      if $\mathcal{N}_u \cap s_D^u \cap s_u^D \neq \emptyset$ then
5:          select arbitrarily $x$ in the set
6:          return x
7:      else
8:          return No Greedy Neighbors
9:      end if
10: end procedure
```

Lemma 1. *Given a destination node D, a greedy path $\{u^i\}$ eventually reaches the destination D.*

Proof. Applying proposition 1 inductively proves that $s_{u^i}^D = s_{u^j}^D$ for all nodes in the greedy path and that $D \in \cap_{i=1\ldots k} s_D^{u^i}$. Because the area of this last intersection decreases, it must eventually hold that the destination D is reached (there cannot be an infinite number of nodes in an infinitesimally small surface). □

Notice that this result follows directly from the results in [Li et al., 2010], since the paths satisfy the required axioms. However, we provide a simple and independent proof on the convergence of a greedy path in corollary 1.

4.4.2 Face routing primitives: Combinatorial approach

In this sub section, a face routing algorithm over VRAC is presented. This algorithm combined with greedy routing leads to a geometric routing algorithm with guaranteed delivery. Let u be the source and D the destination of a packet, then the routing algorithm at u switches to face routing if $\mathcal{N}_u \cap s_D^u \cap s_u^D = \emptyset$. The face routing algorithm selects the node v such that $v \in \mathcal{N}_u$ and the edge (u, v) is the first edge encountered when the line uD is rotated counter-clockwise. The face traversal stops if the path goes through a node belonging to $s_D^u \cap s_u^D$ or, if an edge (v, w) intersects this region. In the former case, greedy routing is restarted, while in the later case it decides to switch the face accordingly (see section 4.4.3). As we implement only the primitives for the left hand traversal rule (see Algorithms 5), face switching is done if the path traverses the region (by selecting w as the next node) inverting the order of the nodes (v, w), i.e. the node v continues the execution of the left hand traversal algorithm by assuming that the data is received from node w.

Notice that our implementation is not an implementation of a classical algorithm like GFG or GPSR [Bose et al., 2001] since it cannot be executed independently of greedy routing for the reasons mentioned in the introduction and substantiated below. We implement the *left hand rule* by determining the first edge encounter in the *counter-clockwise* direction from an edge or the respective empty region (see Algorithm 5). Moreover, we use the face switching strategy used in GFG, which guarantees delivery on an arbitrary planar graph [Frey and Stojmenovic, 2010].

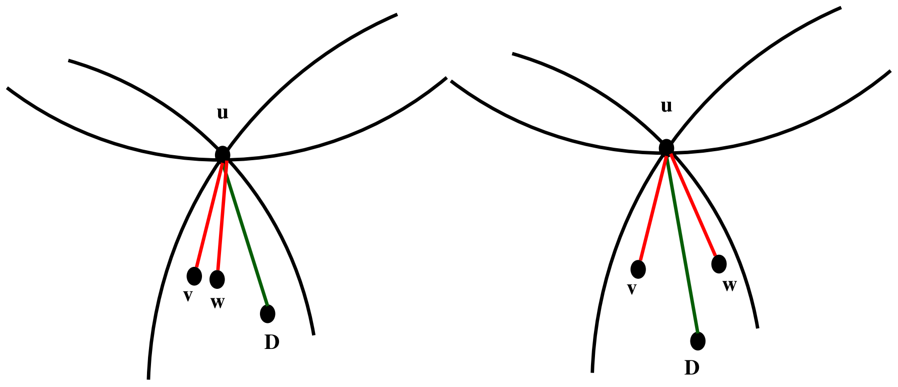
However, our implementation follows the rule of GFG for face switching.

Given an edge (u, v), a first primitive to implement the face traversal is to rotate the edge around u counter-clockwise and to determine the next edge (u, w) that we encounter (see Algorithm 5). Actually, this amounts to find the edge (u, w) that makes the smaller angle with (u, v) where the angle is measured counter-clockwise. In our coordinate system, we cannot compute the angles since we only know the order relations defined by the three anchors.

Similar to greedy routing, face routing primitives are not straight forward to implement over VRAC. In fact, given only the VRAC and initial sectoring, it is not possible to distinguish the edge positions, as required by face routing. For instance, it is not possible to determine the first edge encounters counter-clockwise direction from uD line, if there is no edge between u and D, which is presented in Proposition 2.

Proposition 2. *If the nodes $v, w \in s_D^u$ and, $D \notin \mathcal{N}_u$ while $v, w \in \mathcal{N}_u$ it is not possible on VRAC to determine which edge (u, v) or (u, w) is the next edge to the line uD.*

Proof. Figure 4.3, shows two configurations where $v, w, D \in s_4^u$ and, where the same order



(a) Both edges in the same side of the uD line(b) Two edges in different sides of the uD line

Figure 4.3 – An illustration of the proof of Proposition 2

relations exist between the nodes, i.e. $D <_1 v, w <_1 u$, $D >_2 v, w >_2 u$ and, $D >_3 v, w >_3 u$. In figure 4.3(a) the edge (u, w) is next to the line uD while in figure 4.3(b) (u, v) is next to the line uD. Notice that because of the order conditions u, v, w belong to the same sector of D and D, v, w to the same sector of u, hence the impossibility. □

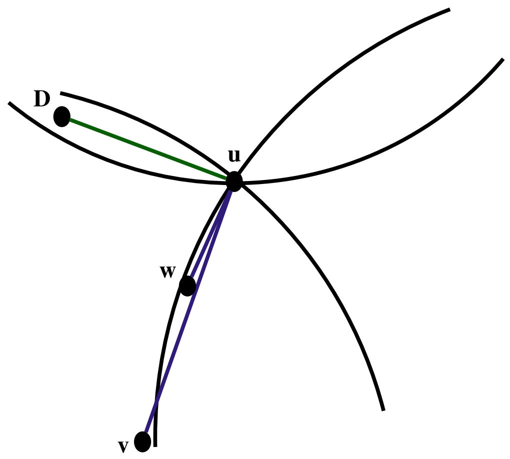
We emphasize that this impossibility is to the fact that uD is not an edge in the graph, but a hypothetical line between the destination and the current node (see figure 3.1(b)). If uD corresponds to an edge, we could investigate the empty region properties (property 2 and property 3) to determine the orientation of an edge around node. In fact, considering these properties, we define an ordering scheme for edges around a given node. First a local ordering criteria is derived within a sector (see proposition 3), which is used to build a global ordering of edges around a node.

Local ordering considers the counter-clockwise ordering and determines the right most node within a sector (more precisely in sectors 2,4 and 6). Proposition 3 presents a procedure to determine the right most edge among two nodes v and w with respect to u, which is denoted as *right(u,v,w)*,

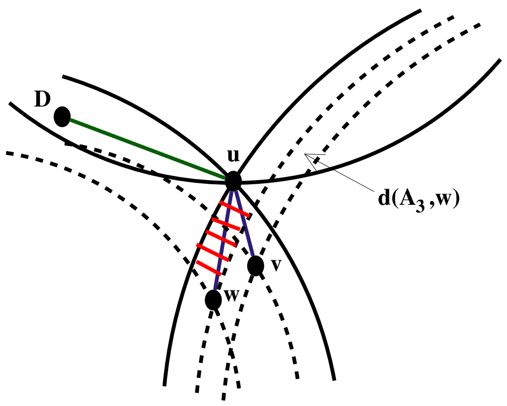
Proposition 3. *Given a node u, let v and w be two nodes such that $(u, v), (u, w) \in \widetilde{E}$ and $s_v^u = s_w^u = s_{2i}^u$ respectively, then the right most edge is determined by;*

$$\textbf{\textit{right}}(u, v, w) = \begin{cases} v & \textit{if}\, d(A_i, v) < d(A_i, w) \\ w & \textit{otherwise} \end{cases} \tag{4.3}$$

Proof. Lets assume that $v, w \in s_4^u$ (proofs for other sectors follow the same argument). Notice that it is not possible that u and v belong to s_1^u, s_3^u or s_5^u, as there can be only one edge in these



(a) Illustration of the impossibility in the proof of Proposition 3, where it is impossible to have a v as in the figure with the presence of w



(b) An illustration that the by comparing $d(A_3, v)$ and $d(A_3, w)$ it is possible to determine the edge that w is on the right of side of v

Figure 4.4 – Illustration of implementation of right function

sectors by Property 1. First we show that the impossibility of an unfavorable setting illustrated in figure 4.4(a). In fact, if w were in the mentioned region then $v \in s_1^w$, like u. But, because $u >_1 v >_1 w$ then w would be connected to v instead of u, leading to a contradiction. In order to prove the main result, it is following argument suffices. As there is an edge (u, w) the node v cannot be in the region highlighted in red lines in figure 4.4(b) (see Property 3). As this holds for v reciprocally, it is possible to determine the edge to the right by comparing $d(A_3, w)$ and $d(A_3, v)$ accordingly. □

An implementation of the procedure *right(u,v,w)* is presented in pseudo code in Algorithm 2. Note that it compares the respective orders in sectors 2,4 and 6 accordingly.

Once the local ordering in each sector can be determined with the procedure *right(u,v,w)*, it can be used to define a global numbering. It comes as an extension to the initial sector numbers (see definition 12). Extended numbering is a relative scheme, where numbers are assigned to a sector relative to the sector s in two possible settings as follows.

1. s = s_D^u when $(u, D) \notin \widetilde{E}$
2. s = s_v^u, when u receives a message from v

First setting corresponds to the scenario, where it switches from greedy routing to face routing as illustrated in figure 4.5(a). In this setting, since there are no neighboring nodes in the greedy region, we need to consider the first edge encounters in the counter-clockwise direction from the greedy region. Second setting corresponds to the scenario where face routing is

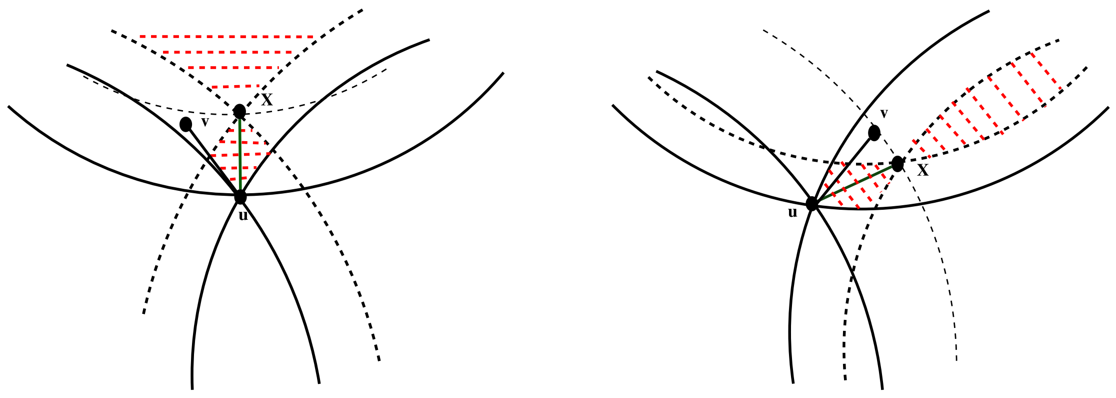
Algorithm 2 Implementation of the routine *right(u,v,w)*

```
procedure RIGHT($u,v,w$)          ▷ $v, w \in \mathcal{N}_u$ and $v, w$ belong to the same sector of $u$
    if $w \in s_2^u$ then
        if $d(A_2, w) < d(A_2, v)$ then return $v$
        else return $w$
        end if
    end if
    if $w \in s_4^u$ then
        if $d(A_3, w) < d(A_3, v)$ then return $v$
        else return $w$
        end if
    end if
    if $w \in s_6^u$ then
        if $d(A_1, w) < d(A_1, v)$ then return $v$;
        else return $w$;
        end if
    end if
end procedure
```

continued as illustrated in figure 4.5(b). Hence, it has to consider the first edge encounters in the counter-clockwise direction with respect to the incoming edge.

We need to define the global numbering such that it determines the first edge situated in the counter-clockwise direction with respect to the above two cases. It is trivial to observe that the initial sector numbers make sure such an ordering (coupled with local ordering), except for the sector s. In sector s, it is possible to distinguish the edges, which are left to the uD line or (u, v) edge due to the empty regions around an edge in the planarized graph (see Propositions 2 and Proposition 3). Following proposition determines (denoted as $left(u, X, v)$) if a given edge $(u, v) \in \widetilde{E}$ is in the left (or right) of the line drawn between u and X. Note that this is a generalization of the above two settings, hence (u, X) may or may not correspond to an edge. In both these cases, the condition $s_X^u \cap s_u^X = \emptyset$ holds. When the face routing starts this is true as the greedy region is empty, where as on the other situation it holds due to the empty region property 3 (see figure 4.5).

Proposition 4. *Let $u, X \in V(G)$ and $j = s_X^u$ such that $s_X^u \cap s_u^X = \emptyset$, left returns true if uv is on*



(a) Starting face routing when $D \in s^u_i$ such that i=1,3,5, i.e $s^u_X \cap s^X_u = \emptyset$, v is on the left of uX

(b) Starting or continuing face routing when either $D\ or\ v \in s^u_i$ such that i=2,4,6, v is on the left of uX

Figure 4.5 – Illustration of two settings considered in left function

the left of uX as below;

$$\textbf{\textit{left}}(u,X,v) = \begin{cases} \text{TRUE} & if(X <_2 v \,\&j = 1) \\ \text{TRUE} & if(X >_1 v \,\&j = 2) \\ \text{TRUE} & if(X <_3 v \,\&j = 3) \\ \text{TRUE} & if(X >_2 v \,\&j = 4) \\ \text{TRUE} & if(X <_1 v \,\&j = 5) \\ \text{TRUE} & if(X >_3 v \,\&j = 6) \\ \text{FALSE} & otherwise \end{cases} \tag{4.4}$$

Proof. Consider the empty regions imposed by the properties of the planar graph as illustrated in figure 4.6. Let j=1, by observation it is clear that a node v to the left of uX follows $X <_2 v$. Similarly when j=2, a node v to the left of uX follows $X >_1 v$ and when j=3 it follows $X <_3 v$. Sectors 4,5 and 6 follows the opposite relation. □

It is important to note that, left function is closer to the right function. Both are defined based on the empty region property and uses a single coordinate to determine the node to the left or right. It should be distinguished that right is only defined for even numbered sectors, where as left is defined for all the sectors. Moreover, left function generalizes the computation, such that it determines if an edge is on the left of a line between two nodes (subject to the empty intersection property or an additional emptiness assumption). Nevertheless, in left function, it is possible to use right function in even numbered sectors, but for the clarity of the implementation we made them independent.

An implementation of procedure left(u, X, v) is presented in Algorithm 3. By utilizing left, now we define the extended sector numbering, denoted as num(u, X, w) as follows.

Algorithm 3 Implementation of the routine *left(u,X,v)*

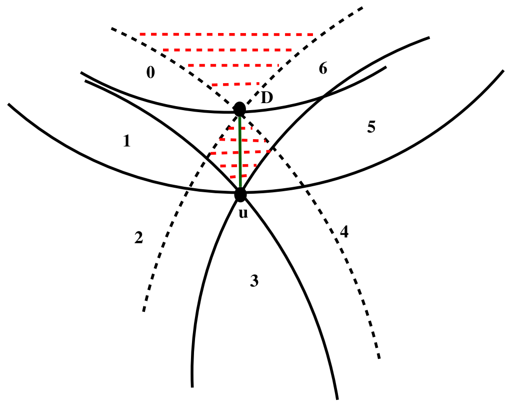
```
procedure LEFT($u$,$X$,$v$)
    $j = s_u^X$
    if $j = 1$ & $X <_2 v$ then return TRUE
    else return FALSE
    end if
    if $j = 2$ & $X >_1 v$ then return TRUE
    else return FALSE
    end if
    if $j = 3$ & $X <_3 v$ then return TRUE
    else return FALSE
    end if
    if $j = 4$ & $X >_2 v$ then return TRUE
    else return FALSE
    end if
    if $j = 5$ & $X <_1 v$ then return TRUE
    else return FALSE
    end if
    if $j = 6$ & $X >_3 v$ then return TRUE
    else return FALSE
    end if
end procedure
```

Definition 16. *Let X be either the destination D or the node v of the incoming edge (see Figure 4.6). Let k be the sector s_w^u of a given node w as defined in Definition 12 and j be the sector number of s_X^u, such that $s_X^u \cap s_u^X = \emptyset$. Extended sector number of w is defined as;*

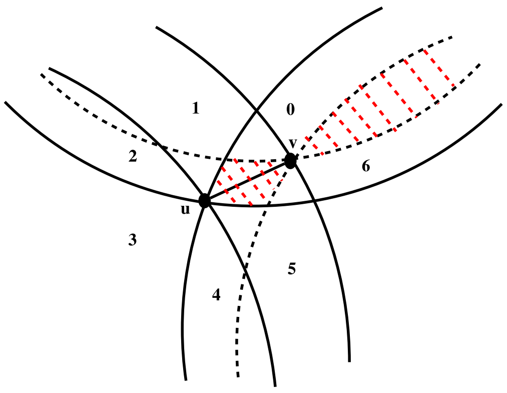
$$\boldsymbol{num}(u,X,w) = \begin{cases} (k+6-j)\ mod\ 6 & if\ k \neq j \\ 0 & if\ k = j\ \&\ left(u,X,w) = \text{TRUE} \\ 6 & if\ k = j\ \&\ left(u,X,w) \neq \text{TRUE} \end{cases} \tag{4.5}$$

Implementation of this function is done by first determining the number k and j according to the definition 12. Note that the nodes on the left of uX line is numbered as 0, while the node on the right of the uX line is numbered as 6 (excluding the empty regions).

Algorithm 4 is then used to implement the next edge function, presented in Algorithm 5. In the next edge procedure, extended numbering and local ordering of nodes within a sector is used to determine the next edge. In fact, it has to compare each edge around in the neighborhood of



(a) When starting face routing consider the uD line



(b) When continuing face routing consider the incoming edge (u, v)

Figure 4.6 – Possibilities for extended sector numbering, where sector s is further divided into 0 and 6

Algorithm 4 Implementation of the routine *num(u,X,w)*

```
1:  procedure NUM($u$,$X$,$w$)
2:      $k = s_u^w$
3:      $j = s_u^X$
4:      if $k \neq j$ then
5:          return $(k + 6 - j) \mod 6$
6:      else
7:          if LEFT$(u, X, w)$ = TRUE then
8:              return 0
9:          else
10:             return 6
11:         end if
12:     end if
13: end procedure
```

a node in an iterative manner to determine the next edge. Algorithm 4 presents the complete procedure for determining the next edge. Lemma 2, presents the correctness of the next edge procedure. This procedure has to be incorporated into a combined greedy face routing algorithm. Once it is possible to determine the next edge, it is possible to perform the right/left hand rule to traverse around a face in the planar graph. This has to be combined with a face switching strategy to formulate the complete greedy face routing algorithm.

Algorithm 5 Implementation of the routine *nextedge(u,X)*

```
1: procedure NEXTEDGE($u$,$X$)
2:     $v = w \in \mathcal{N}_u$                          ▷ starts with an arbitrary node
3:     for each $w\prime$ in $\mathcal{N}_u \setminus v$ do
4:         if NUM($u, v$)=NUM($u, w\prime$) then
5:             $v$=RIGHT($u, v, w\prime$)
6:         else
7:             if NUM($u, v$)>NUM($u, w\prime$) then
8:                 $v = w\prime$
9:             end if
10:        end if
11:    end for
12:    return v
13: end procedure
```

Lemma 2. *Procedure nextedge in Algorithm 5 determines the edge (u, v) next to the line uD or incoming edge uw.*

Proof. The proof is followed by the correctness of sub routines introduced above. According to the algorithm, it compares two edges at a time and determine which one is the next to the line uX. It keeps the next edge out of the two and iteratively compare all the other edges. Initially $nextedge$ is set arbitrarily from the neighborhood. In line 4, it checks if the two nodes w and $nextedge$ for their extended numbering based on the NUM subroutine. If the numbers are same, it uses the RIGHT subroutine to determine the right most edge. If the numbers are not equal it assigns the $nextedge$ the edge with the least number. Following the proof of the $num(u, X, w)$ function, it returns the edge which is next to the uX line. Therefore, the subroutine nextedge always returns the first edge to the counter-clockwise direction. □

4.4.3 Face switching

In order to formulate the complete algorithm, it is required to determine when to switch the face. According to the proof that GFG delivers data with certainty for any planar graph given in [Bose et al., 2001], when an edge (v, w) of face routing cuts the source destination line uD, face traversal must change the traversed face if the line wD is on the right of the edge (w, v). That

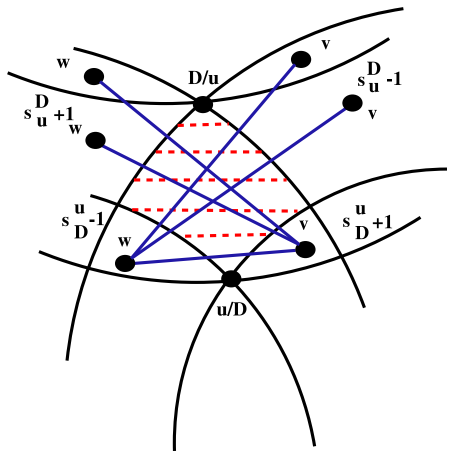


Figure 4.7 – There are 8 edges that can be traversed in both directions and, by choosing the two possibility for source u and destination D there are 16 different configurations lead to *Face Switching*. Note that only 5 edges are illustrated for the clarity

means that the angle between (w, v) and the line wD measured counter-clockwise is larger than π, i.e. we need to rotate the line (w, v) for an angle larger than π to match the line uD.

In our case, we cannot detect the intersection of the line uD. What we detect is that the edge (v, w) crosses the region $s_D^u \cap s_u^D$. By direct inspection and using the *Property* 2, we conclude that a crossing that triggers a face switching occurs if and only if (the arrow $\rightarrow$ indicates the direction of the data, and the number of the sectors are all mod 6, i.e.s_D^u+1 means $s_D^u+1 (\text{mod } 6)$). Note that the sector numbering used here is the original sectoring, as we do not need to use the extended numbering to detect crossing edges.

$$\begin{aligned}
&s_D^u+1 \rightarrow s_D^u-1, \text{ or } s_u^D+1, \text{ or } s_u^D+2\\
&s_u^D-2 \rightarrow s_D^u-1\\
&s_D^u+2 \rightarrow s_u^D+1\\
&s_u^D-1 \rightarrow s_u^D+1, \text{ or } s_D^u-1, \text{ or } s_D^u-2
\end{aligned} \tag{4.6}$$

We represent on Figure 4.7 the edges that can cross the region $s_D^u \cap s_u^D$. A full proof of this result proceeds by inspection. Among all the edges that cross the sector $s_D^u \cap s_u^D$ we exclude some of them by using the Property 1 illustrated in Figure 4.1(a). The only edges that remain and that lead to *Face Switching* following [Bose et al., 2001] are listed in Equation (4.6).

The guaranteed delivery routing algorithm that we present in Algorithm 6 combines *greedy* and *face* routing in the spirit of classical greedy-face routing algorithm [Karp and Kung, 2000;

Algorithm 6 Implementation of the routine *Route(u,v,D,GREEDY)*

```
procedure ROUTE($u$,$v$,$D$,$mode$)        ▷ $u$ current, $v$ previous $mode = GREEDY, FACE$
    if mode=GREEDY then
        $next$=GREEDY(u,D)
        if $next = NULL$ then
            return NEXTEDGE($u$,$v$)
        else
            return $next$
        end if
    end if
    if mode=FACE then
        $next$=GREEDY(u,D)
        if $next = NULL$ then
            return NEXTEDGE($u$,$v$)
        else
            return $next$
        end if
    end if
end procedure
```

Bose et al., 2001] and, implements the switching face algorithm from [Bose et al., 2001].

4.5 Face routing primitives : Geometric approach

In this section we present a routing algorithm [Samarasinghe and Leone, 2012] which guarantees delivery on the second version of the VRAC system described in Section 4.2. This approach uses geometric properties of the coordinate system to define the required geometric constructs to perform geographic routing. In fact, unlike in the previous approach, in this coordinate system we are able to perform rotation and detection of line segment intersection as in Euclidean space. Here we summarize the basics of this approach while a complete description of the algorithm can be found in [Samarasinghe and Leone, 2012]

As illustrated in Figure 4.1(b), a node divides the physical space into six sectors based on its coordinate value. Sectors are numbered according to the Figure 4.1(b), and two of the components of the coordinates happen to be the borders of the sector. Drawing a similar analogy with the Euclidean space, we treat those two components as axes. Note that these axes are relative to a given sector, but the geometric properties are same in all sectors due to the symmetry. By observation, we derive some geometric properties useful for face routing, which is presented as Proposition 5 summarizing the results in [Samarasinghe and Leone, 2012].

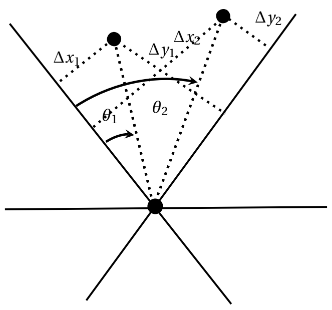


Figure 4.8 – Angle comparison

Consider any sector s of a given node $u = (x_u, y_u, z_u)$ and a neighboring nodes $v = (x_v, y_v, z_v)$ and $w = (x_w, y_w, z_w)$ lying in the same sector. We observe that along a line segment in a sector, at any point on the line it preserves the ratio between perpendicular distances from two sector borders. Let θ_1 and θ_2 be the rising angles from one of the borders of the sector as in Figure 4.8 and $\Delta x_1 = |x_u - x_v|$, $\Delta y_1 = |y_u - y_v|$, $\Delta x_2 = |x_u - x_w|$, $\Delta y_2 = |y_u - y_w|$ are the perpendicular distances from respective borders to the nodes.

Proposition 5. *If* $\theta_1 > \theta_2$ *then* $\frac{\Delta x_1}{\Delta y_1} > \frac{\Delta x_2}{\Delta y_2}$.

Proof. We know that if $\theta_1 > \theta_2$ then $\tan\theta_1 > \tan\theta_2$ when $0 < \theta_1, \theta_2 < \frac{\pi}{2}$. With basic trigonometric relationships we can derive $\tan\theta_1 = \frac{\Delta x_1 \sin\alpha^2}{(\Delta y_1 + \Delta x_1)\cos\alpha}$ and $\tan\theta_2 = \frac{\Delta x_2 \sin\alpha^2}{(\Delta y_2 + \Delta x_2)\cos\alpha}$, where α is the angle of the sector. With a simple algebraic simplification it follows that $\frac{\Delta x_1}{\Delta y_1} > \frac{\Delta x_2}{\Delta y_2}$. □

We use Proposition 5 to define a subroutine to decide the edge with smallest rising angle which is illustrated in pseudo code.

Algorithm 7 Comparison of angles within a sector

```
procedure RIGHT($u$,$v$,$w$)                    ▷ $v, w \in \mathcal{N}_u$
    if $\frac{\Delta x_1}{\Delta y_1} > \frac{\Delta x_2}{\Delta y_2}$ then
        return $v$
    else
        return $w$
    end if
end procedure
```

Using the subroutine 7 and sector numbers we define the Algorithm 8 to find the first edge towards clockwise/counter-clockwise direction. It starts its search from a given sector and finds the edge with smallest rising angle in that sector. If there are no edges in the sector, it continues the search through the following sectors according to their numbering. Hence apparently it searches the whole space around a given node.

Algorithm 8 Rotation clockwise/counter-clockwise

```
procedure FACENEXTEDGE($u$,$D$)
    $i = s^u_D$
    for $i = 1 \rightarrow 6$ do
        if $\mathcal{N}^i_u \neq \emptyset$ then                    ▷ $\mathcal{N}^i_u = \mathcal{N}_u$ in sector $i$
            $right$=RIGHT($v, w$)                                      ▷ $v, w \in \mathcal{N}^i_u$
            for $x \in \mathcal{N}^i_u \setminus v, w$ do
                $right$=RIGHT($x, right$)
            end for
            return $right$
        end if
    end for
end procedure
```

Algorithm 9 Check if two points are in the same side against a line

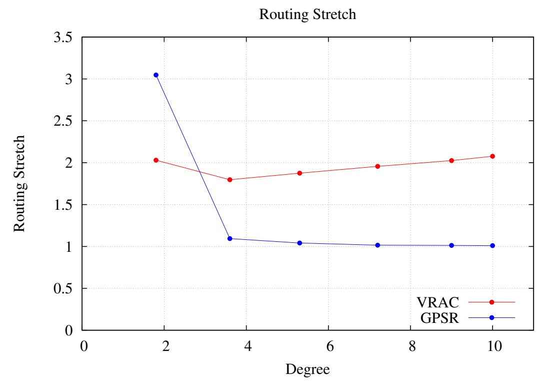
```
procedure ISINSAMESIDE($u$,$v$,$w$,$x$)
    if $\mathcal{N}^i_u \neq \emptyset$ then
        $i = S^w_u$
        $j = S^x_u$
        if $(i \neq j) \vee (j \neq i + 2)$ then
            return $(|i - j| >= 2)$
        end if
    end if
end procedure
```

In order to perform face routing, we need to detect when a possible next edge intersects with the line segment between two other points. Due to the geometric properties of lines, it follows that two line segments get intersected, when the end points of the line segments do not mutually locate in the same side of the other line segment. By inspection of sector arrangement and the possibilities to satisfy this requirement, we define Algorithm 10 to check intersection of two line segments. This algorithm essentially compares the sector positions of two other points w, x compared to a selected point v with respective to point u. It repeats this process for the other combination of points and check whether it satisfy the above mentioned requirement.

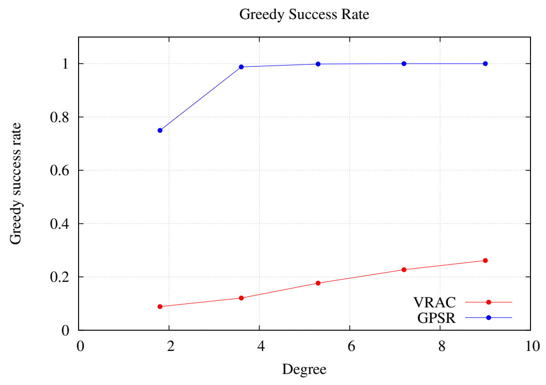
Face traversal performed with the classical right/left hand rule, where a node rotates according to the rule and finds the first neighbor clockwise or counter clockwise. In addition to the face traversal, face changes should be performed accordingly. While there are several variants of the face changing criteria, all of them checks whether whether the face traversal intersects with the connecting line between current local-minima and the destination. We use the defined geometric concepts to implement the face routing algorithm proposed in GFG, which is proven its delivery grantees in an arbitrary graph. Face routing algorithm is illustrated in Algorithm

Algorithm 10 Check intersection based on the sector numbers and angle comparison

```
procedure IsIntersecting($u$,$v$,$w$,$x$)
    if IsInSameSide($u, v, w, x$) then
        if IsInSameSide($w, x, u, v$) then
            return TRUE
        else
            return FALSE
        end if
    elsereturn FALSE
    end if
end procedure
```



(a) Routing stretch



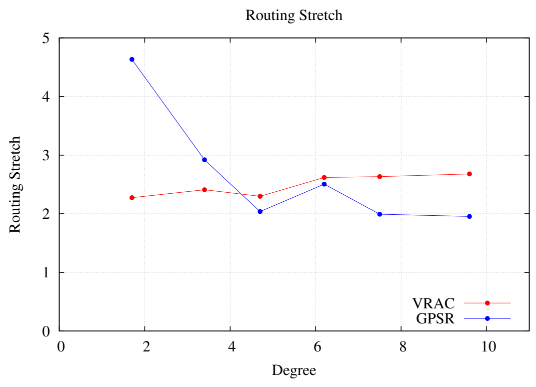
(b) Greedy success rate

Figure 4.9 – Routing stretch in hop distance and greedy success rate without obstacles

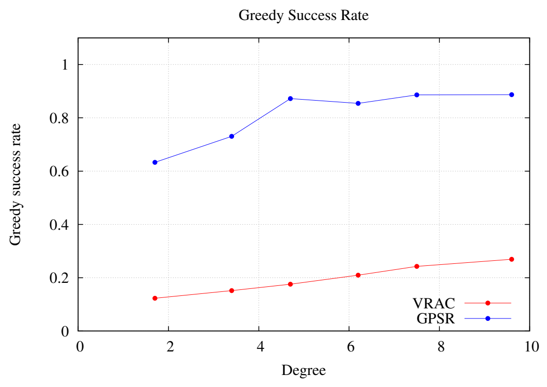
4. It uses the rotation algorithm to search its neighbors clockwise or counter-clockwise and check for intersections based on the Algorithm 3. If it detects an intersection, following the GFG algorithm it changes the rotation direction accordingly.

4.5.1 Numerical Validation

We evaluate our geographic routing algorithm in comparison with GPSR protocol. Evaluation is done in a simulation environment developed in Java, which purely focuses on routing algorithms, while ideal radio characteristics and link layer complexities are abstracted. We extract a planar sub graph of the communication graph according to the schnyder's criteria explained earlier. For GPSR, underlying planar subgraph is a gabrial graph [Gabriel and Sokal, 1969]. We analyze the *stretch factor* and *greedy success ratio*, both in hop distance and Euclidean distance by varying the average degree of a node. We simulate two settings of networks, one with a random distribution of nodes and the other with random obstacles. Nodes are distributed over a 1000 x 1000 square unit area, with 50 units of radio coverage.

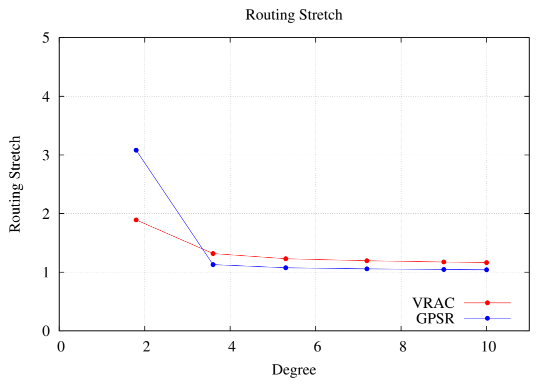


(a) Routing stretch

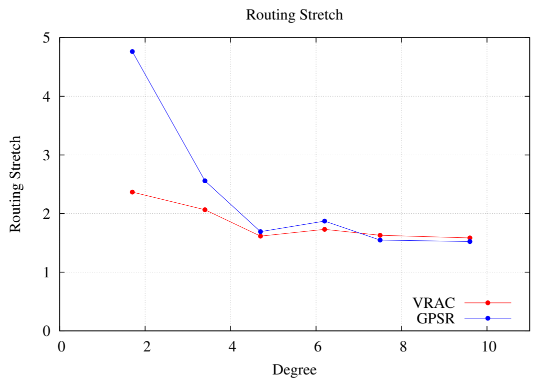


(b) Greedy Success Rate

Figure 4.10 – Routing Stretch in hop distance and Greedy Success Rate **with Obstacles**



(a) Without Obstacles



(b) With Obstacles

Figure 4.11 – Routing Stretch in Euclidean Distance

According to the Figure 4.9(a), VRAC performs better in stretch, when the network is sparser (degree 2). In this instance greedy success rate of GPSR reports the lowest compared to denser instances. Thus it has to perform face routing more often, apparently resulting in higher stretch. Comparatively, VRAC exhibits a poor greedy success (Figure 4.9(b)) even with denser networks (roughly 8% to 22%), yet due to efficient face routing stretch is maintained lower (below 2.3 in all instances).

When the obstacles are present, VRAC reports lower stretch even for denser networks(Figure 4.10(a)), compared to GPSR. This is due to the poor greedy success of GPSR due to the obstacles, see Figure 4.10(b) and comparatively inefficient face routing. Overall, when obstacles are present, stretch performance are closer in both approaches. Finally we analyze the stretch factor in terms of Euclidean distance, see Figures 4.11(a) and 4.11(b). More importantly, stretch in both Euclidean and hop distances for VRAC exhibit a lower variance across sparser and denser network instances. We emphasize that, VRAC does not perform costly computations, hence reducing the localization overhead. This leads to an effective design trade-off for network design.

4.6 Greedy Routing over Virtual Raw Anchor Coordinates

While combined greedy face routing guarantees delivery, it is important to investigate the conditions on VRAC, where greedy routing with delivery guarantees can be performed. In this section, we study a special case of a schnyder graph, namely a *saturated graph* and formulate a greedy routing algorithm.

4.6.1 Schnyder Characterization and Saturated Graph

Let $G = (V, E)$ represent the communication graph, where V, E are the vertex and edge sets respectively. We investigate a network in a standard VRAC setting, hence all the properties are as described in the section 4.2. Considering a planar sub graph extracted according to the schnyder properties, a node u has at most one edge $(u, v) \in E$ such that $v \in s^u_{2i-1}$. Moreover, such a node v satisfies that $v <_i z \; \forall z \in s^u_{2i-1}$, i.e. $v = min_i\{z \mid z \in s^u_{2i-1}\}$. To develop the argument for a greedy algorithm, we rewrite the order relations in each sector as in proposition 6.

Proposition 6.

$$\textit{If}\, \mathbf{v} \in \mathbf{s^u_1},\ \mathbf{z} \neq \mathbf{v} \textit{ we have } \left.\begin{array}{l} z <_2 u \\ z <_3 u \end{array}\right\} \Rightarrow z >_1 v. \tag{4.7}$$

$$\textit{If}\, \mathbf{v} \in \mathbf{s^u_3},\ \mathbf{z} \neq \mathbf{v} \textit{ we have } \left.\begin{array}{l} z <_1 u \\ z <_3 u \end{array}\right\} \Rightarrow z >_2 v. \tag{4.8}$$

$$\textit{If}\, \mathbf{v} \in \mathbf{s^u_5},\ \mathbf{z} \neq \mathbf{v} \textit{ we have } \left.\begin{array}{l} z <_1 u \\ z <_2 u \end{array}\right\} \Rightarrow z >_3 v. \tag{4.9}$$

As described in section 4.3, it is possible to perform a local planarization algorithm to obtain a planar sub graph of $G(V, E)$. An immediate saturation condition on such a planar sub graph, leads to the following definition.

Definition 17. ***Saturated graph****[Leone and Samarasinghe, 2016] A planar graph is saturated if there exists exactly one edge in each sector s^u_{2i-1}, $i = 1, 2, 3$ for each node u.*

The definition holds for planar graphs given in an abstract way. For instance, a maximal planar graph drawn as a Schnyder drawing is a saturated graph (see section 4.7). Indeed, such a planar graph admits a Schnyder representation and the definition refers to this representation. Our greedy routing algorithm is performed on this planar saturated graph and in the rest of the text all references to the graph are implicitly to this graph. We propose a metric free characterization of a greedy path[2] and show that it guarantees delivery when the graph is saturated. We use the combinatorial properties (partial orders) to reason on the delivery guarantees of our algorithm. These combinatorial properties are derived from geometric properties of a saturated graph, yet the greedy path construction is also valid if we assume that the order relation $<_i$ are given in another (abstract) way. This is why in the rest of the paper we avoid direct reference to VRAC coordinate system and make only use of the order relations.

If the (planarized) graph is saturated then each internal node u, has exactly one edge in each sector s_1^u, s_3^u, s_5^u and an indeterminate $(0,1,2,\ldots)$ number in the remaining sectors s_2^u, s_4^u, s_6^u. Since the representation is standard the sectors s_1^u, s_3^u and s_5^u contain the nodes A_1, A_2 and A_3 respectively and are not empty. The maximality assumption implies that if there is an option of adding an edge and keeping the planarity property then the edge is present [Schnyder, 1989].

For routing from a node u to a destination $D \in s_1^u \cup s_3^u \cup s_5^u$ the natural option is to follow the edge (u, v) such that $v \in s_D^u$ $(= s_1^u$ or s_3^u or $s_5^u)$. Next, from v, if $D \in s_1^v \cup s_3^v \cup s_5^v$ we repeat the same strategy. However, it may happen that $D \not\in s_1^v \cup s_3^v \cup s_5^v$, see Proposition 9. In this case $D \in s_2^v \cup s_4^v \cup s_6^v$ and the existence of an edge in the sector s_D^u is not provided by the Schnyder's characterization (4.1). Nevertheless, in Proposition 8 we show that saturation implies the existence of an edge in the sector S_D^u.

Common approach most of the greedy routing algorithms ([Dhandapani, 2010; He and Zhang, 2010]) follow is to compute the greedy embedding given a graph and to use the underlying metric of the respective space to perform greedy routing. However, we avoid the computation of the planar embedding, hence the usage of a metric function for greedy routing. For instance [Angelini et al., 2010] presents an algorithm to compute the greedy embedding of planar triangulations. We rely on the metric-free definition of greedy paths in [Li et al., 2010] without embedding the graph. The coordinate system that we use differs from previous work ([He and Zhang, 2013; Dhandapani, 2010]), that are based on Schnyder's characterization of planar graphs. They use Schnyder drawing as the coordinate system [Schnyder, 1989], which is more complex to compute than VRAC.

4.6.2 Characterization of Greedy Paths

We characterize the greedy path as in section 4.4.1, following the transitivity and odd-symmetry properties.

[2]Given two nodes u and v we don't assume that we can compute the distance $d(u, v)$.

Definition 18. *For destination node D, a path $\{u^k\}$ is a greedy path if there exists $i \in \{1,2,3\}$ such that*

$$\forall k \;\; u^{k+1} <_i u^k, \text{ or } \forall k \;\; u^{k+1} >_i u^k. \tag{4.10}$$

For a greedy path there is a coordinate that changes monotonically.

It is important to distinguish this definition of a greedy path compared to the definition 15. This is a generalized definition as it only considers the existence of a monotonically increasing or decreasing coordinate along the path. In the following, we build greedy paths from u to D such that $u <_i u^k <_i D$ the fact that D is an upper bound and the construction continues while $u^k <_i D$ implies the convergence of the sequence to D. In the proofs of Propositions 8 and 9 we use the assumption that the graph is saturated, to say that given a node u there exists neighboring nodes in the sectors s_1^u, s_3^u, s_5^u. Unfortunately we must proceed with caution if the node u is one of the distinguished nodes A_1, A_2, A_3 since these nodes may not have any neighboring nodes in these sectors. Actually, these nodes do not cause any trouble because there is a path from any internal nodes to them with increasing coordinate $<_i$ respectively. They are also all connected with each other. For these reasons and in order to make our best to simplify the exposition we no longer make any reference to these particular nodes in the proofs.

In the proof of Proposition 8 we use following Proposition.

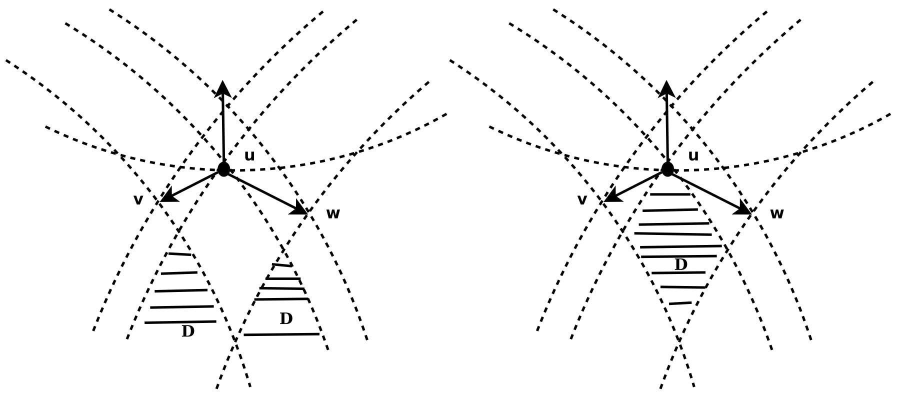
Proposition 7. *If $D' \in s_i^D$ and $D'' \in s_i^{D'}$ then $D'' \in s_i^D$*

Proof. This property follows directly from the transitivity of the inequalities in the definition of the sectors (12). □

Proposition 8. *We assume that the graph G is saturated. Then provided that the destination D belongs to s_2^u (or s_4^u, or s_6^u) then there is a path $\{u^i\}$ in G with $u^0 = u$ such that $u^{i+1} \in s_2^{u^i}$ ($u^{i+1} \in s_4^{u^i}$ or $u^{i+1} \in s_6^{u^i}$ respectively), and the path converges to D.*

Along the path, the order $<_3$ ($<_1$, $<_2$) decreases monotonically if $D \in s_2^u$ ($D \in s_4^u$, $D \in s_6^u$ respectively).

Proof. For concreteness we consider $D \in s_4^u$. If u is connected to D we define $u^1 = D$ and the proposition is true. Otherwise, we prove below that there exists a neighboring node of u, u^1 such that $D \in s_4^{u^1}$ and $D <_1 u^1 <_1 u$. Hence, by applying the construction iteratively we construct the sequence of points that satisfy $u^{i+1} \in s_4^{u^i}$, lower bounded by D and decreases with respect to $<_1$, i.e. $D <_1 u^{i+1} <_1 u^i$. Such a sequence converges to D.



(a) Destination $D \in s^v_{2i}$ or s^w_{2i} (b) Destination $D \in s^u_{2i+1}$ and $u \in s^D_{2i-2}$

Figure 4.12 – Two cases to consider in greedy path construction

Lets prove that given u such that $D \in s^u_4$ there exists x such that $(u, v) \in E$, $D \in s^x_4$ and $D <_1 x <_1 u$. u is internal, by the assumption on saturation, there exists two neighboring nodes of u such that $v \in s^u_3$ and $w \in s^u_5$. we then have

$$D <_1 u,\ D >_2 u,\ D >_3 u\ \Leftrightarrow D \in s^u_4 \tag{4.11}$$

$$v <_1 u,\ v >_2 u,\ v <_3 u\ \Leftrightarrow v \in s^u_3 \tag{4.12}$$

$$w <_1 u,\ w <_2 u,\ w >_3 u\ \Leftrightarrow w \in s^u_5 \tag{4.13}$$

If v (or w) is such that $D \in s^v_4$ (or $D \in s^w_4$) the next point on the path is $u^1 = v$ (or $u^1 = w$) and (4.12) shows that $v = u^1 <_1 u$, and $D \in s^v_4 \Rightarrow v >_1 D$ (or (4.13) shows that $w = u^1 <_1 u$, and $D \in s^w_4 \Rightarrow w >_1 D$).

Otherwise, we have to prove that there exists a neighboring node of u in the sector s^u_4 that satisfies the conditions. We have that $D >_2 u >_2 w$, and $D >_3 u >_3 v$ (using (4.11,4.12,4.13)) and $D \notin s^v_4$ and $D \notin s^w_4$ imply

$$D \notin s^v_4 \Rightarrow \left.\begin{array}{l} D >_1 v\ D <_2 v\ D >_3 v \quad \text{or} \\ D <_1 v\ D <_2 v\ D >_3 v \end{array}\right\} \Rightarrow D <_2 v \tag{4.14}$$

$$D \notin s^w_4 \Rightarrow \left.\begin{array}{l} D >_1 w\ D >_2 w\ D <_3 w \quad \text{or} \\ D <_1 w\ D >_2 w\ D <_3 w \end{array}\right\} \Rightarrow D <_3 w \tag{4.15}$$

Next, because $D \in s^u_4 \Rightarrow u \in s^D_1$ and the assumption of saturation, there exists an edge $(D, D\prime)$ with $D\prime \in s^D_1$. If $D\prime = u$ we are done.

Otherwise, by the property (4.7) and $u \in s^D_1$ we have that $\boxed{D\prime <_1 u}$.

By gathering the inequalities corresponding to $u \in s_1^D$ with the ones deduced from (4.14),(4.15) we obtain $D <_1 D\prime, v >_2 D >_2 D\prime, w >_3 D >_3 D\prime$. Using $D\prime <_1 u$, $D\prime <_2 v$ with property (4.8) we obtain $\boxed{D\prime >_3 u}$.

Last from $D\prime <_1 u$, $D\prime <_3 w$ and property (4.9) (with edge (u, w) instead of (u, v)) we obtain $\boxed{D\prime >_2 u}$. Finally, we have proved that $D\prime \in s_4^u$ with the boxes equations and $D <_1 D\prime <_1 u$. The node $D\prime$ plays the same role as D in the statement of the proposition but with an increasing $<_1$ order position. Because of the bound $D' <_1 u$ we see that by applying iteratively the construction we obtain a sequence $D\prime, D\prime\prime, \ldots$ that converges to u and such that all the points belong to s_4^u. Moreover, along the sequence we have $D\prime \in s_1^D$, $D\prime\prime \in s_1^{D\prime}$,... and Lemma 7 implies that all the points in the sequence belong to s_1^D. In particular, for the point x that is connected to u $x \in s_1^D \Leftrightarrow D \in s_4^x$. We have then proved the existence of a point $x \in s_4^u$ that satisfies $D \in s_4^x$ and such that $D <_1 x <_1 u$. □

Remark 1. Construction of the greedy path if $D \in s_{2i}^u$

In order to route from u to D $\in s_{2i}^u$ *the node u must first check whether for* $v \in s_{2i+1}^u$ *and* $w \in s_{2i-1}^u$ *one of the condition* $D \in s_{2i}^v$ *or* $D \in s_{2i}^w$ *is satisfied and if yes sends the message accordingly. Otherwise, the message is forwarded to (the existing) neighboring node in* $x \in s_{2i}^u$ *such that* $D \in s_{2i}^x$*. This routing scheme converges because the coordinate i decreases along the path and the path doesn't step over D, as all the points in the path are* $>_1 D$.

Proposition 9. *Let us assume that* $(u, v) \in E$ *and* $D, v \in s_1^u$ *(or* s_3^u *or* s_5^u*). Then,* $D \notin s_3^v \cup s_4^v \cup s_5^v$ *(or* $s_1^v \cup s_5^v \cup s_6^v$ *or* $s_1^v \cup s_2^v \cup s_3^v$*).*

Proof. Let us consider $v, D \in_1^u$ the other cases are proved similarly by a permutation of the indices. We have

$$v \in s_1^u \Leftrightarrow u <_1 v \quad u >_2 v \quad u >_3 v$$
$$D \in s_1^u \Leftrightarrow u <_1 D \quad u >_2 D \quad u >_3 D$$

Part b) of the Schnyder's conditions (4.1) implies that D must be larger than u and v for one order and we see on the two inequalities above that it can only be $<_1$. The condition $D \in s_3^v \cup s_4^v \cup s_5^v$ implies that $v >_1 D$ and hence there is no $i \in 1, 2, 3$ such that $u, v <_i D$ and the result in proved. □

Remark 2. Construction of the greedy path if $D \in s_{2i-1}^u$

The practical implication of Proposition 9 for routing is to prove the existence of a greedy path from u to $D \in s_{2i-1}^u$*. We decompose the construction in two parts and for concreteness we consider* $D \in s_1^u$.

Part 1. *The maximality assumption implies the existence of a node $v \in s_1^u$ such that $(u, v) \in E$. If $v = D$ we are done. Else, u sends the message to v and the first coordinate $<_1$ increases, the second one $<_2$ and the third one $<_3$ decrease. If $D \in s_1^v$ then v repeats the same procedure and the coordinates continue to be updated monotonically and $D >_1 v$ because $D \in s_1^v$ and this implies that the first part of the construction converges to D or switches to the second part.*

Part 2. *If the path reaches a node v such that $D \notin s_1^v$ the construction of the path continues with this second part. In this case $D \in s_2^v$ or $D \in s_6^v$ must be satisfied because of Proposition 9. In both cases we have $D >_1 v$ and we can apply Proposition 8 that shows the existence of a sequence of nodes v' with $D \in s_2^{v'}$ or $D \in s_6^{v'}$ respectively and this sequence eventually reaches D. If $D \in s_2^{v'}$ then by Proposition 8 the coordinate $<_3$ continues to decrease along the second part of the construction. If $D \in s_6^{v'}$ the coordinate $<_2$ continues to decrease. In both cases we have shown that along the two parts of the construction one coordinate ($<_2$ or $<_3$) decreases monotonically and the resulting path is then greedy.*

4.6.3 Routing in Maximal Planar Graph

In [Dhandapani, 2010], it is proven that, using Schnyder's characterization of planar graphs (4.1), the existence of an embedding of the graph on the plane[3] such that greedy routing is successful (using the natural metric). In [He and Zhang, 2013] the authors define a similar coordinate system and design a greedy routing algorithm. Both papers use a *realizer* as defined in [Schnyder, 1989].

Theorem 3.

1. *There is a greedy routing algorithm on every saturated planar graph. (saturated version)*
2. *Every Schnyder drawing of a planar triangulation is a greedy embedding*

Our results are summarized in the Theorem 3 in two similar forms, where the proof of the two forms are apparent from the above sections. The pseudo-code of the algorithm is provided in Algorithm 11 and the correctness of the algorithm is proved in the remarks 1 and 2 of the construction of the path if $D \in s_{2i}^u$ or $D \in s_{2i+1}^u$ that follow the Propositions 8 and 9.

4.7 Every Schnyder Drawing is a Greedy Embedding

In this section, we establish some results on the connection between a greedy embeddings and a saturated graph (see section 4.6). We consider a maximal planar graph and its schnyder drawing [Schnyder, 1990], which results in a saturated graph. Then we show that every schnyder drawing admits a greedy embedding, based on the greedy routing algorithm we formulated in section 4.6.

[3]Actually on the plane in $\mathbb{R}^3$ such that $x + y + z = 1$.

Algorithm 11 Pseudo-code of the greedy routing

```
1:  INPUT Source $u$, Destination $D$
2:  repeat
3:      if $D \in \mathcal{N}_u$ then $u = D$                    ▷ $\mathcal{N}_u$ is the set of neighbors of $u$
4:      else
5:          if $D \in s^u_{2i-1}$ then
6:              $u = v \in s^u_{2i-1}$ s.t. $(u,v) \in E$                    ▷ $v$ is unique
7:          else          ▷ $D \in s^u_{2i}$ consider $v \in s^u_{2i-1}$ and $w \in s^u_{2i+1}$ s.t. $(u,v),(u,w) \in E$
8:              if $D \in s^v_{2i}$ then
9:                  $u = v$
10:             else
11:                 if $D \in s^w_{2i}$ then
12:                     $u = w$
13:                 else
14:                     $u = x \in s^u_{2i}$ s.t. $D \in s^x_{2i}$          ▷ must exist by Proposition 8
15:                 end if
16:             end if
17:         end if
18:     end if
19: until u=D
```

[Dhandapani, 2010] showed that every planar triangulated graph can be drawn on the plane as a greedy embedding. They generalize the classical Schnyder drawing [Schnyder, 1990] leading to a family of planar drawings, then they show that there exists a greedy drawing in this set of drawings.

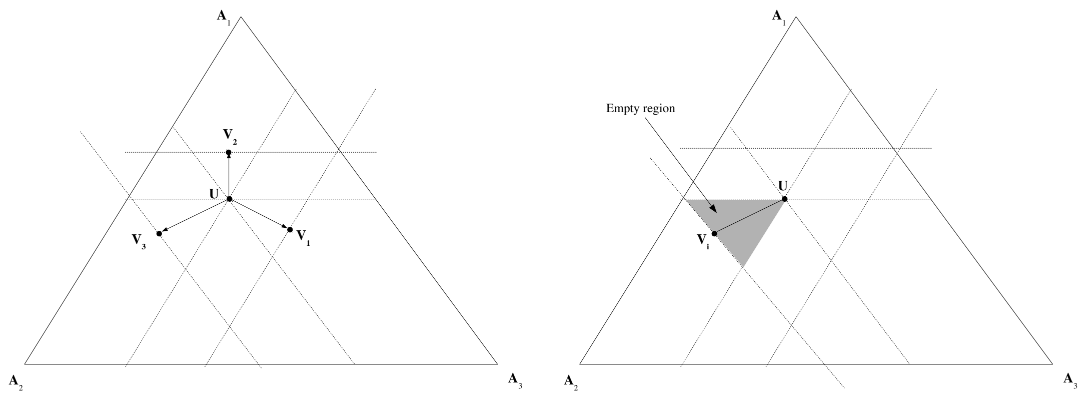
Following result proves that every Schnyder drawing is a greedy embedding. We emphasize the use of a generalized definition of a greedy routing [Li et al., 2010], on which our algorithm is based upon.

4.7.1 Schnyder Drawing

Given a planar triangular graph, a Schnyder drawing [Schnyder, 1990] is a straight line drawing of the graph on the plane. In this article, we consider such a drawing on $\mathbb{R}^3$, such that the external nodes A_1, A_2 and A_3 are placed on $(1,0,0), (0,1,0)$ and $(0,0,1)$ respectively. Hence the external face forms an equilateral triangle and the nodes are placed on the plane designated by $x + y + z = 1$.

The Schnyder drawing is computed based on a combinatorial description of a planar triangular graph, which is called a *realizer*, defined as follows.

Theorem 4. ***Realizer*** *[Schnyder, 1990] Given a plane triangulation $G(V,E)$, there exist three directed edge-disjoint trees, T_1, T_2 and T_3, namely the realizer of G, such that for each inner*



(a) Illustration of Lemma 3: There are exactly three edges in the three regions

(b) Gray region is empty of nodes; Enclosing triangle property (Lemma 6) [Dhandapani, 2010]

Figure 4.13 – Two cases to consider in greedy path construction

vertex u;

1. *u has an outgoing edge in each of* T_1, T_2 *and* T_3
2. *the counter-clockwise order of the edges incident on v is as follows: leaving in* T_1*, entering in* T_3*, leaving in* T_2*, entering* T_1*, leaving in* T_3*, entering in* T_2

The Schnyder planar drawing algorithm [Schnyder, 1990], initially constructs a realizer in linear time. A realizer, in turn leads to three paths from each node towards their root nodes in each tree. These paths partition the nodes into three regions R_i such that $i = 1,2,3$. Let n_i be the total number of nodes in region R_i including the nodes in the two paths, those border the region R_i. Now Schnyder algorithm places each node u on $\frac{1}{n}(n_1, n_2, n_3)$ leading to a planar drawing (see [Nishizeki and Rahman, 2004] for details) [Schnyder, 1990]. [4]

We present two important properties of a Schnyder drawing in Lemma 3 and 4, which are illustrated in Figures 4.13(a) and 4.13(b)[5]

Lemma 3. ***The Three Wedges Property*** *(Lemma 4 [Dhandapani, 2010])*
In every Schnyder drawing the three outgoing edges at an internal vertex v have slopes that fall in the intervals P_1 *in* $[60^o, 120^o]$, P_2 *in* $[180^o, 240^o]$ *and* P_3 *in* $[300^o, 360^o]$, *with exactly one edge in each interval as shown in Figure 4.13(a)*

Schnyder drawing leads to an important void region around and out-going edge as constitutes in Lemma 4

[4]Note that the drawing we consider forms an equilateral triangle, while in [Nishizeki and Rahman, 2004] they present an algorithm where the outer face does not form an equilateral triangle

[5]See [Dhandapani, 2010] for the proof

Lemma 4. ***The enclosing triangle property*** *(Lemma 6 [Dhandapani, 2010])*
Given a vertex u and an outgoing edge (u,v) belonging, without loss of generality to P_1, the equilateral triangle formed by drawing lines with slopes of $0^o, 60^o$ and 120^o, as shown in Figure 4.13(b) is free of any other vertices. This results holds for both regions P_2 and P_3.

In the following section, we introduce the definition of a saturated graph and show that every Schnyder drawing implies a saturated graph.

4.7.2 Schnyder drawings and saturated graphs

A saturated graph is defined without a reference to an embedding. Our greedy routing algorithm in section 4.6, uses the saturated graph property to prove the delivery guarantees. Following lemma constitutes a straight forward connection between a Schnyder drawing and a saturated graph.

Lemma 5. *Every Schnyder drawing implies a saturated graph.*

Proof. Due to the three wedge property of a Schnyder drawing (see Lemma 3), we know that there is exactly one edge (u, v_i) in sectors s_i^u where $i = 1,3,5$, which is a saturated graph by definition. □

In section 4.6, we devised a greedy routing algorithm which guarantees delivery, when the graph is saturated. We consider a schnyder drawing of a maximal planar graph and construct the second variant of the VRAC system (see figure 4.1(b)). This immediately results in a saturated graph, such that each v_i satisfies $v_i = min_i\{w \in s_{2i-1}^u\}$, due to the enclosing triangle property. Therefore, the greedy routing algorithm formulated in section 4.6 can be applied. Note that in this algorithm, we do not use a metric to define the greedy path, instead use a generalized definition of a greedy path. Following lemma concludes the resulting connection between greedy embeddings and a Schnyder drawing of a planar triangular graph.

Lemma 2. *Every Schnyder drawing is a greedy embedding.*

4.8 Conclusion

Geographic routing over virtual coordinate systems has studied extensively as an alternative to real localization systems. Despite numerous proposals in theoretical perspective, they are far from being practical in real network settings. Use of raw distance measures from a set of anchors as the coordinate like in VRAC, posed to be promising mainly due to its simplicity offered in wireless sensor network environments. VRAC does not inherit classical geometric properties like in Euclidean coordinate system. Therefore greedy and face routing is not trivial to perform on VRAC. In this chapter we present combinatorial and geometric approaches to construct basic properties needed to perform both greedy and face routing. Further more we prove that, based on those constructs it can perform delivery guaranteed face routing for arbitrarily connected graphs. We evaluate our approach with GPSR, comparing routing stretch and greedy success rate. Results suggest that VRAC leads to an effective design trade-off for network design.

We introduce the concept of a saturated graph with respect to the VRAC coordinate system and investigate greedy routing over them. We establish that every schnyder drawing is a greedy embedding based on a metric free greedy routing algorithm over the saturated graphs.

5 Robust and Scalable Greedy Routing with Greedy Zone Routing

In this chapter, we introduce a scalable and robust geometric routing protocol called, *Greedy Zone Routing*(GZR). GZR design is based upon the hypothesis that the geometry of a network has to be in an abstract level (zone level) rather than in an individual node level. This ensures resilience towards the dynamic nature of wireless ad-hoc network topologies. Moreover, GZR reduces the length of coordinates, as they are assigned in the zone level.

In section 5.1, main motivation for GZR design is presented, highlighting the drawbacks of state-of-the-art greedy routing protocols on dynamic topologies. Section 5.2 presents the greedy zone routing protocol and its design rational. Section 5.3 presents a comprehensive evaluation of the GZR protocol comparing with PIE [Herzen et al., 2011], a state-of-the-art greedy routing protocol.

5.1 Robust and Scalable Greedy Routing

A key motivation in geometric routing is the scalability, especially to be resilient to network topology changes while maintaining a negligible amount of resources. Despite numerous proposals with sound theoretical background, geometric routing has not thrived to be implemented as a standard protocol. In a more practical perspective, industrial and system research community tend to rely on simpler techniques, when designing protocols. RPL [Winter et al., 2012] is one such example, where a simple tree-based routing (in fact a Destination Oriented Directed Acyclic Graph (DODAG)) is adapted into a standard protocol for routing in low power lossy networks. Routing based on a tree structure is a classical approach for routing employed in various routing protocols, such as in RPL and classical spanning tree protocol (STP). It organizes the nodes into a tree structure whose edges are directed towards the root node, hence each node designating a parent node. Additionally, every node has to learn the set of descendants underneath. Once this information is gathered, a node can decide to route a message upwards or downwards along the tree to reach the desired destination. As the construction of the tree can be done based only on the neighborhood of a node in a distributed fashion, this technique is practically appealing.

Tree-based routing is ubiquitous in standard networking protocols, such as Spanning Tree Protocol and RPL. In this routing paradigm nodes have a list of descendants and the address of their parents. If the desired destination is not among the descendants, the packet is forwarded upwards towards the root. Maintaining a routing tree is efficient as it can be done in a completely decentralized manner. Especially in networks like wireless ad-hoc networks with highly dynamic topologies, routing trees can be a feasible choice for protocol design. Nevertheless, routing on a tree structure can be inefficient when the network size grows resulting in higher routing stretch and bigger routing tables. In fact, a tree route could unnecessarily traverse upwards the tree, while shorter paths exist towards the destination which are unknown to the protocol. A heuristic based approach is proposed in [Duquennoy et al., 2013] to discover the shorter paths in the RPL tree. Nonetheless, this mechanism still maintains a single tree across the network, which could still lead to poor performance when the network size grows.

The practical applications of most of the geographic routing algorithms is hindered by various unrealistic assumptions made in the design. In contrast, Kleinberg's result on the existence of greedy embedding [Kleinberg, 2007] was a major leap forward towards the integration of greedy routing into protocols: not only it is free of unrealistic assumptions, but also it is based on a spanning tree of the network, and hence applicable in arbitrarily connected graphs. Provided its efficiency, tree routing can be incorporated to achieve scalable routing protocols. It has to be used to cover a smaller area of the network, where comparatively higher routing stretch can be traded-off with lower overheads

5.1.1 Scalability

In the context of routing in wireless ad-hoc networks, amount of resources such as memory and communication overhead utilized by a protocol is a crucial factor. A scalable routing protocol suppose to utilize a constant amount of resource, when the network size grows. In other words, even if the network grows in size, resource requirement by the protocol should not grow as rapid as the network growth[1].

5.1.2 Robustness

In a greedy embedding, the geometry has to be constructed based on the connectivity of the communication graph. In [Kleinberg, 2007; Herzen et al., 2011; Westphal and Pei, 2009], a

[1]This is one perspective of scalable routing, mostly from a local routing perspective

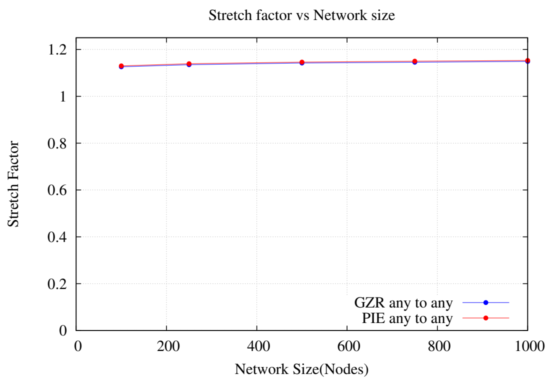
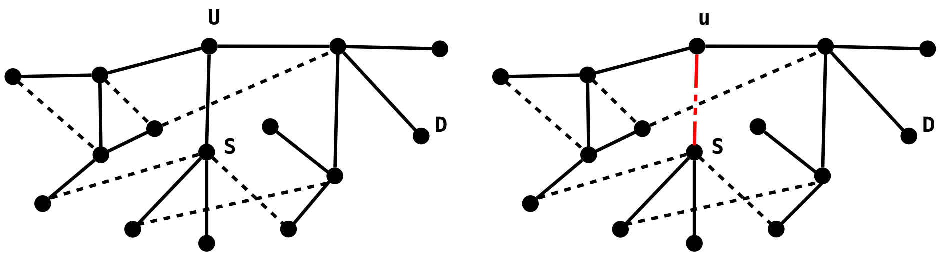


Figure 5.1 – Coordinate length of PIE over the network size



(a) Network connectivity and spanning tree used for embedding

(b) Failure of edge (S, U) introducing a local minima

Figure 5.2 – Pathological example of a greedy embedding when original spanning tree is changed

spanning tree of the network (in this case a sub set of the connectivity) is extracted and the greedy embedding is constructed accordingly. Therefore, once the embedding is constructed, changes in the underlying communication graph (hence the spanning tree) can disturb the greedy properties, which is a major drawback in this approach. Obviously in a practical deployment, coordinates have to be updated periodically. More specifically, the spanning tree has to be reconstructed and coordinates accordingly updated. Figure 5.2, illustrates a pathological example of a greedy embedding, where a *local minimum* results by the change of the underlying communication graph. In Figure 5.2(a), an extracted spanning tree is represented with thick lines, while other existing edges are represented in broken lines. As the greedy embedding is a isometric tree embedding of the spanning tree distances considered for greedy forwarding are similar to the hop distances in the tree. In this particular example, distance between S and D is 3. When the failure of edge (S, U) is considered S no longer has a closer neighbor towards D making S a *local minimum.* This phenomenon occurs irrespective of the degree of a particular node, even at a node with higher number of neighbors.

Considering the state-of-the-art geometric routing proposals, we establish the hypothesis that the geometry of the network has to be maintained as independent as possible from the underlying communication graph. In this way, geometric coordinates can provide a consistent address space as well as the advantages of geometric routing. Based on this hypothesis, we propose to divide the routing process in two levels. The network is partitioned into zones of a fixed diameter and tree-based routing is performed within the zones.

Provided its efficiency, tree routing can be incorporated to achieve scalable routing protocols. It has to be used to cover a smaller area of the network, where comparatively higher routing stretch can be traded-off with lower overheads.

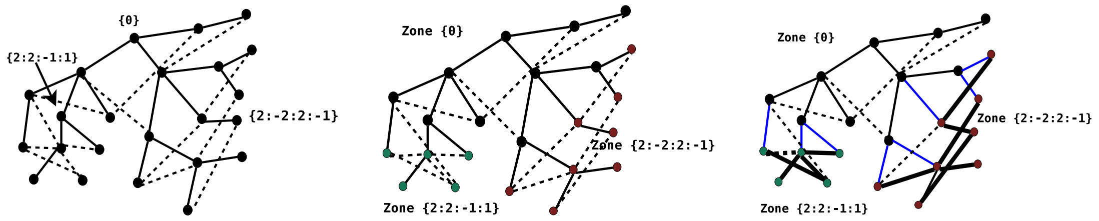
5.1.3 Zone Level Geometry

In this subsection, we present our hypothesis that the geometry of a network has to be established in an abstract level rather than in the individual node level. It allows to maintain a static geometry which is resilient to the changes in the network topology. Such an approach can reduce the expensive communications required to maintain the geometric coordinates in the individual node level. Based on this hypothesis, our protocol divides the network into zones and establish smaller trees spanning each zone, where size of the trees is limited by the diameter of the zones, resulting in constant routing stretch for routes within zones. More importantly, our protocol assigns geometric coordinates to zones, such that geographic greedy routing can be performed between zones reducing the routing scope to a smaller virtual graph whose nodes are zones. Intra-zone routing is done using the internal tree. Our protocol is compared against PIE [Herzen et al., 2011], which is a simple an efficient protocol. It extracts a spanning tree from the given graph and assigns geometric coordinates to the nodes starting with the root. This protocol performs well as long as no frequent updates occur in the network. In this case, the entire tree has to be reconstructed. In our case, the trees are circumscribed to the zones and only trees have to be reconstructed only locally. Besides, the fact that we use no global tree improves the stretching factor, which is an inherent problem in tree routing with large trees.

5.2 Greedy Zone Routing

5.2.1 Overview of Greedy Zone Routing

Greedy Zone Routing (GZR) [Samarasinghe and Leone, 2015] is a two-level routing mechanism over connected subnetworks referred to as zones. Every node belongs exactly to one zone. Nodes having neighbors in a different zone are considered *bordering nodes*. A bordering node maintains a routing tree within its zone advertising its neighboring zone. All nodes in the zone join the routing tree (which is thus a spanning tree within the subnetwork) establishing routes to reach the neighboring zone. This is shown in Figure 5.3.



(a) Connectivity graph and the spanning tree which is embedded greedily

(b) Zone establishment with possible coordinate assignment

(c) Zone tree edges in thick lines and bordering edges in blue

Figure 5.3 – Overview of Greedy Zone Routing

Greedy Zone Routing protocol initially constructs a spanning tree of the network. The set of nodes is partitioned in zones according to a predefined hop distance. In the zone level the network is a set of zones with adjacencies, making up a zone graph. Each zone is addressable by its geometric coordinates. More important, zone coordinates correspond to a greedy embedding (see below.) Routing is performed on two levels depending on the location of the destination node. When the destination node is within the same zone, plain tree routing is used. Otherwise, the message is passed on to the zone which is geometrically closest to the destination.

5.2.2 Greedy Zone Embedding

Greedy Zone Embedding on $\mathbb{R}^2$

In this approach zone embedding is done on the Euclidean plane ($\mathbb{R}^2$) as a delaunay triangulation.

Triangular graphs are a special class of graphs, where every bounded face is a triangle. Note that here we consider the combinatorial description of graphs (vertices and edges), hence no geometric information is present with the graph. A Delaunay realization of a triangular graph is a coordinate assignment of nodes on the plane, such that the Voronoi diagram of the nodes is the dual of the graph. We state the following property on Delaunay triangulation, which makes it suitable for greedy routing.

Property 4. *[Bose and Morin, 2004] Greedy routing always guarantees delivery on Delaunay triangulations.*

Accordingly, if a graph is realized as a Delaunay triangulation, greedy routing is guaranteed to succeed. Although, arbitrary graph may not be Delaunay realizable [Lillis and Pemmaraju, 2008]. Even for realizable graphs, algorithms are iterative in nature and cannot be extended to a distributed algorithm. Therefore we consider the use of a special triangular graph called

Algorithm 12 Greedy Embedding

```
1: Initialization:
2: if $u.id = 0$ then                                              ▷ Root Node
3:     ROOT.x← $x_0$, ROOT.y← $y_0$
4:     Assign $\Delta(ROOT, v, w)$ as in Lemma 3 in [Dillencourt, 1990]
5:     Send NOTIFY to $v(x_1, y_1), w(x_2, y_2)$
6:     Send CONTROL to $v$
7: end if
8: Process:
9: On receive:NOTIFY($x_i, y_i$)
10: Assign the coordinate received
11:
12: On receive: CONTROL at $x$ from $u$
13: find $w|\Delta(x, u, w)$
14: Assign $\Delta(x, u, w)$ as in Lemma 3 in [Dillencourt, 1990]
15: Send NOTIFY to $u, w$
16: Send CONTROL to $w$
```

maximal outer planar graphs, which are Delaunay realizable [Dillencourt, 1990]. In a maximal outer-planar graph, all the inner faces are triangles and every node lies on the outer face. We reorganize the adjacencies of the zone graph as a maximal outer-planar graph according to the Property 5. Note that this process can be performed in a distributed fashion only with zone's neighborhood information.

Property 5. *Let V_0, V_1 and V_2 be three mutually connected nodes. Starting from Δ $V_0V_1V_2$, an iterative process of adding a new node V_i and adding two edges to two nodes in the outer-face of the existing graph result in a maximal outer-planar graph.*

Once the maximal outer-planar graph is constructed, greedy embedding algorithm is performed. As illustrated in pseudo code in Algorithm 12, by assigning the root zone the coordinate $(0,0)$. Then execution follows as a simple graph traversal by message passing. When a node receives the CONTROL message, it is suppose to assign the coordinates its neighboring triangle. This coordinate assignment is based on the Lemma 3 in [Dillencourt, 1990] such that the geometric constraints of a Delaunay triangulation are satisfied. This algorithm terminates when the root node receives a control message and takes $O(n^2)$ message rounds.

Greedy Zone Embedding on $\mathbb{R}^k$

Consider a connected graph $G(V,E)$ and a tree subgraph $T(V,E')$ where $E' \subseteq E$ (i.e., $T(V,E')$ is a spanning tree of $G(V,E)$.) Let $d_T(u,v)$ denote the tree distance between two nodes in hops. An embedding of the tree T is a mapping $\phi : V \to X$, where X is the metric space of interest. This metric space (X,d) is associated with a metric d such as Euclidean metric. An isometric tree embedding is an embedding of a tree in a metric space X, such that $d_T(u,v) = d(\phi(u),\phi(v))$. In other words, it preserves the tree distances between nodes in the embedded space.

The first step is to extract a spanning tree T of the graph $G(V,E)$ and to assign greedy coordinates to the nodes. Once the initial greedy embedding is completed, greedy zones are established. This is accomplished by partitioning the nodes of the network into zones such that each node belong to exactly one zone. Since the zones are subtrees of the spanning tree T, the new graph (whose nodes are zones) has still a tree structure. After that, new border link between zones are sought and the resulting structure is in general no longer a tree.

We use the isometric tree embedding scheme proposed in PIE [Herzen et al., 2011] to perform the greedy zone embedding. Compared to the embedding on hyperbolic spaces, coordinate computation in PIE is efficient and can be done in a distributed fashion. Initially, PIE extracts a spanning tree of the network and embed it on $\mathbb{R}^k$ metric space. The metric function use in this space is the standard l_∞-norm defined as follows;

$$d(x,y) = \|x-y\|_\infty = max_i|x_i - y_i| \text{ for all } x_i \in x, y_i \in y$$

The coordinate assignment of PIE [Herzen et al., 2011] starts by assigning 0 to the root node. Then it recursively assigns the coordinates to the descendants of the tree as follows: the parent node u enumerates its children and produces a binary map b of them. Each child v having a position b_i in the binary map is assigned a coordinate c_v consisting of the concatenation of (1) a prefix, which is the incremented coordinate of the parent, pre(c_u), and (2) a suffix, which is the incremented binary position suf(b_i) of the child. The strings of bits are treated as vectors. The deeper the tree, the higher the dimension of the space in which the coordinates is embedded. The increments are done as follows.

$$\text{pre}(c_u)[j] = \begin{cases} c_u[j]+1 & \text{if } c_u[j] \geq 0 \\ p_i[j]-1 & \text{otherwise} \end{cases}$$

$$\text{suf}(b_i)[j] = \begin{cases} -1 & \text{if } b_i[j] = 0 \\ 1 & \text{otherwise} \end{cases}$$

It is practical to separate the digits of the coordinates, for instance with colons. As an example, if the root has 5 children, the corresponding binary map is (000, 001, 010, 011, 100). Since the root's coordinate is 0, the children receive the prefix 1. Children get the suffixes (-1:-1:-1, -1:-1:1, -1:1:-1, -1:1:1, 1:-1:-1) and the coordinates (1:-1:-1:-1), (1:-1:-1:1), (1:-1:1:-1), (1:-1:1:1), and (1:1:-1:-1). The goal of this assignment is to ensure an isometric embedding. This is shown in Fig. 5.4.

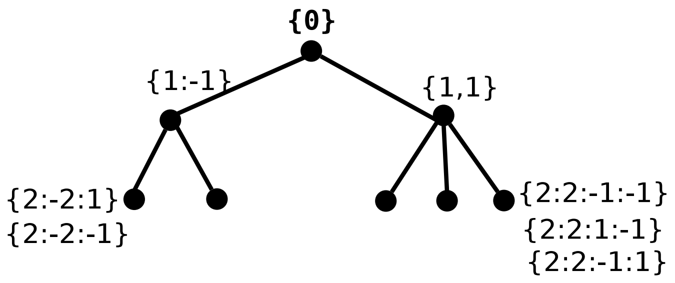


Figure 5.4 – Assigning coordinates to nodes. Prefixes are in black and suffixes in red.

Routing is carried out in two different levels. In the node level (intra-zone routing), tree routing is performed. In the zone level (inter-zone routing) greedy routing is performed based on the coordinates of the zones (zones are assigned the coordinate of the root of the corresponding subtree.) We describe the process in more detail below.

Initial Greedy Coordinates Assignment

Embedding protocols based on spanning trees (as in [Herzen et al., 2011]) initially compute the coordinates based on a particular instance of the network. This makes them sensitive to changes in the topology. One of the main goals of greedy zone routing protocols is to minimize the dependency on a specific topological configuration. Typically they perform embedding for individual nodes and extend this embedding to the zone level. Zones have a better tolerance to topology changes.

Initially we perform the isometric tree embedding explained above. The initial embedding is done over the spanning tree (see Algorithm 1.) This embedding will be extended to the zone level in the next phase. Notice that in practice spanning tree protocols consider several link and MAC layer properties. We adopt here a naive approach.

The process is the following: a pre-defined node initiates the process by assigning itself coordinate {0}. It invites its neighbors to join the spanning tree. If the node has no parent assigned, it will accept the invitation. Once a node determines its parent node, it notifies its membership to it. The process goes on recursively in a distributed way. The coordinates of the spanning tree are embedded in $\mathbb{R}^k$ for some $k \in \mathbb{N}$. The algorithm completes in $O(Diam)$ synchronized rounds of communication where $Diam$ is the diameter of the network. The process is carried out by Algorithm 13.

Algorithm 13 Initial Greedy Embedding

```
1: Initialization:
2: if $u.id = 0$ then                                ▷ Root Node
3:     Assign coordinate 0 to ROOT
4:     Broadcast (TREEAD(ROOT, HOPS=0))
5: end if
6: Process:
7: On receive: (TREEAD(ROOT, HOPS=0))               ▷ Spanning tree construction
8: PARENT← FROM
9: unicast CHILDAD($u.id$) to PARENT
10: On receive: CHILDAD($v.id$)
11: $\{S[i]\} \leftarrow v$                          ▷ Enumerate children
12: Broadcast CHILDENUM$\{S[i]\}$                    ▷ Binary map
13: On receive: CHILDENUM$\{S[i]\}$
14: $c_u \leftarrow$ according to $S[u]$             ▷ Coordinate assignment
```

Greedy Zone Establishment

Once the initial greedy embedding of the network is computed, greedy zones are established. The nodes are partitioned into disjoint zones such that each node belongs to exactly one zone.

Definition 19 (Zones). *A* zone Z_i *of a connected graph $G(V, E)$ is a connected subgraph of G with a pre-defined diameter D in hop distance.*

Since the original graph is a tree (the spanning tree T) it is always possible to partition it into disjoint zones of diameter D. The zones will be subtrees of the original spanning tree. An important concept in our approach is that of *bordering nodes*, defined next.

Definition 20 (Bordering nodes, adjacent zones). *Let $G(V, E)$ be a connected graph partitioned into n disjoint zones Z_i, $1 \leq i \leq n$ such that any node of V belongs to a zone. A node $u \in Z_i$ is a* bordering node *if there is a node $v \in Z_j$ with $i \neq j$ and $(u, v) \in E$. In this case, we say that zones Z_i and Z_j are* adjacent.

Since zones are connected subgraph, any node within a zone may reach an adjacent zone by first reaching the corresponding bordering node. We may think of zones as making up a logical graph, as defined below.

Definition 21 (Zone tree). *Let $G(V, E)$, Z_i as before. The* zone tree $G_Z(V', E')$ *is a graph where $V' = \{Z_i\}_i$ and $(Z_i, Z_j) \in E'$ if Z_i and Z_j are adjacent.*

In the greedy zone establishment phase we extend the isometric tree embedding to the zone level. The nodes form zones based on a pre-defined diameter D. Nodes situated at heights

Algorithm 14 Greedy Zone Embedding

```
 1: Initialization:
 2: Set $D \leftarrow$ diameter of zone
 3: if $d_T(\text{ROOT} = k(\frac{D}{2}+1)$ then                    ▷ Root nodes
 4:     Assign $zc_u \leftarrow c_u$                     ▷ For root notes, $c = zc$
 5:     Broadcast(ZONEAD($zc_u$)) to all children
 6: end if
 7: Process:
 8: On receive:ZONEAD($zc_v$)
 9: if $zc_v < c_u$ and $d_T(v,u) < \frac{D}{2}$ then
10:     Join zone $zc_v$ and assign zone coordinate $zc_u \leftarrow zc_v$
11:     Broadcast(ZONEAD($zc_u$)) to all children
12: end if
```

$k(\frac{D}{2}+1)$ become the roots of the subtrees, which will inherit their coordinates. Each one of the subtrees will end in a bordering node. This is achieved in Algorithm 14, where the coordinates of a node u is denoted by c_u and its corresponding zone coordinates by zc_u

Making some key observations on the zone graph embedding, we present two results in the following.

Proposition 10. *The logical graph of zones has a tree structure.*

Proof. A zone is a subtree of the spanning tree. We must show that between two zones there is a single path. Assume there are two zones Z_i and Z_j such that there are two paths in $G_Z(V',E')$ connecting them. Then, since zones are connected, any node of Z_i may reach any node of Z_j by two different paths. This contradicts the the fact that T is a tree. □

Proposition 11. *The Zone tree $G_Z(V',E')$ admits a greedy embedding.*

Proof. We start from a spanning tree T whose nodes have greedy coordinates (this means, that for each two nodes $u,v \in T$, if u and v are not neighbors, there is a node w such that $d_T(w,v) < d_T(u,v)$. Since the structure is a tree, this property holds if we consider only the roots of the subtrees: either two roots u and v are connected by a path consisting only of non-root nodes or there is a root w in between, i.e., $d_T(u,v) < d_T(w,v)$. Since the coordinates of the roots of the subtrees are the coordinates of the zones, the zone tree admits a greedy embedding. □

As a final remark for this section, note that the length of the addresses in our coordinate assignment is in $\Omega(\log(n))$. In fact the coordinate size grows poly-logarithmically, hence not possible

Algorithm 15 Neighboring Zone Discovery

```
 1: Initialization:
 2: Set $L_u \leftarrow$ list of neighbors $v$ of $u$ with $zc_u \neq zc_v$        ▷ Bordering nodes
 3: if $L_u \neq$ nil then
 4:     Broadcast (JOINTREE($u, L_u$))
 5: end if
 6: Process:
 7: On receive:JOINTREE($v, L_w$)
 8: if $zc_u = zc_v$) and not already joined then
 9:     Append ($v, L_w$) ro routing table
10:     Broadcast (JOINTREE($u, L_w$))
11: end if
```

to guarantee succinct coordinates. On the contrary, such expense can be compromised by the efficiency of coordinate computation in a distributed fashion.

5.2.3 Zone Neighborhood Discovery

The zone neighborhood discovery protocol is used to discover the connectivity of the zone graph. This protocol is initiated by the bordering nodes as follows. Let (u, v) be a bordering edge, i.e., $zc_u \neq zc_v$. Both u and v start to broadcast the coordinates of the neighboring zones in a zone neighborhood notification message inviting the other nodes to join a spanning tree across their respective zones. Nodes in a different zone ignore the broadcast, hence restricting the broadcast to the zone of the initiating node. Upon reception of zone neighborhood notification, nodes update their routing tables with the route to reach the corresponding neighboring zone. The message is rebroadcast so that all the nodes in the zone get the routing information. This process creates a routing tree with a depth of D hops in a distributed fashion. This process is shown in Algorithm 15. Observe that now we are no longer using the spanning tree T but the whole graph.

5.2.4 Routing over Greedy Zones

Routing over greedy zones is done in two levels. Routing between two zones (inter-zone) is done by greedily forwarding the message to the closest zone. Let C_s and C_d be the coordinates of source and destination zones. As mentioned earlier distance function used to compare the closest neighbor is standard l_p norm. Note that these coordinates can have different lengths, but only the common coordinates will be considered. For instance distance between coordinates $2:-2:1:-1$ and $3:1:-1$ is $max|(2-3):(-2-1):(-1-(-1))|$, resulting 3. Once the closest zone towards the destination zone is determined, a node looks up its routing table and forward the message along the respective route towards the zone (this will follow the tree structure rooted at a bordering node).

Once the message reached the destination zone, routing is done based on the routing trees within the zone. Since these trees are spanning over smaller regions, tree routing is comparatively efficient. Also as there are multiple trees spanning a given zone, it can tolerate failures and network dynamics.

Another important aspect of greedy zone routing is the spatial diversity gained by the partitioning of the network. Assume node s in zone Z_i wants to send a message to node t in zone Z_j. Since s knows its neighboring zones and their coordinates, it has the opportunity of routing the message to one of the closer zones Z_k towards the destination zone Z_j (obviously assuming there are multiple zones which are closer towards Z_j). Furthermore, as there can be multiple routing trees towards Z_k, makes further choices for routing. This diversity can be used to device further optimized routing mechanisms, in terms of traffic or energy concerns. We emphasize the distinction of GZR to a classical cluster based routing scheme. GZR benefits the freedom of greedy routing in the zone level without maintaining routing tables. Whereas in cluster based routing, a costly procedure is required to maintain routing tables to route between clusters.

5.3 Evaluation

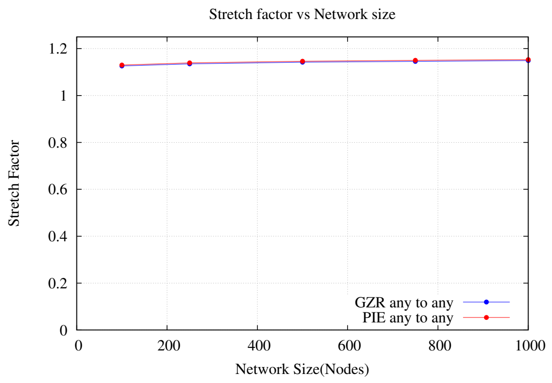
In this section, we present a comprehensive evaluation of greedy zone routing. Greedy Zone Routing protocol is implemented in OMNET++ discrete event based network simulator. We generate random geometric graph graphs of different sizes and GZR is performed. Comparative analysis is carried out against reference implementations of classical tree routing protocol and PIE.

We carry out experiments to validate the hypothesis we established in Section 5.1.2, which was the main design rational of GZR. First, we evaluate classical routing metrics, like stretch factor and packet reception rate for simulated traffic flows. In order to analyze the scalability of the network, network size is varied and metrics are observed. Most importantly, the protocol overhead in terms of control message exchanged per node is evaluated for dynamic network topologies.

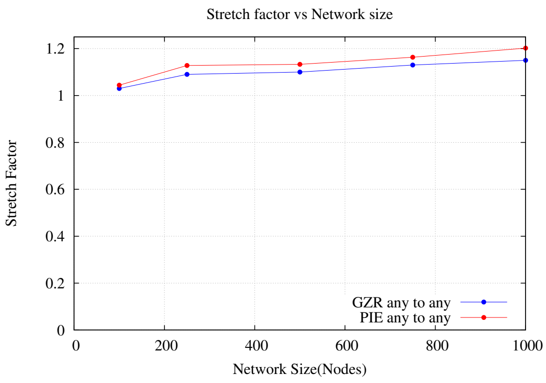
5.3.1 Routing Metrics

Stretch Factor

Stretch factor is the ratio between length of the routing path and the shortest path. It is an important metric, especially to analyze the scalability of routing protocols. We analyze the



(a) Stretch Factor vs Network Size For Upward Traffic



(b) Stretch Factor vs Network Size For Uni-cast Traffic

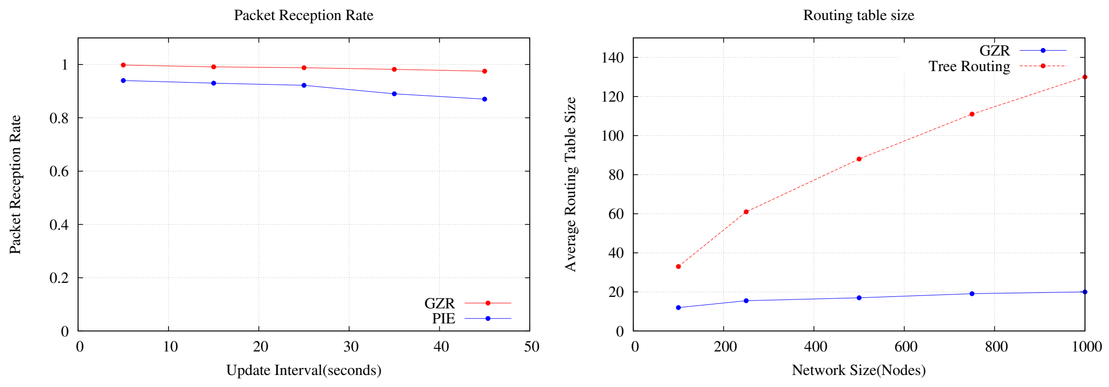
Figure 5.5 – Stretch Factor

routing stretch for two traffic patterns, namely converge-cast and unicast. In converge-cast one designated node is considered as the sink and other nodes generate traffic towards it.

For the converge-cast (upwards routing) scenario, routing tree performs the best as it construct a shortest path towards the sink node. Comparatively both PIE and GZR demonstrate comparatively low routing stretch as illustrated in Figure 5.5(a). It is important to emphasize that tree routing achieves optimum stretch in the expense of very large routing tables (see Figure 5.6(b)). Also in a request response traffic scenario, where both downstream and upstream traffic is operating, tree routing can lead to congestion. This is due to the bottleneck possibly created by following the same route for many nodes. We also performed routing stretch comparison for routing between arbitrary nodes(uni-cast) in the network (Figure 5.5(b)). In this scenario, GZR and PIE both perform exceedingly better compared to tree routing, obviously due to the use of maximum connectivity possible in the greedy route decision making compared to the routing tree (Figure 5.5(b)). GZR performs slightly better as in the zone level routing trees are optimal. Most importantly, routing stretch in both GZR and PIE are almost bounded when the network size grows.

Packet Reception Rate

In the presence of network dynamics, packet reception rate provides a more realistic metric for routing protocols. We simulate uni-cast traffic pattern and analyze its behavior, emulating network topology changing over time. Probabilistic link failures are emulated and protocol update intervals are varied to analyze the behavior. It is obvious that higher communication overheads lead to more reliable packet delivery. Our simulation scenario varies the update interval from 5 to 45 simulation time units and estimated the packet reception in a random any to any traffic scenario. As in Figure 5.6(a), GZR outperforms PIE, yet in PIE also a reasonable



(a) Packet Reception Rate (b) Routing Table Size

Figure 5.6 – Packet Reception Rate and Routing Table Size

packet reception is maintained. As GZR does not recompute the coordinates, packet loss could only occur due to zone level routing unavailabilities, whereas in PIE packets can be dropped by reaching a *local-minimum*. Even more appropriate analysis would be to compare the packet reception rate against the overhead, which is omitted due to space limitations.

Routing Table Size

Greedy Zone Routing maintains a routing table for zone level routing, which is based on spanning trees within zones. Let zone Z_i has d neighboring zones and since GZR maintains m trees to reach a given zone, $m.d$ routing entries occupy the routing table. Compared to tree routing (more specifically the average routing table size of a node) this number is order of magnitude lower (see Figure 5.6(b)). However, in PIE routing table size is even smaller compared to GZR, since it only carries the neighboring nodes and their coordinates. Nevertheless, limiting the routing tree scope to zone level has drastically reduced the routing table size. In fact, Figure 5.6(b) shows an almost constant routing table sizes for different sizes of networks.

5.3.2 Control Overhead

The key distinction of GZR with other proposals including PIE [Herzen et al., 2011; Houthooft et al., 2015] is that, GZR does not require to recompute the coordinates periodically in order to maintain the greedy embedding. Even if some minor changes in the underlying spanning tree occurred, as there is no way to detect this in a distributed setting, whole computation phase has to be executed. Message complexity of tree construction grows with the number of nodes (at least as linearly). Once the zones are established in GZR, trees are spanned only

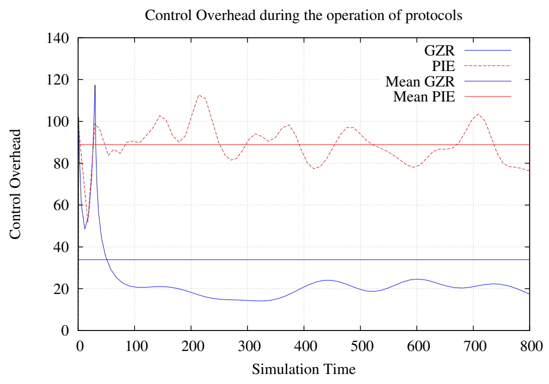


Figure 5.7 – Control overhead during the operation of protocols

within zones with fixed diameter(in hop distance). Therefore, message complexity does not grow with the size of the network. This is the key distinction of GZR as opposed to PIE, which results in the scalability of GZR.

This aspect is very important in practice as this could result in a lot of overhead during the operation of protocol. In order to analyze this behavior in PIE, we introduce a periodic update of coordinate computation and analyzed the control overhead. More specifically, a network of 100 nodes simulated with an update interval of 30 simulation time units. Aggregated number of control packets processed by each individual node within a given time is counted.

Figure 5.7 shows the comparison of control overhead between GZR and PIE. On average a GZR node processes 33 control packets while, PIE processes 89 packets. In the initialization phase of GZR, overhead is similar to PIE as both performs the initial embedding. Once the zones are established, as scope of routing trees are limited within the zone, hence overhead is reduced.

5.4 Conclusion

In this chapter, we propose a new routing protocol for wireless ad-hoc networks called Greedy Zone Routing(GZR), especially focusing on robustness and scalability. The design rational is to use geographic greedy routing in an abstract level, instead of in individual node level. In other words, GZR assigns geographic coordinates to a collection of nodes in a network called a zone. Further more, GZR utilizes classical tree based routing to route messages within zones. Nodes laying closer to zone borders establish paths for other nodes in the zone to reach neighboring zones. Geographic coordinates of zones are assigned such that, greedy forwarding of messages between zones guaranteeing delivery to destination zones.

Greedy Zone Routing overcomes the cumbersome re-computation of greedy coordinates in dynamic network topologies. It assigns the coordinate once to the zones and there after maintains zone level routing trees, greatly reducing the control overhead of the protocol. We evaluate GZR comparing with a state-of-the-art greedy routing protocol and demonstrate that it provides a lower routing stretch and smaller routing table sizes, while maintaining the overheads 50% lower. Future work would investigate the advantage of flexibility offered by the geometric zones in the context of opportunistic routing. Also it is also important to consider greedy zone establishments such that quality of service requirements are achieved.

6 Internet of Things Framework over RPL Routing

In this chapter, we present an application scenario for scalable wireless ad-hoc network routing. We use a state-of-the-art routing protocol for low power lossy networks and develop a service oriented framework, over which Internet of things applications can be developed. This framework is a service oriented framework, which enables various wireless devices to be connected to Internet and expose their services over standard protocols for the web. These web services are implemented over an application layer protocol for energy constrained devices, called constrained object access protocol (COAP) [Shelby et al., 2014]. COAP follows the principles of REST web services [Dunkels et al., 2009], where the devices are characterized based on the resources they expose. For instance, a sensor can have a resources such as *temperature* and *humidity*. Syndesi framework connects various different devices over different physical layer protocols and amalgamate them in a service oriented fashion. Based on syndesi, applications such as smart environments and crowd sensing scenarios can be implemented. We present a personalized smart office scenario in section 6.3 based on the syndesi framework.

6.1 Routing for low power lossy networks: RPL routing

Network protocols tend to adapt simplest mechanisms as possible in their execution. Handful of examples of simple designs can be observed in standardized protocols, which are successfully deployed in real networking settings; Ethernet is one such example. Similarly for wireless ad-hoc networks, simple techniques have adapted when defining protocols. RPL is one such standard protocol for routing in wireless ad-hoc networks, which is based on a simple tree based routing mechanism.

6.2 A service oriented framework for Internet of Things: Syndesi

The aim of the *Syndesi* framework [Evangelatos et al., 2013] is to connect various devices over different technologies into a service oriented framework. Syndesi uses the RPL routing protocol in the IP layer, which is implemented on an embedded operating system named Contiki. RPL is an IPv6 routing protocol, defined for low power lossy networks. RPL construct a tree like structure of the network with a pre-designated root node. In fact, the structure is a direction oriented directed acyclic graph (DODAG), which is not a tree, but directed towards the root node. The root node is connected over to an IPv4/IPv6 network, hence the RPL instance connects to the Internet. IPv6 has evaluated to be efficient for such a scenario in [Evangelatos et al., 2012].

Application protocol deploys on the devices is COAP, which is a REST style application layer protocol designed for low power devices. Syndesi defines all the auxiliary services in COAP to control and monitor devices, which are connected over wireless medium. Also syndesi works as a proxy, enabling COAP services on the devices to be accessed over REST services. In other words, syndesi mediates between COAP and REST services.

6.2.1 Architecture of Syndesi

There are two main components of the Following we describe each component of the framework in detail.

Backbone WSN

The *Backbone WSN* comprises the sensors which are responsible for the monitoring and control of the smart environment. These sensors are capable of communicating over the wireless medium. For the backbone WSN, TelosB sensor nodes are used. The TelosB node is an open source platform designed to enable cutting-edge experimentation. These nodes form a wireless sensor network based on the 6LoWPAN protocol implementation available in the Contiki operating system. Therefore, the network is IPv6 addressable with optimized IP services such as routing for low power wireless links. Some of the WSN's motes are used as actuators taking part also in the *Electrical-Electronic Interface* which is connected to electrical and electronic appliances.

Gateway

The gateway of the *Syndesi* framework is interconnected with the *Backbone WSN*, the *WSN for Occupants' Identification* and the *Web*. It contains the two base-stations of the two different

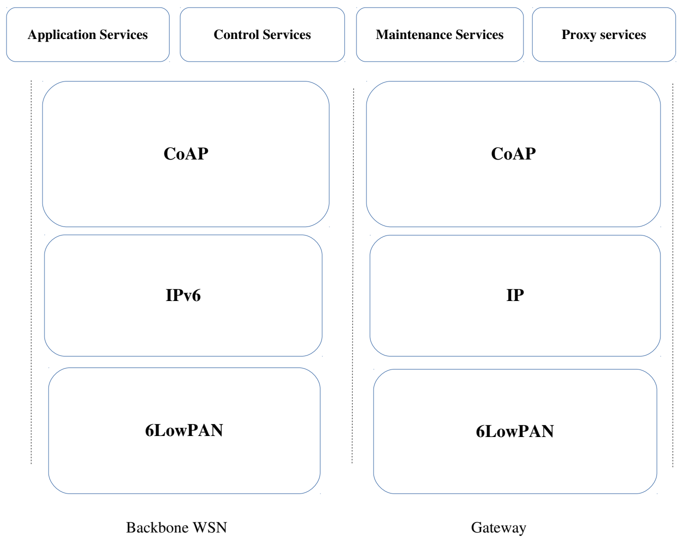


Figure 6.1 – A Layered view of Syndesi Framework

WSNs so as to provide connectivity with them. On the other side it provides a proxy server in order to make the whole framework accessible to the Web connecting both CoAP and HTTP enabled systems.

6.2.2 Layered overview of Syndesi

Interconnection of heterogeneous networks has to be handled by standard messaging and communication protocols. As our framework comprises two different wireless networking technologies (Zigbee and 6LoWPAN), a service oriented architecture can be used to unify the different components. In other words, we define REST style services using COAP in the embedded domain, and mediate them with HTTP services using the proxy service. This mediation takes place in the gateway, which eventually connects both WSN with the web. Therefore it leads to an easy application integration of the framework. We use the light weight JavaScript Object Notation (JSON) as the message format, since it is preferred over XML, especially for the embedded domain applications. Figure 6.1 illustrates the layered overview of our framework across its components. The set of gray boxes illustrates the main services provided by the framework, which are described below.

Actuator Service

Sensors and actuators available in the sensor platform act as an interface between the physical environment and the framework. Such sensors provide readings of the environment parameters via the system software available in the Contiki operating system. Similarly we implemented actuator drivers in Contiki. Services can use these actuator drivers to make them available to the framework.

Control services for the backbone WSN

Control services provide auxiliary services to facilitate the control and monitoring of the application services. For example, these can be services defined to ensure the quality of service requirements of the framework. We define these services in such a way that the overhead added by them is compensated with the quality of service they provide to the system.

Application Services Application services are the services required by specific applications intended to be built upon the framework. In general, these application services expose sensors as services, which are inputs to the framework or expose actuators as services, which control objects of the system. These individual services are combined to perform useful application tasks.

Localized Service Mapping Most of the algorithms performed in smart environment applications need to have a knowledge of the physical locations of the sensors and actuators (nodes of the WSN). Moreover, such algorithms need to be aware of the physical correspondence of the nodes within a context of the considered space. In a smart building scenario nodes have to be mapped with the floor plan of the building. By contrast, IP networks do not maintain any knowledge of the nodes in the network. Therefore, to compensate the fact that IP addressing is location independent, there should be a service discovery mechanism based on the location. This ideally should maintain a mapping between the physical location of a node with its IP address and the services provided.

Due to the lack of low cost and accurate localization protocols implemented in practical settings, we implemented a centralized location mapping service. In other words, we assume a sensor is bound to a specific location and identified by a unique location identifier. We label the nodes of the network with this location identifier when deploying the individual nodes. These labels form a logical tree structure which is rooted at the base station connected to the gateway. The framework maintains a centralized registry of these labels, essentially a tree structure of labels.Maintaining a tree topology is natural mainly since the objects needed to be considered in a smart infrastructure system can be modeled as a hierarchical structure. Especially in a scenario like a smart building system, whole system can be hierarchically modeled as a tree with the root being the central coordinator of the system.

Maintenance Services As most of the smart environment systems are time and safety critical, there must be provisions to ensure such demands. Therefore, we designed additional services to ease the management and monitor the health of the system. We first define a periodic status update service which sends sensor readings and the status of a node to the central registry. In addition, each node exposes a service which can be invoked to check the presence of a node (namely, a "heartbeat" service). Furthermore a node maintains a state of itself and changes its state accordingly.

Proxy Service Both application and framework services are COAP services, which are not possible to invoke over standard Web protocols. Therefore, to expose these services to the Web, we implemented a proxy service, which supports RESTfull interaction over HTTP with COAP service. We define RESTfull services interfaces to utilize both application and framework services, which can be used to develop required applications based on our framework. Due to its compliance with standard Web protocols, these applications are easily portable across all the platforms and devices.

6.3 Proof of Concept

For the proof of concept of *Syndesi* framework, we implemented transform an office environment into a smart personalized environment. The objectives of the application are personalization, improved comfort, safety and energy efficiency.

6.3.1 Description of the environment

The implementation took part in two rooms of an office, where 8 people work. We connected to our framework the existing electrical and electronic devices which are used every day. As it is shown in the 3D representation of our office (Figure 6.2) the electrical and electronic devices connected to the framework are marked with a cross, and the sensors for monitoring the environment with a "star". Following we present the devices connected to our framework in more detail. We connected the 8 personal desk lamps placed on each desk. There are six floor lamps that are placed in between of the desks which produce enough light to cover with sufficient working light the two desks next to it and sufficient ambient light for the area around them. We have placed a fan and a heater in each room to maintain comfortable room temperatures in the office as our infrastructure is old and it has insufficient HVAC system. On top of one of the windows we placed a roll-curtain which we connected to an electrical motor for raising and lowering it. On the inside part of the doors we installed electric locks while on the top of the doors we installed a siren alarm together with an emergency red light.

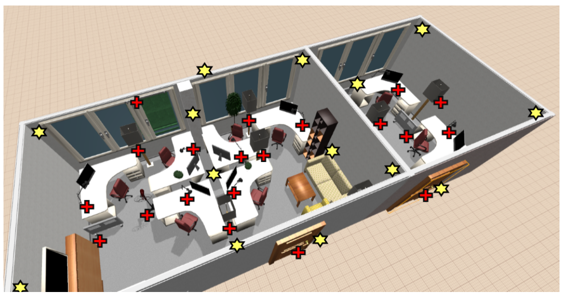


Figure 6.2 – Office Environment of implementation. The cross represents the electrical-electronic interface and the "star" the sensors for monitoring the environment

6.3.2 Description of scenarios

The implemented scenario involved all the 8 person who work in these two rooms. For each person was created a profile account in a database which contains basic personal information about him/her such as: Name, Age, Sex, Profession, Desk Location, preferable Temperature and Lighting condition. Consider now an ordinary day when "Bob" is arriving to his office; he touches with his personal and unique NFC-tag the NFC-sensor on his office door.Then the system identifies his unique ID in the database and it disengages the electrical door lock. At the same time, if the luminosity inside the room is below the default threshold then the system raises the curtains. In the case when the outside light is still below his preferences, then his desk lamp will switch on. The same algorithms apply for the personalized temperature, heating and ventilation control. The above scenario represents an example of a centralized behavior of the framework.

On the other hand, the distributed behavior of the framework is presented in the following scenario: people are working at the office when suddenly a fire breaks out in a lamp in a room. The system then recognizing the extreme high temperature due to the fire, automatically cuts off the power to this lamp and triggers immediately the alarm system. In that case the messages generated by the sensor are transmitted directly to the actuators and afterwards the gateway will be updated concerning the status of the environment.

6.4 Conclusion

In this chapter, we present a service oriented framework for Internet of Things applications. It allows various devices to connect over standard protocols like REST and COAP and provide a unified framework to access and control. In the IP layer, a protocol called RPL is employed, which is a standardized protocol for low power lossy networks. We demonstrate an IOT scenario implementation over syndesi. Further work on this direction would investigate the scalability of such applications over RPL routing. An ideal extension is to implement a geometric routing scheme like greedy zone routing [Samarasinghe and Leone, 2015] such that the network is partitioned into zone.

7 Conclusion

Geometric routing is a promising approach for routing in wireless ad-hoc networks. It avoids building routing tables, instead uses the geometric coordinates to perform routing. Hence it is considered to be a scalable routing scheme for networks with dynamic topologies. Nevertheless, geometric routing has not been deployed in real networks, due to two main practical challenges. Firstly, it requires the nodes to be equipped with geometric coordinates. Secondly, the fundamental algorithms (Face routing) require to have a planar sub graph of the underlying connectivity graph. But by the time of this writing, there is no algorithm to perform the localization, unless ideal radio communication range is assumed. These challenges have hindered the deployment of various proposed geometric routing algorithms.

Towards the applicability of algorithms on VRAC

In this thesis, we investigated a efficient localization scheme; namely VRAC, which provides the nodes coordinates in a simple manner. VRAC is an anchor based localization scheme, where anchors are places across the network, In chapter 4.2, we proposed a several geometric routing algorithms over VRAC, considering different configurations and conditions in the node distribution in the network. Chapter 4.4 presents a combined greedy face routing algorithm, which provides delivery guarantees on UDG connectivity graphs. In chapter 4.7, we investigated the criterion where greedy routing can be performed on VRAC. Finally we establish a connection between a greedy embedding and a Schnyder drawing.

Algorithms, we designed have to rely on the Schnyder planar graph, which can only be computed given an underlying UDG connectivity. Nevertheless, we emphasize the effectiveness of VRAC compared to the most of the other proposals in the literature, where distributed computations of coordinates are required. Therefore we believe that VRAC is an effective solution to the problem of coordinates in geometric routing and further work on algorithms will lead to a practical deployment.

Applying greedy zone routing in practice

In chapter 5.2, we propose a scalable and robust greedy routing algorithm. This protocol relies on the hypothesis that the geometry of a graph has to be in an abstract level, rather than in an individual node level. It leads to more robust greedy embedding, where individual link failures does not disturb the greedy routing. We base this protocol on a tree based greedy embedding protocol, which is one of the most practical greedy routing protocols proposed [Herzen et al., 2011]. We show that this scheme become impractical when the network is large, due to the length of the coordinate.

Possible future work is to implement the greedy zone routing in a realistic setting, perhaps to support IP routing. This can be compared with a RPL instance and investigate the scalability. For instance, an IOT scenario such as Syndesi [Evangelatos et al., 2013] can be considered, where a platform to control and monitor IOT devices over an IP network. A crucial question to answer in this direction of research is, up to which number of nodes, RPL can provide the required quality of service.

A Some Notions from Graph Theory

We consider a graph $G(V,E)$ with the set of vertices V and set of edges E, such that $(u,v) \in E$ such that $u, v \in V$. In general, $G(V,E)$ is normally considered as an undirected graph, where the direction of the edge is not insignificant. Subsequently, directed graphs are graphs where the edge set consists of ordered pairs of vertices.

A.1 Paths and Connectivity

A path in a graph is an essential concept for routing in networks.

Definition 22 (**Path**)**.** *A path between two vertices u_0 and u_{k-1} in a graph $G(V,E)$ is a sequence of vertices $< u_0, u_1, ..., u_{k-1} >$ such that $(u_i, u_{i+1}) \in E$.*

The length of the path is the number of edges in the path $< u_0, u_1, ..., u_{k-1} >$, which can also be refered to as the distance between the nodes u_0 and u_{k-1}, which is denoted as $d(u_0, u_{k-1})$. A path contains a *cycle* if it repeatedly contains a node.

Definition 23 (**Shortest Path**)**.** *A path between two vertices s and t in a graph $G(V,E)$ is the shortest path if and only if, there is no other path between s and t such that $d(s,t)$ is shorter.*

Definition 24 (**Diameter**)**.** *Diameter D of a graph is the longest shortest path in a graph.*

Connectivity is an important concept in graph theory, defined as below. A *connected graph* is a graph where, between every pair of nodes there exists a simple path.

Definition 25. *A graph is k-connected, if and only if the graph does not get disconnected after removing k nodes.*

A connected graph, can be referred to as $1 - connected$.

A.2 Some special graphs

There are important classes of graphs, which are frequently used in modelling networks.

Definition 26 (**Tree**)**.** *$T(V,E)$ is a graph, where for each pair of nodes (u,v), there exist a unique path between them.*

In a tree, there is a unique node referred to as the *root* of the tree. With respect to the root of the tree, all the other nodes are referred to as *child nodes.* The distance from the root node to a given node is the depth of that node.

Definition 27 (**Spanning tree**)**.** *$T(V,E\prime)$ of a graph $G(V,E)$ is a spanning tree of G, if it contains all the nodes and $E\prime \subseteq E$.*

Definition 28 (**Directed acyclic graph**)**.** *$G(V,E)$ is a graph, where there are no directed cycles in it.*

Another important class of graphs is triangulations (or maximal planar graphs). In a combinatorial view point, a triangular graph can be defined as follows.

Definition 29 (**Directed acyclic graph**)**.** *$G(V,E)$ is a graph, where every bounded face is a triangle*

Note that an embedding of such a graph, can be drawn on the plane as a triangular drawing.

B Publications

Conferences and Workshops

1. **Samarasinghe, K.**, Leone, P.: Geometric Routing with Minimal Local Geometry, *In IEEE International Conference on Parallel and Distributed Systems* ICPADS 2012
2. **Samarasinghe, K.**, Leone, P.: Combinatorial approach for geographic routing with delivery guarantees, *In International Conference on Sensor Networks* SENSORNETS 2014
3. Leone, P, **Samarasinghe, K.**: Greedy Routing on Virtual Raw Anchor Coordinate (VRAC) System, *In IEEE International Conference on Distributed Computing in Sensor Systems* DCOSS 2016
4. Leone, P, **Samarasinghe, K.**: Every Schnyder Drawing is a Greedy Embedding, arXiv:1609.04173v1
5. **Samarasinghe, K.** and Leone, P. Greedy Zone Routing: Scalable Routing in Large Scale Wireless Ad-hoc Networks, *In IEEE International Conference on Sensing, Communication and Networking 2015* SECON 2015
6. **Samarasinghe, K.** and Leone, P. Greedy Zone Routing: Robust and Scalable Routing in Wireless Ad-hoc Networks, *In IEEE International Conference on Advanced Information Networking and Applications* AINA 2016
7. O. Evangelatos, **K. Samarasinghe**, J. Rolim: Syndesi: A Framework for Creating Personalized Smart Environments Using Wireless Sensor Networks, *In IEEE International Conference on Distributed Computing in Sensor Systems* DCOSS 2013
8. O. Evangelatos, **K. Samarasinghe**, J. Rolim: Evaluating design approaches for smart building systems, *IEEE International Conference on Mobile Ad hoc and Sensor Systems* MASS 2013

Journals

1. Leone, P., **Samarasinghe, K.**: Geographic Routing on Virtual Raw Anchor Coordinate Systems, *Theoretical computer science '15, Elsevier* 2015

Bibliography

Patrizio Angelini, Fabrizio Frati, and Luca Grilli. An algorithm to construct greedy drawings of triangulations. *Journal of Graph Algorithms and Applications*, 14(1):19–51, 2010.

James Aspnes, David Goldenberg, and Yang Yang. On the computational complexity of sensor network localization. *Algorithmic Aspects of Wireless Sensor Networks*, pages 32–44, 2004.

Prosenjit Bose and Pat Morin. Online routing in triangulations. *SIAM journal on computing*, 33(4):937–951, 2004.

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.

Prosenjit Bose, Paz Carmi, and Stephane Durocher. Bounding the locality of distributed routing algorithms. *Distributed computing*, 26(1):39–58, 2013.

Nirupama Bulusu, John Heidemann, and Deborah Estrin. Gps-less low-cost outdoor localization for very small devices. *IEEE personal communications*, 7(5):28–34, 2000.

Mirela Ben Chen, Craig Gotsman, and Camille Wormser. Distributed computation of virtual coordinates. In *Proceedings of the twenty-third annual symposium on Computational geometry*, SCG'07, pages 210–219. ACM, 2007.

Thomas H.. Cormen, Charles Eric Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*, volume 6. MIT press Cambridge, 2001.

Raghavan Dhandapani. Greedy drawings of triangulations. *Discrete & Computational Geometry*, 43(2):375–392, 2010.

Reinhard Diestel. *Graph theory*. Graduate texts in mathematics. Springer, New York, Berlin, Paris, 1997. ISBN 0-387-98210-8.

Michael B Dillencourt. Realizability of delaunay triangulations. *Information Processing Letters*, 33(6):283–287, 1990.

Adam Dunkels et al. Efficient application integration in ip-based sensor networks. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 43–48. ACM, 2009.

Simon Duquennoy, Olaf Landsiedel, and Thiemo Voigt. Let the tree bloom: scalable opportunistic routing with orpl. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SENSYS'13, pages 2:1–2:14. ACM, 2013.

David Eppstein and Michael T Goodrich. Succinct greedy graph drawing in the hyperbolic plane. In *Graph Drawing*, pages 14–25. Springer, 2009.

Orestis Evangelatos, Kasun Samarasinghe, and Jose Rolim. Evaluating design approaches for smart building systems. In *IEEE International Conference on Mobile Ad-Hoc and Sensor Systems*, MASS'12, pages 1–7. IEEE, 2012.

Orestis Evangelatos, Kasun Samarasinghe, and Jose Rolim. Syndesi: A framework for creating personalized smart environments using wireless sensor networks. In *IEEE International Conference on Distributed Computing in Sensor Systems*, DCOSS'13, pages 325–330. IEEE, 2013.

Qing Fang, Jie Gao, Leonidas J Guibas, Vin De Silva, and Li Zhang. Glider: Gradient landmark-based distributed routing for sensor networks. In *IEEE International Conference on Computer Communications*, INFOCOM'05, pages 339–350. IEEE, 2005.

Roland Flury, Sriram V Pemmaraju, and Roger Wattenhofer. Greedy routing with bounded stretch. In *IEEE Conference on Computer Communications*, INFOCOM'09, pages 1737–1745. IEEE, 2009.

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 *NSDI 2005. 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, NSDI'05, pages 329–342. USENIX Association, 2005.

Hannes Frey and Ivan Stojmenovic. On delivery guarantees and worst-case forwarding bounds of elementary face routing components in ad hoc and sensor networks. *Computers, IEEE Transactions on*, 59(9):1224–1238, 2010.

K Ruben Gabriel and Robert R Sokal. A new statistical approach to geographic variation analysis. *Systematic Biology*, 18(3):259–278, 1969.

Michael T Goodrich and Darren Strash. Succinct greedy geometric routing in the euclidean plane. In *Algorithms and Computation*, pages 781–791. Springer, 2009.

Zygmunt J Haas. A new routing protocol for the reconfigurable wireless networks. In *Universal Personal Communications Record, 1997. Conference Record., 1997 IEEE 6th International Conference on*, volume 2, pages 562–566. IEEE, 1997.

Xin He and Huaming Zhang. Schnyder greedy routing algorithm. In *Theory and Applications of Models of Computation*, pages 271–283. Springer, 2010.

Xin He and Huaming Zhang. On succinct convex greedy drawing of 3-connected plane graphs. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, SODA'11, pages 1477–1486. SIAM, 2011.

Xin He and Huaming Zhang. A simple routing algorithm based on schnyder coordinates. *Theoretical Computer Science*, 494:112–121, 2013.

Julien Herzen, Cedric Westphal, and Patrick Thiran. Scalable routing easy as pie: A practical isometric embedding protocol. In *19th International Conference on Network Protocols, IEEE*, ICNP'09, pages 49–58. IEEE, 2011.

Rein Houthooft, Sahel Sahhaf, Wouter Tavernier, Filip De Turck, Didier Colle, and Mario Pickavet. Robust geometric forest routing with tunable load balancing. In *IEEE Conference on Computer Communications*, INFOCOM'15, pages 1382–1390. IEEE, 2015.

Florian Huc, Aubin Jarry, Pierre Leone, and José Rolim. Virtual raw anchor coordinates: a new localization paradigm. In *Algorithms for Sensor Systems*, pages 161–175. Springer, 2010.

Florian Huc, Aubin Jarry, Pierre Leone, and Jose Rolim. Efficient graph planarization in sensor networks and local routing algorithm. In *IEEE International Conference on Distributed Computing in Sensor Systems*, DCOSS'12, pages 140–149. IEEE, 2012.

Mark L Huson and Arunabha Sen. Broadcast scheduling algorithms for radio networks. In *IEEE Military Communications Conference*, volume 2 of *MILCOM'95*, pages 647–651. IEEE, 1995.

Brad Karp and Hsiang-Tsung Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, MOBICOM'00, pages 243–254. ACM, 2000.

Robert Kleinberg. Geographic routing using hyperbolic space. In *IEEE International Conference on Computer Communications*, INFOCOM'07, pages 1902–1909. IEEE, 2007.

Fabian Kuhn, Rogert Wattenhofer, and Aaron Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 267–278. ACM, 2003.

Tom Leighton and Ankur Moitra. Some results on greedy embeddings in metric spaces. *Discrete & Computational Geometry*, 44(3):686–705, 2010.

Pierre Leone and Kasun Samarasinghe. Greedy routing on virtual raw anchor coordinate (vrac) system. In *International Conference on Distributed Computing in Sensor Systems*, DCOSS'16, pages 52–58. IEEE, 2016.

Pierre Leone, Luminita Moraru, Olivier Powell, and Jose Rolim. Localization algorithm for wireless ad-hoc sensor networks with traffic overhead minimization by emission inhibition. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, ALGOSENSOR'06, pages 119–129. Springer, 2006.

Ben Leong, Barbara Liskov, and Robert Morris. Geographic routing without planarization. In *3rd Conference on Networked Systems Design & Implementation-Volume 3*, NSDI'06, pages 25–25. USENIX Association, 2006.

Yujun Li, Yaling Yang, and Xianliang Lu. Rules of designing routing metrics for greedy, face, and combined greedy-face routing. *Mobile Computing, IEEE Transactions on*, 9(4):582–595, 2010.

Kevin M Lillis and Sriram V Pemmaraju. On the efficiency of a local iterative algorithm to compute delaunay realizations. In *International Workshop on Experimental and Efficient Algorithms*, pages 69–86. Springer, 2008.

Takao Nishizeki and Md Saidur Rahman. *Planar graph drawing*, volume 12. World Scientific, 2004.

Christos H Papadimitriou and David Ratajczak. On a conjecture related to geometric routing. In *Algorithmic Aspects of Wireless Sensor Networks*, pages 9–17. Springer, 2004.

Mathew Penrose. *Random geometric graphs*. Number 5. Oxford University Press, 2003.

Charles E Perkins and Elizabeth M Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, page 90. IEEE Computer Society, 1999.

Radia Perlman. An algorithm for distributed computation of a spanningtree in an extended lan. In *ACM SIGCOMM Computer Communication Review*, volume 15, pages 44–53. ACM, 1985.

Ananth Rao, Sylvia Ratnasamy, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic routing without location information. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, MOBICOM'03, pages 96–108. ACM, 2003.

Kasun Samarasinghe and Pierre Leone. Geographic routing with minimal local geometry. In *IEEE International Conference on Parallel and Distributed Systems*, ICPADS'12, pages 901–906. IEEE, 2012.

Kasun Samarasinghe and Pierre Leone. Combinatorial approach for geographic routing with delivery guarantees. In *International Conference on Sensor Networks*, SENSORNETS'14, pages 195–204, 2014.

Kasun Samarasinghe and Pierre Leone. Greedy zone routing: Scalable routing in large scale wireless ad-hoc networks. In *IEEE International Conference on Sensing, Communication, and Networking*, SECON'15, pages 172–174. IEEE, 2015.

Walter Schnyder. Planar graphs and poset dimension. *Order*, 5(4):323–343, 1989.

Walter Schnyder. Embedding planar graphs on the grid. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '90, pages 138–148, 1990.

Yi Shang and Wheeler Ruml. Improved mds-based localization. In *IEEE International Conference on Computer Communications*, volume 4 of *INFOCOM'04*, pages 2640–2651. IEEE, 2004.

Yi Shang, Wheeler Ruml, Ying Zhang, and Markus PJ Fromherz. Localization from mere connectivity. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 201–212. ACM, 2003.

Zach Shelby, Klaus Hartke, and Carsten Bormann. The constrained application protocol (coap). Technical report, 2014.

Godfried T Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern recognition*, 12(4):261–268, 1980.

Cedric Westphal and Guanhong Pei. Scalable routing via greedy embedding. In *IEEE Conference on Computer Communications*, INFOCOM'09, pages 2826–2830. IEEE, 2009.

T. Winter, A. B. P. Thubert, T. Clausen, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and J. Vasseur. RPL: IPv6 Routing Protocol for Low Power and Lossy Networks,(RFC 6550). *IETF ROLL WG, Tech. Rep*, 2012.